Житомирський державний університет імені Івана Франка
Фізико-математичний факультет
Кафедра комп'ютерних наук та інформаційних технологій

**Оксана Наконечна, Татяна Ярмоленко,
Вікторія Алексеєнко, Богданна Якимчук**

Інструктивно-методичні рекомендаціїї до
лабораторних робіт з дисципліни

«Технології розподілених систем та паралельних обчислень»

Житомир 2023

УДК 004.42/.421

М 54

Рекомендовано до друку вченою радою Житомирського державного університету імені Івана Франка
(протокол № 2 від 27 січня 2023 р.)

Рецензенти:

ГУК Віталій – кандидат технічних наук, старший викладач кафедри програмного забезпечення автоматизованих систем Черкаського національного університету ім. Б. Хмельницького.

КОЛЕСНІКОВА Ірина – кандидат педагогічних наук, старший викладач кафедри методики викладання навчальних предметів КЗ «Житомирський обласний інститут післядипломної педагогічної освіти» Житомирської обласної ради.

ЩЕХОРСЬКИЙ Анатолій – кандидат фізико-математичних наук, доцент, доцент кафедри комп'ютерних наук та інформаційних технологій Житомирського державного університету імені Івана Франка.

М 54 **Наконечна О. А., Ярмоленко Т. А., Алексеєнко В. В., Якимчук Б. М.** Інструктивно-методичні рекомендації з дисципліни «Технології розподілених систем та паралельних обчислень» / уклад.: Оксана Наконечна, Тетяна Ярмоленко, Вікторія Алексеєнко, Богданна Якимчук. Житомир: Житомир: Вид-во ЖДУ ім. Івана Франка, 2023. 74 с.

Інструктивно-методичні рекомендації до лабораторних робіт передбачають оволодіння майбутніми фахівцями необхідними компетентностей та програмних результатів навчання та формує відповідність професійної підготовки фахівця запитам сучасного суспільства. Лабораторні завдання містять теоретичні відомості, план виконання завдань з вказівками.

УДК 004.42/.421

Зміст

Вступ

У рамках послідовних принципів обробки даних можливості подальшого нарощування продуктивності комп'ютерів практично вичерпали себе, що обумовлено в основному скінченою швидкістю розповсюдження сигналів.

Пошук вирішення проблеми підвищення продуктивності йде в напрямку розвитку принципів паралельної обробки інформації. Отримати суттєвий приріст в продуктивності можна лише у використанні принципово нових комп'ютерних архітектур, які основані на паралельній обробці даних.

Основні принципи паралелізму були впроваджені ще в перші експериментальні паралельні машини, які з'явились в 60-70 роках минулого століття. У векторній CRAY-1, матричній ILLIAC-4, ортогональній OMEN та в інших комп'ютерах тих часів були започатковані основні напрямки паралельних обчислень: конвеєризація, процесорні матриці, асоціативна адресація.

В наші дні принципи паралелізму використовуються в більшості обчислювальних пристроїв, найбільш поширеними паралельними комп'ютерами є кластерні системи та багатоядерні персональні комп'ютери. Прогрес в обчислювальній техніці викликав інтенсивний розвиток відповідного програмного забезпечення. Утворився окремий розділ в програмуванні – паралельне програмування.

Тема 1. Основні теорії паралельних обчислень

Лабораторна робота №1. Архітектура паралельних обчислювальних систем.

Мета: Ознайомити студентів з архітектурою паралельних обчислювальних систем.

Основні поняття: мультипроцесори, мультикомп'ютери, класифікація Флінна, SISD, SIMD, MISD, MIMD.

План

1. Мультипроцесори.
2. Мультикомп'ютери.
3. Класифікація комп'ютерних систем.
   3.1 Класифікація Флінна (1966).
   3.2 Класифікація комп'ютерів паралельної дії.

І. Теоретичні відомості

Мультипроцесори

Паралельний комп'ютер, у якому всі процесори спільно використовують загальну фізичну пам'ять, називається *мультипроцесором*, або *системою з спільною пам'яттю*. (Рис. 1.1) Усі процеси, що спільно працюють у мультипроцесорі, можуть мати єдиний віртуальний адресний простір, що відображений на спільну пам'ять. Будь-який процес може зчитати слово з пам'яті чи записати слово в пам'ять. Більше нічого не потрібно. Два процеси мають можливість легко обмінюватися інформацією - для цього один з них просто записує дані у спільну пам'ять, а інший їх зчитує.

(**Адреса** — символ чи група символів, які ідентифікують регістр, окремі частини пам'яті чи деякі інші джерела даних чи місце розміщення інформації.)

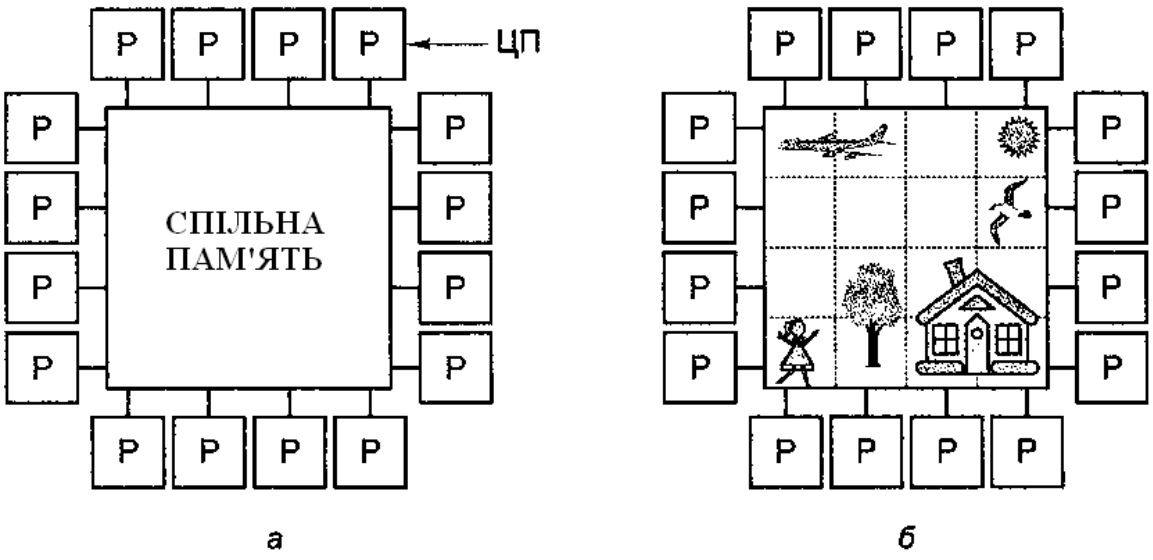


Рис. 1.1. Мультипроцесор з 16 процесорів, що мають спільну пам'ять (а); зображення, розділене на 16 секцій, кожну з яких аналізує окремий процесор (б)

Завдяки можливості взаємодії двох і більше процесів мультипроцесори вельми популярні. Дана модель зрозуміла програмістам і дозволяє вирішувати широке коло завдань.

Оскільки всі процесори в мультипроцесорі використовують єдиний адресний простір, функціонує тільки одна копія операційної системи. Відповідно, є тільки одна карта сторінок пам'яті і одна таблиця процесів. Коли процес блокується, його процесор зберігає свій стан в таблицях операційної системи, а потім переглядає ці таблиці в пошуках іншого процесу, який потрібно запустити. Саме така організація, в основі якої лежить єдина система, і відрізняє мультипроцесор від мультикомп'ютера.

Якщо всі процесори мають рівний доступ до всіх модулів пам'яті і до всіх пристроїв введення-виведення, і між процесорами можлива повна взаємозамінність, такий мультипроцесор називається *симетричним* (Symmetrie Multiprocessor, SMP).

Мультикомп'ютери

У другому варіанті паралельної архітектури кожен процесор має власну пам'ять, доступну тільки цьому процесору. Така схема називається *мультикомп'ютером*, або *системою з розподіленою пам'яттю* (Рис. 1.2, а). Ключова відмінність мультикомп'ютера від мультипроцесора полягає в тому, що кожен процесор у мультиком'ютері має власну локальну пам'ять, до якої цей процесор може звертатися, виконуючи команди LOAD і STORE, але ніякий інший процесор за допомогою цих команд не може отримати доступ до локальної пам'яті даного процесора. Таким чином, мультипроцесори мають один фізичний адресний простір, що розподілений між всіма процесорами, а мультикомп'ютери містять окремі фізичні адресні простори для кожного процесора.

Оскільки процесори в мультикомп'ютері не можуть взаємодіяти один з одним простими зверненнями до спільної пам'яті, процесори обмінюються повідомленнями через комунікаційну мережу, що їх пов'язує. У якості прикладів мультикомп'ютерів можна назвати IBM BlueGene / L, Red Storm і кластер Google.

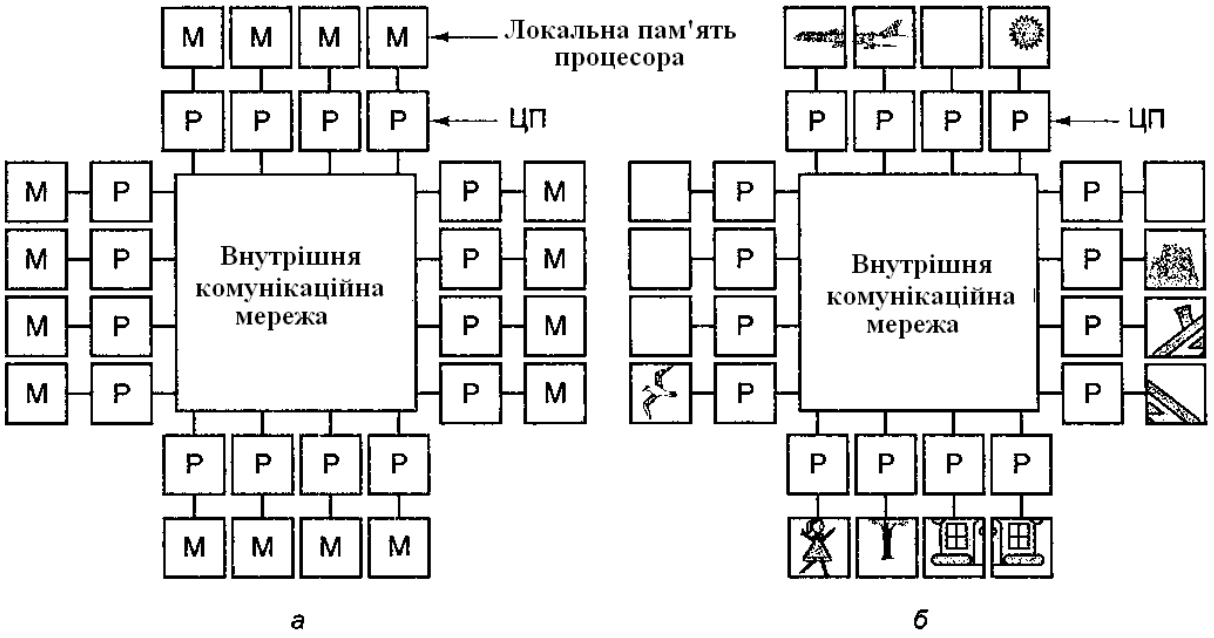


Рис. 1.2. Мультикомп'ютер з 16 процесорів, кожний з яких має власну пам'ять (а); бітова карта зображення з рис. 1, розділена між 16 модулями пам'яті (б)

Виникає питання: навіщо взагалі створювати мультикомп'ютери, якщо мультипроцесори набагато простіше програмувати? Відповідь проста: створити великий мультикомп'ютер простіше і дешевше, ніж мультипроцесор з такою ж кількістю процесорів. Реалізація загальної пам'яті, спільно використовуваної кількома сотнями процесорів, - це дуже складна задача, а розробити мультикомп'ютер, що містить 10000 процесорів і більше, досить легко.

Таким чином, ми стикаємося з дилемою: мультипроцесори складно розробляти, але легко програмувати, а мультикомп'ютери легко будувати, але важко програмувати. У результаті постійно робляться спроби створення гібридних систем. Ці спроби призвели до усвідомлення того факту, що спільну пам'ять можна реалізовувати по-різному, причому кожен варіант буде мати переваги і недоліки. Практично всі дослідження в галузі паралельних комп'ютерних архітектур спрямовані на створення гібридних форм, які поєднують у собі переваги обох систем. Тут важливо домогтися масштабованості, тобто розробити таку систему, яка буде продовжувати справно працювати при додаванні все нових і нових процесорів.

Класифікація комп'ютерних систем

Класифікація Флінна (1966)

За багато років було запропоновано і побудовано безліч видів паралельних комп'ютерних систем, тому хотілося б їх як-небудь класифікувати. Це з різними результатами намагалися робити багато дослідників, але, на жаль, хорошої класифікації досі немає. Найчастіше використовують класифікацію Флінна (1966), але навіть вона є досить грубої (*Таблиця 1.1*).

Класифікація заснована на понятті потоку, під яким розуміється послідовність команд або даних, оброблюваних процесором. На основі числа потоків команд і потоків даних виділяють чотири класи архітектур.

Таблиця 1.1. Класифікація паралельних комп'ютерних систем за Флінном

Потоки команд	Потоки даних	Категорія	Приклад
Один	Один	SISD	Класична машина фон-Неймана
Один	Багато	SIMD	Векторний суперкомп'ютер, матричний процесор
Багато	Один	MISD	?
Багато	Багато	MIMD	Мультипроцесор, мультикомп'ютер

SISD (Single Instruction stream Single Data stream - один потік команд з одним потоком даних) - це класична послідовна комп'ютерна архітектура фон Неймана. Комп'ютер фон Неймана має один потік команд і один потік даних і в кожний момент часу може виконувати тільки одну дію.

У машини, що відносяться до категорії *SIMD* (Single Instruction-stream Multiple Data-stream - один потік команд з декількома потоками даних), є один блок управління, що видає по одній команді, але при цьому є декілька АЛП (арифметико-логічний пристрій), які можуть обробляти кілька наборів даних одночасно . Прототип SIMD-машин - процесор ILLIAC IV. Хоча SIMD-машини не відносяться до числа широко розповсюджених, в деяких звичайних комп'ютерах для обробки мультимедійних даних використовуються SIMD-команди. У будь-якому випадку існує одна область, де ідеї, почерпнуті з "світу SIMD", виходять на перший план, - це потокові процесори.

MISD (Multiple Instruction-stream Single Data-stream - кілька потоків команд з одним потоком даних) - досить дивна категорія. Тут кілька команд оперують одним набором даних. Важко сказати, чи існують такі машини, хоча дехто відносять до категорії MISD машини з конвеєрами.

Остання категорія – *MIMD* (Multiple Instruction-stream Multiple Data-stream – кілька потоків команд з декількома потоками даних). Тут кілька незалежних процесорів працюють як частина великої системи. У цю категорію потрапляють більшість паралельних процесорів. І мультипроцесори, і мультикомп'ютери - це MIMD-машини.

Класифікація комп'ютерів паралельної дії

Розширимо класифікацію Флінна (рис. 1.3). SIMD-машини у нас поділяються на дві підгрупи. У першу підгрупу потрапляють численні суперкомп'ютери та інші машини, які оперують векторами, виконуючи одну і ту ж операцію над кожним елементом вектора. В другу підгрупу

потрапляють машини типу ILLIAC IV, в яких головний блок управління посилає команди декільком незалежним АЛП.

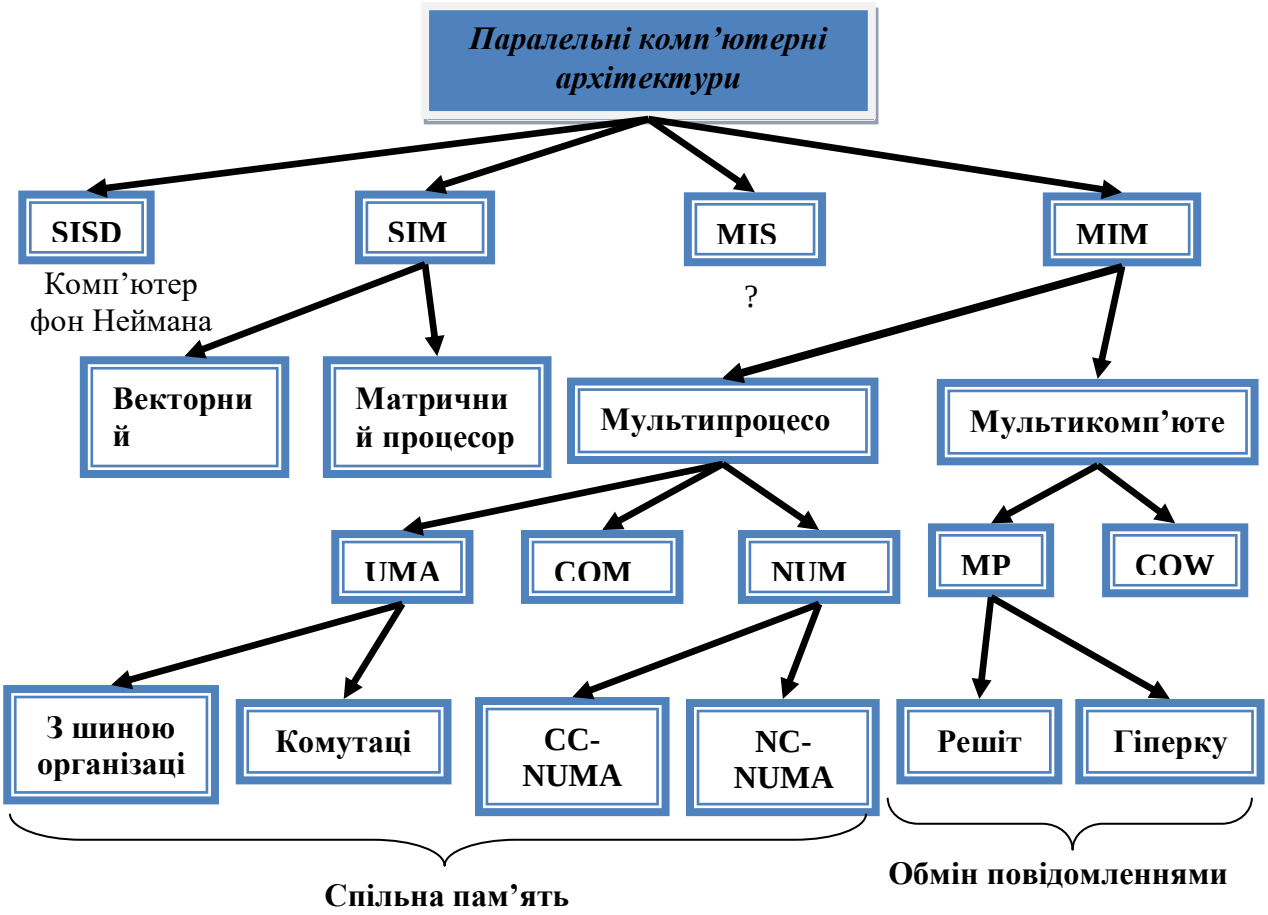


Рис. 1.3. Класифікація комп'ютерів паралельної дії

У нашій класифікації категорія MIMD поділена на мультипроцесори (машини зі спільною пам'яттю) і мультикомпьютери (машини з обміном повідомленнями). Існує три типи мультипроцесорів. Вони відрізняються один від одного механізмом доступу до спільної пам'яті і називаються *UMA* (Uniform Memory Access - однорідний доступ до пам'яті), *NUMA* (Nonuniform Memory Access - неоднорідний доступ до пам'яті) і *COMA* (Cache Only Memory Access - доступ тільки до кеш-пам'яті) . Таке розбиття на підкатегорії має сенс, оскільки у великих мультипроцесорах пам'ять зазвичай ділиться на декілька модулів.

У UMA-машинах кожен процесор має один і той ж час доступу до будь-якого модулю пам'яті. Іншими словами, кожне слово може бути зчитане з пам'яті з тією ж швидкістю, що і будь-яке інше слово. Якщо це технічно неможливо, найшвидші звернення сповільнюються, щоб відповідати самим повільним, тому програміст не помітить ніякої різниці. Це й означає "однорідний" доступ. Така однорідність робить продуктивність передбачуваною, а цей фактор дуже важливий для створення ефективних програм.

NUMA-машина, навпаки, не володіє властивістю однорідності. Зазвичай у кожного процесора є один з модулів пам'яті, який розташовується

до нього ближче, ніж інші, тому доступ до цього модуля пам'яті відбувається набагато швидше, ніж до інших. У цьому випадку з погляду продуктивності дуже важливо, де опиняться програма і дані. Доступ до COMA-машинам теж виявляється неоднорідним, але з іншої причини.

Мультикомп’ютери теж можна розділити на дві додаткові категорії. До категорії MPP (Massively Parallel Processor - процесор з масовим паралелізмом) відносяться дорогі суперкомп'ютери, які складаються з великої кількості процесорів, пов'язаних високошвидкісною внутрішньої комунікаційною мережею. У якості добре відомого комерційного прикладу можна назвати суперкомп'ютер SP/3 компанії IBM.

Друга категорія мультикомп’ютерів включає звичайні персональні комп'ютери або робочі станції, які зв'язуються відповідно до тієї чи іншої комерційної комунікаційної технології. З точки зору логіки принципової різниці тут немає, але потужний суперкомп'ютер вартістю у мільйони доларів безумовно використовується інакше, ніж зібрана кінцевими користувачами комп'ютерна мережа, яка обходиться у багато разів дешевше будь-якої MPP-машини. Ці системи іноді називають мережами робочих станцій (Network Of Workstations, NOW), кластерами робочих станцій (Cluster Of Workstattions, COW), або просто кластерами (cluster).

II. Практична частина

Інструкція до виконання

1. Опрацювати теоретичні відомості.
2. Скласти міні-конспект відповідно до переліку тем:
   a) Кластери.
   b) Комунікаційне середовище паралельних обчислень.
   c) Нейрокомп’ютерні технології.
   d) Обчислювальні комп’ютери з розподіленою пам’яттю (мультикомп’ютери)
   e) Паралельні комп’ютери з спільною пам’яттю (мультипроцесори).
   f) Гібридні паралельні архітектури.

Поточні контрольні питання:

1. Що таке мультипроцесори?
2. Що таке мультиком’ютери?
3. Яка основна характеристика класифікації Флінна?
4. Що таке SISD?
5. Що таке SIMD?
6. Що таке MISD?
7. Що таке MIMD?
8. До якої категорії за класифікацією комп’ютерів паралельної дії належать кластери?

Рекомендована література:

1. А.Ю. Дорошенко, В.М. Кислоокий, О.Л. Синявський. Архітектура і операційні середовища комп’ютерних систем. Методичний посібник і конспект лекцій.- Київ: НаУКМА, 2005

2. Паралельні та розподілені обчислення [Текст] : навч. підруч. для студентів вищ. навч. закл. / А. Луцків, С. Луценко, В. Пасічник. – Львів : Магнолія 2006, 2017. – 565, [1] с. : схеми.

3. Аксак Н.Г. Паралельні та розподілені обчислення: підручник / Н.Г. Аксак, О.Г. Руденко, А.М. Гуржій. – Х.: Компанія СМІТ, 2009. – 480с.
4. Згуровський М.З., Петренко А.І. Е-наука на шляху до семантичного Грід. Частина 1: Об'єднання Web- і Грід- технологій // Системні дослідження та інформаційні технології. - Київ, №1, 2010. - С. 26-38.

Тема 2. Граф як модель обчислювальних процесів

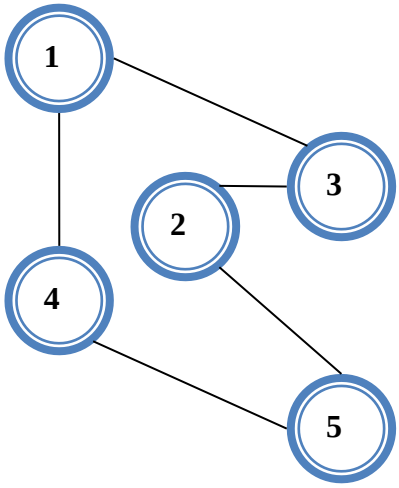
Лабораторна робота №2. Послідовний алгоритм Флойда-Уоршела

Мета: Ознайомити студентів з послідовним алгоритмом Флойда-Уоршела.

План

1. Побудова графу за заданою матрицею суміжності.
2. Знаходження найкоротшого шляху на заданому графі з використанням алгоритму Флойда-Уоршелла.
3. Розробка програми для знаходження найкоротших і найдовших шляхів.

I. Теоретичні відомості



Неорієнтований граф

Поняття графу

Граф або *неорієнтований граф* G — це впорядкована пара $G := (V, E)$, для якої виконуються наступні умови:

- V — множина *вершин* або *вузлів*,
- E — множина пар (у випадку неорієнтованого графу — невпорядкованих) вершин, які називають *ребрами*.

Граф (геометричний граф) — це фігура на площині, яка складається з непорожньої скінченної множини V точок (вершин) і скінченної множини E орієнтованих чи не орієнтованих ліній (ребер), що з'єднують деякі пари вершин.

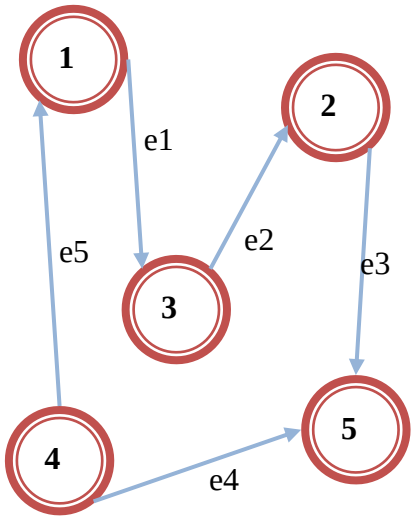
Маршрутом орграфа називають послідовність вершин і *дуг*, виду $v_0\{v_0, v_1\}v_1\{v_1, v_2\}v_2 \ldots v_n$ (вершини можуть повторюватися).

Довжина маршруту — кількість дуг у ньому.

Шлях — *маршрут* орграфа без повторюваних дуг, *простий шлях* — без повторюваних вершин. Якщо існує шлях з однієї вершини в іншу, то друга вершина *досяжна* з першої.

Контур — замкнений *шлях*.

Орієнтований граф (коротко *орграф*) — (мульти) граф, ребрам якого присвоєно напрямок. Орієнтовані ребра називаються також *дугами*, а в деяких джерелах і просто ребрами.



Орієнтований граф

Способи завдання графів

1. Графічний спосіб.
2. Списком ребер графа: $e_1(1,3), e_2(3,2), e_3(2,5), e_4(4,5), e_5(4,1)$.

3. Матрицею суміжності

Матриця суміжності графа G зі скінченною кількістю вершин n (пронумерованих числами від 1 до n) — це квадратна матриця A розміру n, в якій значення елементу a_{ij} рівне числу ребер з i-ї вершини графа в j-у вершину.

$$A=\begin{pmatrix}0&0&1&0&0\\0&0&0&0&1\\0&1&0&0&0\\1&0&0&0&1\\0&0&0&0&0\end{pmatrix},\quad a_{i,j}=\begin{cases}1, ÿêùî\ (v_i,v_j)\in E,\\ 0, ³íàêøå\ .\end{cases}$$

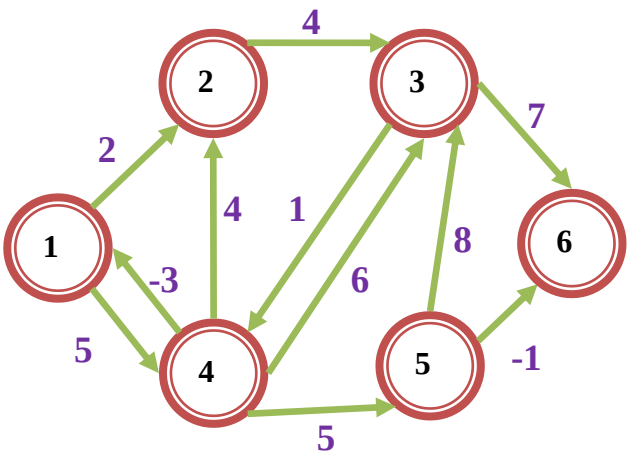
4. Матрицею інцидентності

Ма́триця інциде́нтності — одна з форм представлення графа, в якій вказуються зв'язок між інцидентними елементами графа (ребро (дуга) і вершина). Стовпці матриці відповідають ребрам, рядки — вершинам. Ненульове значення у клітинці матриці вказує на зв'язок між вершиною і ребром (їх інцедентність).

$$B=\begin{pmatrix}1&0&0&0&-1\\0&-1&1&0&0\\-1&1&0&0&0\\0&0&0&1&1\\0&0&-1&-1&0\end{pmatrix},\quad b_{i,j}=\begin{cases}1, & \text{якщо ребро виходить з } v_i \text{ і входить у } v_j,\\ -1, & \text{якщо ребро входить у } v_i \text{ і виходить з } v_j\\ 0. & \text{якщо між } v_i \text{ і } v_j \text{ немає ребра.}\end{cases}$$

Нехай G –граф $G=(V,E)$, для якого набір вершин $v_i, 1\le i\le n$, задається множиною V, а список дуг графа $e_j=(v_{s_i},v_{t_j}), 1\le j\le m$ визначається множиною E. У загальному випадку дугам графа можуть бути приписані деякі числові характеристики (ваги) $w_j, 1\le j\le m$ (*зважений граф*).

Приклад зваженого графу:



Зважений граф

Алгоритм Флойда – Уоршела

Означення**.** *Алгоритм Флойда – Уоршела* **–** це алгоритм для знаходження найкоротших відстаней між усіма вершинами зваженого графу без циклів з від'ємними вагами з використанням динамічного програмування.

Для знаходження найкоротших шляхів між усіма вершинами графа використовується не перебір всіх можливостей, що призведе до великого часу роботи і потребує більше пам'яті, а динамічне програмування, тобто всі підзадачі, які згодом знадобляться для вирішення вихідної задачі, прораховуються заздалегідь і потім використовуються.

Цей алгоритм був одночасно опублікований у статтях Роберта Флойда (Robert Floyd) та Стівена Уоршелла (Stephen Warshall) у 1962 р., хоча у 1959 році Бернард Рой (Bernard Roy) опублікував майже такий же алгоритм, але це залишилось непоміченим.

Алгоритм

Нехай вершини графа $G = (V, E)$, $|V| = n$ пронумеровані від 1 до n і введено позначення $d_{i,j}^k$ для довжини найкоротшого шляху від i до j, який окрім самих вершин i, j проходить тільки через вершини $1 \ldots k$. Очевидно, що $d_{i,j}^0$ - довжина (вага) ребра, якщо таке існує (в іншому випадку його довжина може бути позначена як ∞).

Існує два варіанти значення:

1. Найкоротший шлях між i, j не проходить через вершину k, тоді $d_{i,j}^k = d_{ij}^{k-1}$.
2. Існує більш короткий шлях між i, j, що проходить через k, тоді він спочатку йде від i до k, а потім від k до j. У цьому випадку, очевидно, $d_{i,j}^k = d_{ik}^{k-1} + d_{kj}^{k-1}$.

Таким чином, для знаходження значення функції досить вибрати мінімум з двох позначених значень.

Тоді рекурентна формула для $d_{i,j}^k$ має вигляд:

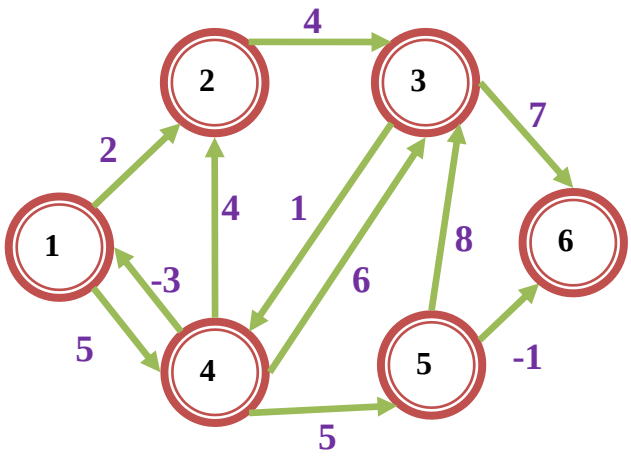
$d_{i,j}^0$ - довжина ребра (i, j);

$d_{i,j}^k = \min(d_{i,j}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$.

Алгоритм Флойда-Уоршелла послідовно обчислює всі значення $d_{i,j}^k, \forall i, j$ для k від 1 до n. Отримані значення є довжинами найкоротших шляхів між вершинами.

На кожному кроці алгоритм генерує матрицю D. Матриця містить довжини найкоротших шляхів між усіма вершинами графа. Перед роботою алгоритму матриця D заповнюється довжинами ребер графа.

Приклад:



k=0:

$$A=\begin{pmatrix}0&1&0&1&0&0\\0&0&1&0&0&0\\0&0&0&1&0&1\\1&1&1&0&1&0\\0&0&1&0&0&1\\0&0&0&0&0&0\end{pmatrix}\quad D^0=\begin{pmatrix}0&2&\infty&5&\infty&\infty\\\infty&0&4&\infty&\infty&\infty\\\infty&\infty&0&1&\infty&7\\-3&4&6&0&5&\infty\\\infty&\infty&8&\infty&0&-1\\\infty&\infty&\infty&\infty&\infty&0\end{pmatrix}$$

k=1, D^1=:

1	1	2	3	4	5	6
1	0	2	∞	5	∞	∞
2	∞	0	4	∞	∞	∞
3	∞	∞	0	1	∞	7
4	-3	-1	6	0	5	∞
5	∞	∞	8	∞	0	-1
6	∞	∞	∞	∞	∞	0

$d^1_{1,1}=\min(d^0_{1,1},d^0_{1,1}+d^0_{1,1})=0,$

Висновок, якщо $i=j$, то $d^k_{i,j}=d^k_{i,i}=\min(d^{k-1}_{i,i},d^{k-1}_{ik}+d^{k-1}_{ki})=d^{k-1}_{i,i}=0$, тобто головна діагональ залишається не змінною.

$d^1_{1,2}=\min(d^0_{1,2},d^0_{1,1}+d^0_{1,2})=\min(2,0+2)=2,$

$d^1_{1,3}=\min(d^0_{1,3},d^0_{1,1}+d^0_{1,3})=\min(\infty,0+\infty)=\infty,$

$d^1_{1,4}=\min(d^0_{1,4},d^0_{1,1}+d^0_{1,4})=\min(5,0+5)=5.$

Висновок, якщо $i=k$, то $d^k_{i,j}=d^k_{k,j}=\min(d^{k-1}_{k,j},d^{k-1}_{kk}+d^{k-1}_{kj})=d^{k-1}_{k,j}=d^{k-1}_{i,j}$, аналогічно $j=k$, $d^k_{i,j}=d^k_{i,k}=\min(d^{k-1}_{i,k},d^{k-1}_{ik}+d^{k-1}_{kk})=d^{k-1}_{i,k}=d^{k-1}_{i,j}$.

$d^1_{2,3}=\min(d^0_{2,3},d^0_{2,1}+d^0_{1,3})=\min(4,\infty+\infty)=4,$

$d^1_{2,4}=\min(d^0_{2,4},d^0_{2,1}+d^0_{1,4})=\min(\infty,\infty+5)=\infty,$

$d^1_{2,5}=\min(d^0_{2,5},d^0_{2,1}+d^0_{1,5})=\min(\infty,\infty+\infty)=\infty,$

$d^1_{2,6}=\min(d^0_{2,6},d^0_{2,1}+d^0_{1,6})=\min(\infty,\infty+\infty)=\infty,$

$d^1_{4,2} = \min(d^0_{4,2}, d^0_{4,1} + d^0_{1,2}) = \min(4, -3+2) = -1,$

$d^1_{4,3} = \min(d^0_{4,3}, d^0_{4,1} + d^0_{1,3}) = \min(6, -3+\infty) = 6.$

k=2, D^2=:

2	1	2	3	4	5	6
1	0	2	6	5	∞	∞
2	∞	0	4	∞	∞	∞
3	∞	∞	0	1	∞	7
4	-3	-1	3	0	5	∞
5	∞	∞	8	∞	0	-1
6	∞	∞	∞	∞	∞	0

$d^2_{1,3} = \min(d^1_{1,3}, d^1_{1,2} + d^1_{2,3}) = \min(\infty, 2+4) = 6,$

$d^2_{1,4} = \min(d^1_{1,4}, d^1_{1,2} + d^1_{2,4}) = \min(5, 2+\infty) = 5,$

$d^2_{3,1} = \min(d^1_{3,1}, d^1_{3,2} + d^1_{2,1}) = \min(\infty, \infty+\infty) = \infty,$

$d^2_{3,1} = \min(d^1_{3,1}, d^1_{3,2} + d^1_{2,1}) = \min(\infty, \infty+\infty) = \infty,$

$d^2_{4,3} = \min(d^1_{4,3}, d^1_{4,2} + d^1_{2,3}) = \min(6, -1+4) = 3,$

$d^2_{5,3} = \min(d^1_{5,3}, d^1_{5,2} + d^1_{2,5}) = \min(8, \infty+\infty) = 8.$

k=3, D^3=:

3	1	2	3	4	5	6
1	0	2	6	5	∞	13
2	∞	0	4	5	∞	11
3	∞	∞	0	1	∞	7
4	-3	-1	3	0	5	10
5	∞	∞	8	9	0	-1
6	∞	∞	∞	∞	∞	0

$d^3_{1,2} = \min(d^2_{1,2}, d^2_{1,3} + d^2_{3,2}) = \min(2, 6+\infty) = 2,$

$d^3_{1,4} = \min(d^2_{1,4}, d^2_{1,3} + d^2_{3,4}) = \min(5, 6+1) = 5,$

$d^3_{1,6} = \min(d^2_{1,6}, d^2_{1,3} + d^2_{3,6}) = \min(\infty, 6+7) = 13,$

$d^3_{2,4} = \min(d^2_{2,4}, d^2_{2,3} + d^2_{3,4}) = \min(\infty, 4+1) = 5,$

$d^3_{2,6} = \min(d^2_{2,6}, d^2_{2,3} + d^2_{3,6}) = \min(\infty, 4+7) = 11,$

$d^3_{4,6} = \min(d^2_{4,6}, d^2_{4,3} + d^2_{3,6}) = \min(\infty, 3+7) = 10$

$d^3_{5,4} = \min(d^2_{5,4}, d^2_{5,3} + d^2_{3,4}) = \min(\infty, 8+1) = 9,$

$d^3_{5,6} = \min(d^2_{5,6}, d^2_{5,3} + d^2_{3,6}) = \min(-1, 8+7) = -1,$

k=4, D^4=:

4	1	2	3	4	5	6
1	0	2	6	5	10	13
2	2	0	4	5	10	11
3	-2	0	0	1	6	7
4	-3	-1	3	0	5	10
5	6	8	8	9	0	-1
6	∞	∞	∞	∞	∞	0

$d^4_{1,2} = \min(d^3_{1,2}, d^3_{1,4} + d^3_{4,2}) = \min(2{,}5 + (-1)) = 2,$

$d^4_{1,3} = \min(d^3_{1,3}, d^3_{1,4} + d^3_{4,3}) = \min(6{,}5 + 3) = 6,$

$d^4_{1,5} = \min(d^3_{1,5}, d^3_{1,4} + d^3_{4,5}) = \min(\infty{,}5 + 5) = 10,$

$d^4_{1,6} = \min(d^3_{1,6}, d^3_{1,4} + d^3_{4,6}) = \min(13{,}5 + 10) = 13,$

$d^4_{2,1} = \min(d^3_{2,1}, d^3_{2,4} + d^3_{4,1}) = \min(\infty{,}5 + (-3)) = 2,$

$d^4_{2,3} = \min(d^3_{2,3}, d^3_{2,4} + d^3_{4,3}) = \min(4{,}5 + 3) = 4,$

$d^4_{2,5} = \min(d^3_{2,5}, d^3_{2,4} + d^3_{4,5}) = \min(\infty{,}5 + 5) = 10,$

$d^4_{2,6} = \min(d^3_{2,6}, d^3_{2,4} + d^3_{4,6}) = \min(11{,}5 + 10) = 11,$

$d^4_{3,1} = \min(d^3_{3,1}, d^3_{3,4} + d^3_{4,1}) = \min(\infty{,}1 + (-3)) = -2,$

$d^4_{3,2} = \min(d^3_{3,2}, d^3_{3,4} + d^3_{4,2}) = \min(\infty{,}1 + (-1)) = 0,$

$d^4_{3,5} = \min(d^3_{3,5}, d^3_{3,4} + d^3_{4,5}) = \min(\infty{,}1 + 5) = 6,$

$d^4_{3,6} = \min(d^3_{3,6}, d^3_{3,4} + d^3_{4,6}) = \min(7{,}1 + 10) = 7,$

$d^4_{5,1} = \min(d^3_{5,1}, d^3_{5,4} + d^3_{4,1}) = \min(\infty{,}9 + (-3)) = 6,$

$d^4_{5,2} = \min(d^3_{5,2}, d^3_{5,4} + d^3_{4,2}) = \min(\infty{,}9 + (-1)) = 8,$

$d^4_{5,3} = \min(d^3_{5,3}, d^3_{5,4} + d^3_{4,3}) = \min(8{,}9 + 3) = 8,$

$d^4_{5,6} = \min(d^3_{5,6}, d^3_{5,4} + d^3_{4,6}) = \min(-1{,}9 + 10) = -1,$

k=5, D^5=:

5	1	2	3	4	5	6
1	0	2	6	5	10	9
2	2	0	4	5	10	9
3	-2	0	0	1	6	5
4	-3	-1	3	0	5	4
5	6	8	8	9	0	-1
6	∞	∞	∞	∞	∞	0

$d^5_{1,2} = \min(d^4_{1,2}, d^4_{1,5}5 + d^4_{5,2}) = \min(2{,}10 + 8) = 2,$

$d^5_{1,3} = \min(d^4_{1,3}, d^4_{1,5}5 + d^4_{5,3}) = \min(6{,}10 + 8) = 6,$

$d^5_{1,4} = \min(d^4_{1,4}, d^4_{1,5}5 + d^4_{5,4}) = \min(5{,}10 + 9) = 5,$

$d^5_{1,6} = \min(d^4_{1,6}, d^4_{1,5}5 + d^4_{5,6}) = \min(13{,}10 + (-1)) = 9,$

$d^5_{2,1} = \min(d^4_{2,1}, d^4_{2,5}5 + d^4_{5,1}) = \min(2{,}10 + 6) = 2,$

$d^5_{2,3} = \min(d^4_{2,3}, d^4_{2,5}5 + d^4_{5,3}) = \min(4{,}10 + 8) = 4,$

$d^5_{2,4} = \min(d^4_{2,4}, d^4_{2,5}5 + d^4_{5,4}) = \min(5{,}10 + 9) = 5,$

$d^5_{2,6} = \min(d^4_{2,6}, d^4_{2,5}5 + d^4_{5,6}) = \min(11{,}10 + (-1)) = 9,$

$d^5_{3,1} = \min(d^4_{3,1}, d^4_{3,5}5 + d^4_{5,1}) = \min(-2{,}6 + 6) = -2,$

$d^5_{3,2} = \min(d^4_{3,2}, d^4_{3,5}5 + d^4_{5,2}) = \min(0{,}6 + 8) = 0,$

$d^5_{3,4} = \min(d^4_{3,4}, d^4_{3,5}5 + d^4_{5,4}) = \min(1{,}6 + 9) = 1,$

$d^5_{3,6} = \min(d^4_{3,6}, d^4_{3,5}5 + d^4_{5,6}) = \min(7{,}6 + (-1)) = 5,$

$d^5_{4,1} = \min(d^4_{4,1}, d^4_{4,5}5 + d^4_{5,1}) = \min(-3{,}5 + 6) = -3,$

$d^5_{4,2} = \min(d^4_{4,2}, d^4_{4,5}5 + d^4_{5,2}) = \min(-1,5+8) = -1,$

$d^5_{4,3} = \min(d^4_{4,3}, d^4_{4,5}5 + d^4_{5,3}) = \min(3,5+8) = 3,$

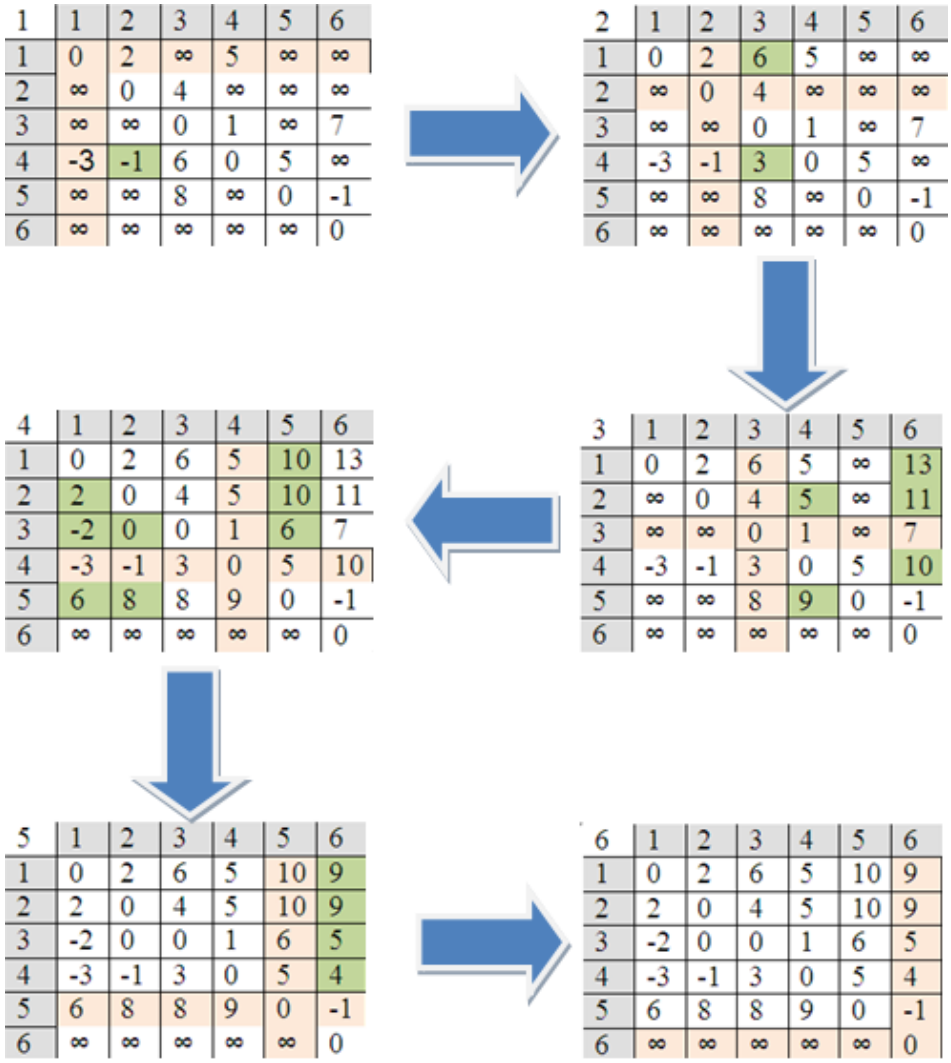
$d^5_{4,6} = \min(d^4_{4,6}, d^4_{4,5}5 + d^4_{5,6}) = \min(10,5+(-1)) = 4,$

k=6, D^6=:

6	1	2	3	4	5	6
1	0	2	6	5	10	9
2	2	0	4	5	10	9
3	-2	0	0	1	6	5
4	-3	-1	3	0	5	4
5	6	8	8	9	0	-1
6	∞	∞	∞	∞	∞	0

$d^6_{1,2} = \min(d^5_{1,2}, d^5_{1,6}5 + d^5_{6,2}) = \min(2,9+\infty) = 2$, і т. д.

Маючи усі матриці, можна скласти результуючу таблицю найкоротших шляхів.



Таблиця найкоротших шляхів:

	1	2	3	4	5	6
1	1-1	1-2	1-2-3	1-4	1-4-5	1-5-6
2	2-4-1	2-2	2-3	2-3-4	2-4-5	2-5-6

3	3-4-1	3-4-2	3-3	3-4	3-4-5	3-5-6
4	4-1	4-1-2	4-2-3	4-4	4-5	4-5-6
5	5-4-1	5-4-2	5-3	5-3-4	5-5	5-6
6	-	-	-	-	-	6-6

Результуюча таблиця найкоротших шляхів:

	1	2	3	4	5	6
1	1-1	1-2	1-2-3	1-4	1-4-5	1-4-5-6
2	2-3-4-1	2-2	2-3	2-3-4	2-3-4-5	2-3-4-5-6
3	3-4-1	3-4-1-2	3-3	3-4	3-4-5	3-4-5-6
4	4-1	4-1-2	4-1-2-3	4-4	4-5	4-5-6
5	5-3-4-1	5-3-4-1-2	5-3	5-3-4	5-5	5-6
6	-	-	-	-	-	6-6

II. Практична частина

Інструкція до виконання

1. Опрацювати теоретичні відомості.
2. Побудувати граф за заданою матрицею суміжності. Варіанти згідно *Додатку №1*.
3. Використовуючи алгоритм Флойда–Уоршелла, знайти найкоротші шляхи на заданому графі (згідно варіанту, *Додаток №2*) для будь-якої пари вершин графу.
4. Скласти програму для знаходження найкоротших шляхів на заданому графі (варіанти згідно Додатку №3) для будь-якої пари вершин графу, використовуючи а лгоритм Флойда – Уоршелла. Результати виконання програми та текст самої програми повинні бути занесені у зошит.
5. Аналогічно до алгоритму Флойда – Уоршелла скласти програму для знаходження найдовших шляхів на заданому графі (згідно варіанту, *Додаток №2*) для будь-якої пари вершин графу

Поточні контрольні питання:

1. Що таке граф?
2. Що таке орієнтований граф?
3. Що таке маршрут орфграфа?
4. Дайте означення довжини маршруту?
5. Що таке шлях?
6. Що таке контур?
7. Назвіть способи подання графів?
8. Що таке матриця суміжності?
9. Що таке матриця інцидентності?
10. Для чого потрібен алгоритм Флойда – Уоршелла?
11. У чому суть алгоритму Флойда – Уоршелла?

Рекомендована література:

1. Асак Н. Г. Паралельні та розподілені обчислення: підруч./ Н. Г. Аксак, О. Г. Руденко, А.М. Гуржій. – Х.: Компанія СМІТ, 2009. – 480с.

2. Качко О.Г. Паралельне програмування. – Харків. нац. ун-т радіоелектроніки. – Харків : ХНУРЕ, 2016. – 403 с.

3. Згуровський М.З., Петренко А.І. Е-наука на шляху до семантичного Грід. Частина 2: Семантичний Web- і семантичний Грід // Системні дослідження та інформаційні технології. - Київ, №2, 2010. - С. 7-25.

4. Т-ефективні алгоритми наближеного розв’язання задач обчислювальної та прикладної математики /[В.К.Задірака, М.Д.Бабич, А.І.Березовський та ін.]. — Київ, 2003 р. — 261 с.

5. Бройнль Т. Паралельне програмування: Початковий курс: Навч. посібник / Вступ. Слово А. Ройтера; Пер. з нім. В.А.Святного. – К.: Вища школа, 1997.

6. – 358 С.

7. Кулаков А.Ю., Клименко І.А. Спосіб формування структури віртуальної GRID системи // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – К.: Век+, 2009. – № 50. - С. 97-100.

Додаток №1

1. Варіант

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

2. Варіант

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

3. Варіант

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

4. Варіант

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

5. Варіант

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

6. Варіант

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

7. Варіант

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

8. Варіант

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

9. Варіант

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

10. Варіант

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

11. Варіант

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

12. Варіант

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

13. Варіант

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

14. Варіант

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

15. Варіант

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

16. Варіант

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

17. Варіант

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

18. Варіант

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

19. Варіант

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

20. Варіант

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

21. Варіант

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

22. Варіант

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

23. Варіант

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

24. Варіант

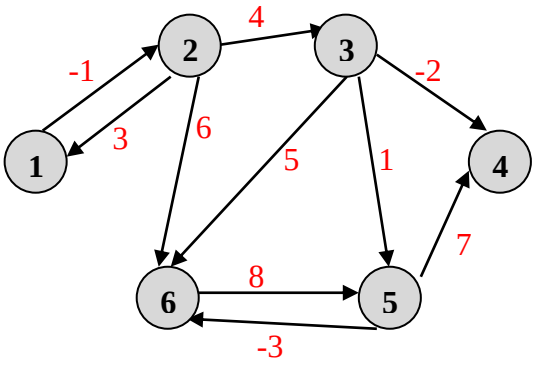
$$\begin{pmatrix} 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

25. Варіант

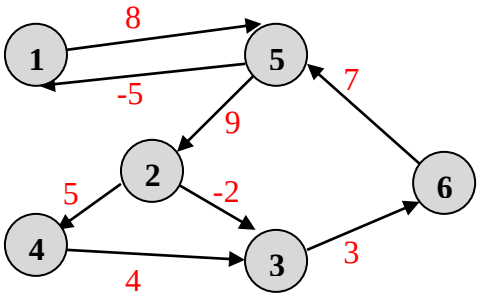
$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Додаток №2

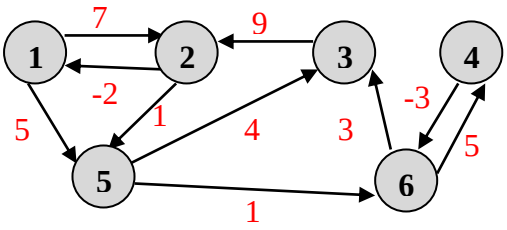
1. Варіант



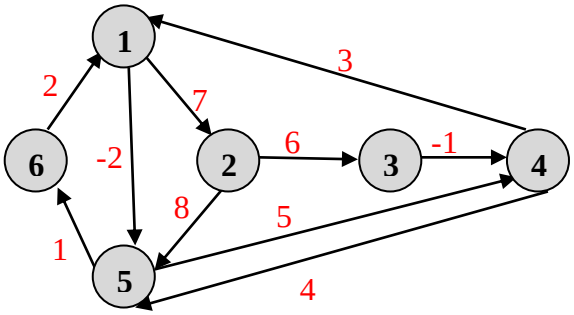
2. Варіант



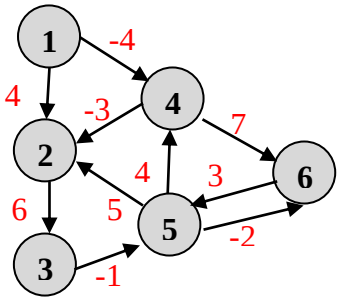
3. Варіант



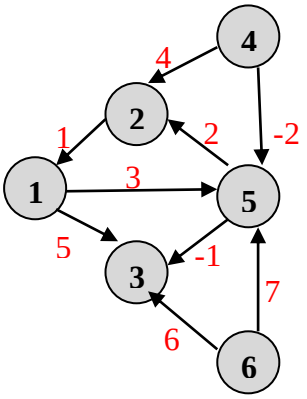
4. Варіант



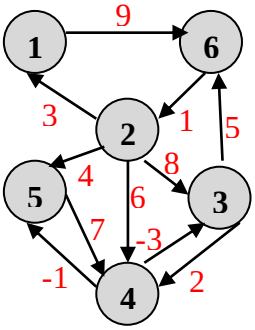
5. Варіант



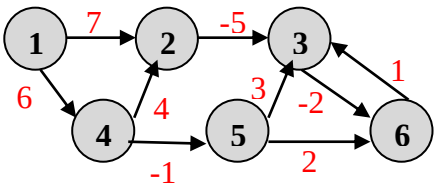
6. Варіант



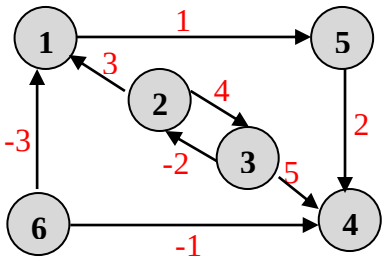
7. Варіант



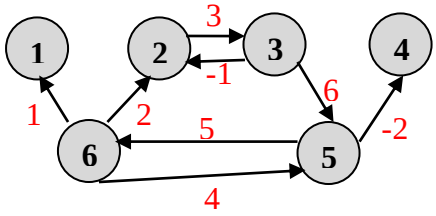
8. Варіант



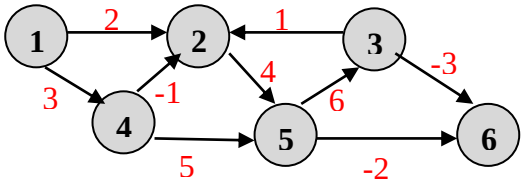
9. Варіант



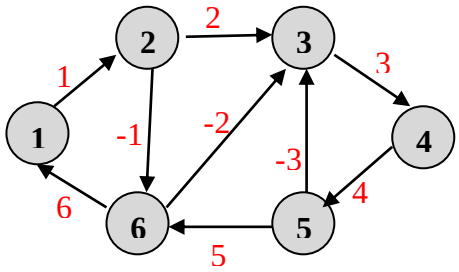
10. Варіант



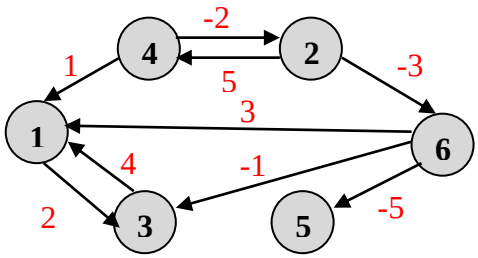
11. Варіант



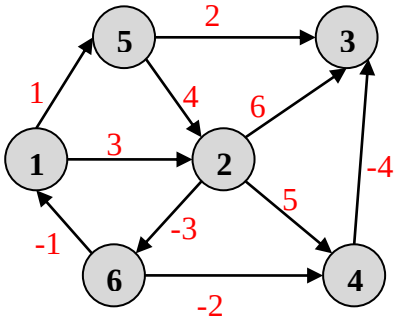
12. Варіант



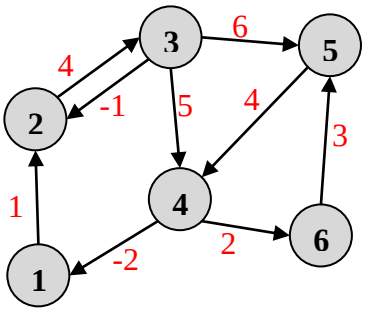
13. Варіант



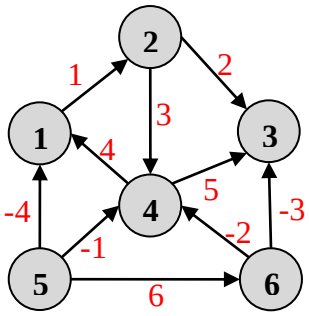
14. Варіант



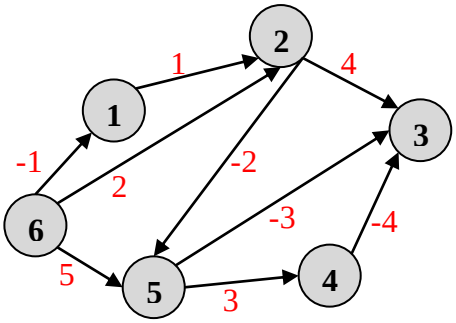
15. Варіант



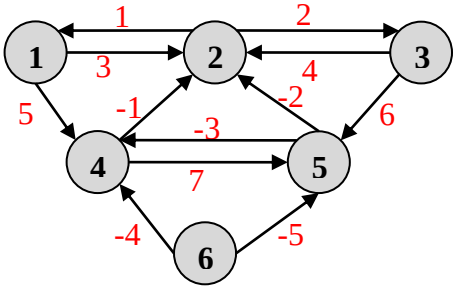
16. Варіант



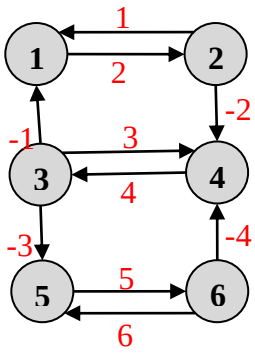
17. Варіант



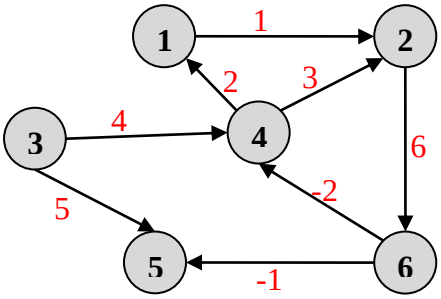
18 Варіант



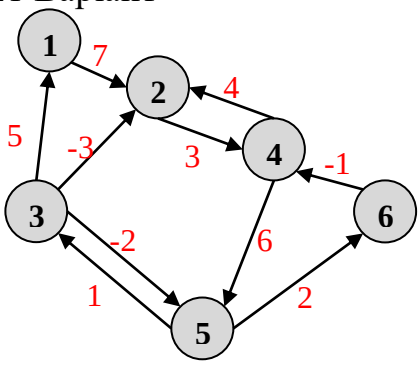
19 Варіант



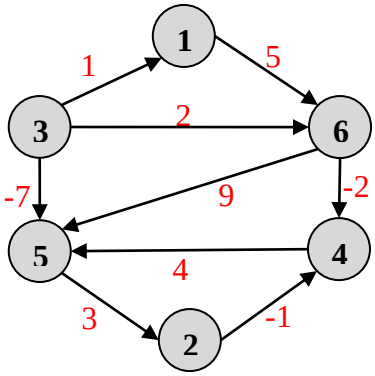
20 Варіант



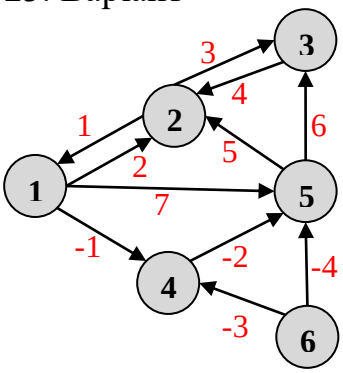
21 Варіант



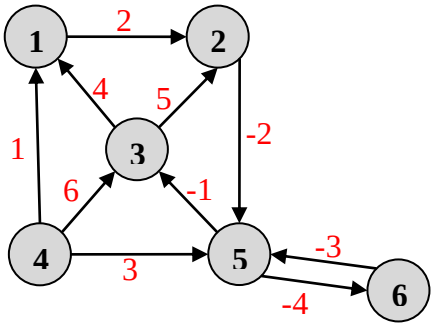
22. Варіант



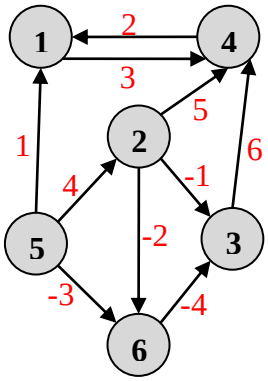
23. Варіант



24. Варіант



25. Варіант



Додаток №3

1. Варіант

$$\begin{pmatrix} 0 & \infty & \infty & \infty & \infty & \infty & \infty \\ 1 & 0 & -1 & 5 & \infty & \infty & \infty \\ 2 & 4 & 0 & 3 & \infty & -2 & 6 \\ 7 & 8 & -4 & 0 & \infty & \infty & \infty \\ \infty & 10 & \infty & 11 & 0 & \infty & 9 \\ 12 & \infty & 13 & \infty & 14 & 0 & \infty \\ -5 & 15 & \infty & -6 & \infty & 16 & 0 \end{pmatrix}$$

2. Варіант

$$\begin{pmatrix} 0 & \infty & 1 & \infty & 15 & \infty & \infty \\ \infty & 0 & -1 & 3 & \infty & 4 & \infty \\ 2 & \infty & 0 & -2 & \infty & \infty & 6 \\ 7 & 8 & 5 & 0 & \infty & \infty & \infty \\ \infty & 10 & \infty & 11 & 0 & \infty & 9 \\ 12 & \infty & 13 & \infty & 14 & 0 & \infty \\ 15 & \infty & \infty & -6 & \infty & -3 & 0 \end{pmatrix}$$

3. Варіант

$$\begin{pmatrix} 0 & 1 & 2 & 7 & \infty & 12 & \infty \\ \infty & 0 & -1 & 5 & \infty & \infty & \infty \\ \infty & \infty & 0 & 3 & \infty & 4 & 6 \\ \infty & 8 & -4 & 0 & \infty & \infty & \infty \\ \infty & 10 & \infty & 11 & 0 & \infty & 9 \\ \infty & \infty & 13 & \infty & 14 & 0 & \infty \\ -5 & 15 & \infty & -6 & \infty & 16 & 0 \end{pmatrix}$$

4. Варіант

$$\begin{pmatrix} 0 & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & 5 & \infty & \infty & \infty \\ 2 & \infty & 0 & \infty & \infty & -2 & 6 \\ 7 & 8 & \infty & 0 & \infty & \infty & \infty \\ \infty & 10 & \infty & \infty & 0 & \infty & 9 \\ 12 & \infty & 13 & \infty & \infty & 0 & \infty \\ -5 & 15 & \infty & -6 & \infty & \infty & 0 \end{pmatrix}$$

5. Варіант

$$\begin{pmatrix} 0 & \infty & 1 & \infty & 3 & \infty & 4 \\ \infty & 0 & \infty & 5 & \infty & \infty & \infty \\ 2 & \infty & 0 & \infty & \infty & -2 & 6 \\ 7 & 8 & \infty & 0 & \infty & \infty & \infty \\ \infty & 10 & \infty & 11 & 0 & \infty & 9 \\ 12 & \infty & 13 & \infty & \infty & 0 & \infty \\ -3 & 14 & \infty & -1 & \infty & \infty & 0 \end{pmatrix}$$

6. Варіант

$$\begin{pmatrix} 0 & \infty & 1 & \infty & 3 & \infty & 4 \\ 2 & 0 & \infty & 5 & \infty & \infty & \infty \\ -2 & \infty & 0 & \infty & \infty & -2 & 6 \\ \infty & 8 & \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & 7 & 0 & \infty & 9 \\ 11 & \infty & 12 & \infty & \infty & 0 & \infty \\ \infty & -4 & \infty & -1 & \infty & 10 & 0 \end{pmatrix}$$

7. Варіант

$$\begin{pmatrix} 0 & 2 & -5 & \infty & \infty & \infty & 4 \\ 1 & 0 & \infty & 5 & \infty & \infty & \infty \\ 3 & \infty & 0 & \infty & \infty & -2 & 6 \\ 7 & 8 & \infty & 0 & \infty & \infty & -4 \\ \infty & -1 & \infty & 11 & 0 & \infty & 9 \\ 10 & \infty & 12 & \infty & \infty & 0 & \infty \\ -3 & \infty & \infty & \infty & \infty & \infty & 0 \end{pmatrix}$$

8. Варіант

$$\begin{pmatrix} 0 & \infty & 1 & \infty & 3 & \infty & 4 \\ \infty & 0 & \infty & \infty & \infty & \infty & \infty \\ 2 & \infty & 0 & \infty & \infty & -2 & 6 \\ 7 & 8 & \infty & 0 & \infty & \infty & \infty \\ \infty & 10 & \infty & 11 & 0 & \infty & 9 \\ \infty & \infty & \infty & \infty & \infty & 0 & \infty \\ -3 & 14 & \infty & -1 & \infty & \infty & 0 \end{pmatrix}$$

9. Варіант

$$\begin{pmatrix} 0 & \infty & 1 & \infty & 3 & \infty & 4 \\ \infty & 0 & 2 & 5 & \infty & 7 & \infty \\ \infty & 12 & 0 & \infty & \infty & -2 & 6 \\ \infty & 8 & \infty & 0 & 13 & \infty & \infty \\ \infty & 10 & \infty & 11 & 0 & -3 & 9 \\ \infty & \infty & \infty & \infty & \infty & 0 & \infty \\ \infty & 14 & \infty & -1 & \infty & \infty & 0 \end{pmatrix}$$

10. Варіант

$$\begin{pmatrix} 0 & \infty & 1 & \infty & 3 & \infty & 4 \\ \infty & 0 & \infty & 10 & \infty & 12 & \infty \\ 2 & \infty & 0 & \infty & \infty & -2 & 6 \\ 7 & \infty & \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & 8 & 11 & 0 & \infty & 9 \\ \infty & \infty & \infty & \infty & 5 & 0 & \infty \\ -3 & 14 & \infty & -1 & \infty & \infty & 0 \end{pmatrix}$$

11. Варіант

$$\begin{pmatrix} 0 & \infty & 1 & \infty & 3 & 10 & 4 \\ 12 & 0 & -1 & 5 & \infty & \infty & \infty \\ 2 & \infty & 0 & -3 & \infty & -2 & 6 \\ 7 & 8 & \infty & 0 & \infty & \infty & \infty \\ \infty & 10 & \infty & 11 & 0 & \infty & 9 \\ \infty & \infty & \infty & -4 & \infty & 0 & \infty \\ 8 & \infty & \infty & \infty & \infty & \infty & 0 \end{pmatrix}$$

12. Варіант

$$\begin{pmatrix} 0 & \infty & 1 & \infty & 3 & 10 & 4 \\ \infty & 0 & -1 & \infty & \infty & \infty & \infty \\ 2 & 5 & 0 & -3 & 8 & -2 & 6 \\ 7 & \infty & 12 & 0 & \infty & 13 & \infty \\ \infty & 10 & \infty & 11 & 0 & \infty & 9 \\ \infty & -5 & \infty & -4 & \infty & 0 & \infty \\ 8 & \infty & \infty & \infty & \infty & \infty & 0 \end{pmatrix}$$

13. Варіант

$$\begin{pmatrix} 0 & \infty & \infty & \infty & 3 & \infty & 4 \\ \infty & 0 & -1 & \infty & \infty & \infty & \infty \\ 2 & 5 & 0 & \infty & 8 & -2 & 6 \\ 7 & \infty & 12 & 0 & \infty & 11 & \infty \\ \infty & 1 & \infty & 10 & 0 & \infty & 9 \\ \infty & \infty & \infty & -4 & \infty & 0 & \infty \\ 8 & \infty & -3 & 10 & \infty & \infty & 0 \end{pmatrix}$$

14. Варіант

$$\begin{pmatrix} 0 & \infty & \infty & \infty & 3 & \infty & 4 \\ \infty & 0 & -1 & \infty & \infty & \infty & \infty \\ 2 & 5 & 0 & \infty & 8 & -2 & 6 \\ 4 & 1 & -5 & 0 & \infty & 11 & \infty \\ \infty & \infty & \infty & \infty & 0 & \infty & 9 \\ \infty & \infty & \infty & \infty & -4 & 0 & \infty \\ 7 & \infty & -3 & 10 & \infty & \infty & 0 \end{pmatrix}$$

15. Варіант

$$\begin{pmatrix} 0 & -6 & \infty & \infty & 3 & \infty & 4 \\ \infty & 0 & \infty & \infty & \infty & \infty & \infty \\ 2 & 5 & 0 & \infty & 8 & -2 & 6 \\ 4 & 1 & -5 & 0 & \infty & 11 & 12 \\ \infty & \infty & \infty & \infty & 0 & \infty & 9 \\ \infty & \infty & \infty & \infty & \infty & 0 & \infty \\ 7 & \infty & -3 & 10 & \infty & \infty & 0 \end{pmatrix}$$

16. Варіант

$$\begin{pmatrix} 0 & \infty & \infty & \infty & 3 & \infty & 11 \\ \infty & 0 & \infty & \infty & \infty & \infty & \infty \\ 2 & \infty & 0 & \infty & 8 & -2 & 6 \\ 4 & \infty & -5 & 0 & \infty & 1 & 12 \\ \infty & 5 & \infty & \infty & 0 & \infty & 9 \\ \infty & \infty & \infty & \infty & \infty & 0 & \infty \\ 7 & \infty & -3 & 10 & \infty & \infty & 0 \end{pmatrix}$$

17. Варіант

$$\begin{pmatrix} 0 & -6 & \infty & \infty & \infty & \infty & 4 \\ \infty & 0 & \infty & \infty & \infty & \infty & \infty \\ 2 & 5 & 0 & \infty & 8 & -2 & 6 \\ 4 & \infty & -5 & 0 & \infty & 11 & 12 \\ \infty & \infty & \infty & \infty & 0 & -1 & 9 \\ \infty & \infty & 1 & \infty & 13 & 0 & \infty \\ 7 & \infty & -3 & 10 & 3 & \infty & 0 \end{pmatrix}$$

18. Варіант

$$\begin{pmatrix} 0 & -6 & \infty & -5 & \infty & 3 & 4 \\ \infty & 0 & \infty & \infty & \infty & \infty & \infty \\ 2 & 5 & 0 & \infty & \infty & -2 & 6 \\ 4 & 1 & \infty & 0 & \infty & 11 & 12 \\ 8 & \infty & \infty & \infty & 0 & \infty & 9 \\ \infty & \infty & \infty & \infty & \infty & 0 & \infty \\ 7 & \infty & \infty & 10 & \infty & \infty & 0 \end{pmatrix}$$

19. Варіант

$$\begin{pmatrix} 0 & -6 & \infty & \infty & 3 & \infty & 4 \\ 4 & 0 & 1 & -5 & 11 & \infty & 12 \\ 2 & 5 & 0 & \infty & 8 & -2 & 6 \\ \infty & \infty & \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 0 & \infty & 9 \\ \infty & \infty & \infty & \infty & \infty & 0 & \infty \\ 7 & \infty & -3 & 10 & \infty & \infty & 0 \end{pmatrix}$$

20. Варіант

$$\begin{pmatrix} 0 & -6 & \infty & \infty & 3 & -1 & \infty \\ 4 & 0 & 1 & -5 & 11 & \infty & 12 \\ 2 & 5 & 0 & \infty & \infty & -2 & 6 \\ \infty & \infty & \infty & 0 & \infty & \infty & \infty \\ 6 & \infty & \infty & 7 & 0 & \infty & 9 \\ \infty & \infty & \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & -3 & 10 & 8 & \infty & 0 \end{pmatrix}$$

21. Варіант

$$\begin{pmatrix} 0 & \infty & \infty & \infty & 3 & -1 & \infty \\ \infty & 0 & 1 & -5 & 11 & \infty & 12 \\ 2 & \infty & 0 & \infty & \infty & -2 & 6 \\ \infty & \infty & \infty & 0 & 12 & \infty & \infty \\ 6 & \infty & 4 & 7 & 0 & \infty & 9 \\ \infty & \infty & \infty & 5 & -6 & 0 & \infty \\ \infty & \infty & -3 & 10 & 8 & \infty & 0 \end{pmatrix}$$

22. Варіант

$$\begin{pmatrix} 0 & 11 & -4 & \infty & \infty & -1 & \infty \\ \infty & 0 & 1 & -5 & \infty & \infty & 12 \\ 2 & \infty & 0 & \infty & \infty & -2 & 6 \\ \infty & 3 & \infty & 0 & \infty & \infty & \infty \\ 6 & \infty & 4 & 7 & 0 & \infty & 9 \\ \infty & 8 & \infty & 5 & \infty & 0 & \infty \\ \infty & \infty & -3 & 10 & \infty & \infty & 0 \end{pmatrix}$$

23. Варіант

$$\begin{pmatrix} 0 & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & 1 & -5 & \infty & \infty & 12 \\ 2 & \infty & 0 & \infty & \infty & \infty & 6 \\ \infty & \infty & \infty & 0 & \infty & \infty & \infty \\ 6 & \infty & 4 & 7 & 0 & \infty & 9 \\ \infty & \infty & \infty & 5 & \infty & 0 & \infty \\ \infty & \infty & -3 & 10 & \infty & \infty & 0 \end{pmatrix}$$

24. Варіант

$$\begin{pmatrix} 0 & \infty & \infty & \infty & -6 & \infty & -2 \\ \infty & 0 & 1 & -5 & \infty & 11 & 12 \\ 2 & \infty & 0 & \infty & 8 & \infty & 6 \\ \infty & 3 & \infty & 0 & -1 & \infty & \infty \\ 6 & \infty & 4 & 7 & 0 & -4 & 9 \\ \infty & \infty & \infty & 5 & \infty & 0 & \infty \\ \infty & \infty & -3 & 10 & \infty & \infty & 0 \end{pmatrix}$$

25. Варіант

$$\begin{pmatrix} 0 & \infty & 4 & \infty & -6 & \infty & -2 \\ \infty & 0 & 1 & -5 & \infty & 11 & 12 \\ 2 & \infty & 0 & \infty & 8 & \infty & 6 \\ \infty & 3 & 6 & 0 & -1 & 7 & \infty \\ \infty & \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & 5 & 9 & 0 & \infty \\ \infty & \infty & -3 & 10 & \infty & 8 & 0 \end{pmatrix}$$

Тема 3. Технологія паралельного програмування MPI

Лабораторна робота №2. Налаштування середовища для MPI-програмування

Мета: навчити студентів налаштовувати середовище для MPI-програмування.

Основні поняття: MPI, MPICH, Wmpiexec, Wmpiregister, Wmpconfig, спільний ресурс.

План

1. Поняття MPI.
2. Принципи роботи MPICH.
3. Встановлення MPICH у Windows.
4. Налаштування MPICH.
5. Запуск MPI-програм.

І. Теоретичні відомості

MPI(Message Passing Interface) – інтерфейс обміну повідомленнями (інформацією) між одночасно працюючими обчислювальним процесами. Він широко використовується для створення паралельних програм для обчислювальних систем з розподіленою пам'яттю (кластерів).

MPICH – сама відома реалізація MPI, створена в Арагонській національній лабораторії (США). Існують версії цієї бібліотеки для всіх популярних операційних систем. І до того ж, вона безкоштовна.

MPICH2 – це швидка і широко використовувана реалізація MPI. Двійка у назві – це не версія програмного забезпечення, а номер стандарту MPI, який реалізований у бібліотеці. MPICH2 відповідає стандарту MPI 2.0, звідси і назва.

Принцип роботи MPICH

MPICH для Windows складається з наступних компонентів:

- Менеджер процесів smpd.exe, який являє собою системну службу. Менеджер процесів веде список обчислювальних вузлів системи, і запускає на цих вузлах MPI-програми та надає їм необхідну інформацію для роботи та обміну повідомленнями.
- Заголовочні файли (h.) і бібліотеки стадії компіляції (lib.), необхідні для розробки MPI-програм.
- Бібліотеки часу виконання (.dll), необхідні для роботи MPI-програм.
- Додаткові утиліти (.exe), необхідні для налаштування MPICH та запуску MPI-програм.

Усі компоненти, крім бібліотек часу виконання, за замовченням встановлюються у папку C:\Program Files\MPICH2; dll-бібліотеки встановлюються C:\Windows\System32.

Менеджер процесів є основним компонентом, який повинен бути встановлений та налаштований на всіх комп'ютерах мережі (бібліотеки часу виконання можна, у крайньому випадку, копіювати разом з MPI-програмою).

Інші файли потрібна для розробки MPI-програм і налаштування деякого «головного» комп'ютера, з якого буде проходити їх запуск(рис. 3.1).

Менеджер працює у фоновому режимі і чекає запитів до нього з мережі з боку «головного» менеджера процесів (за замовчуванням використовується мережевий порт 8676). Щоб якось захистити себе від хакерів та вірусів, менеджер вимагає пароль при звертанні до нього. Коли один менеджер процесів звертається до іншого, він передає йому свій пароль. Звідси слідує, що потрібно вказувати один і той самий пароль при встановленні MPICH на комп'ютери мережі.

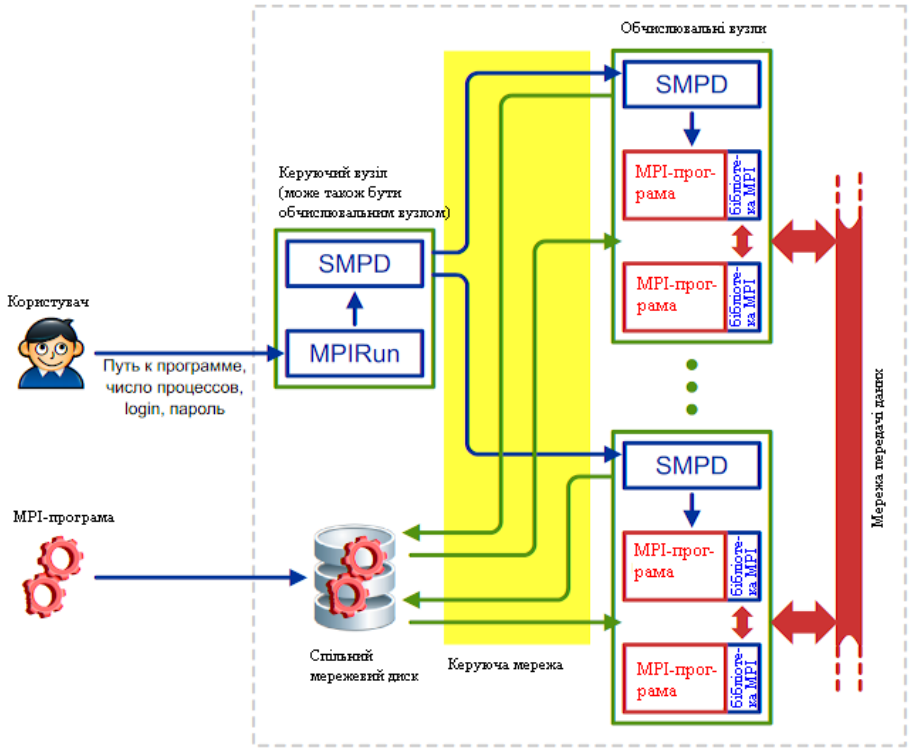


Рис. 3.1. Схема роботи MPICH на кластері.

Запуск MPI-програми проводиться наступним чином:

1. Користувач з допомогою програми Mpirun (чи Mpiexec, при використанні MPICH2 для Windows) вказує ім'я виконуваного файлу MPI-програми і потрібну кількість процесів. Крім того можна вказати імя користувача і пароль: процеси MPI-програми будуть запускатися від імені цього користувача.
2. Mpirun передає відомості про запуск локальному менеджеру процесів, у якого є список доступних обчислювальних вузлів.
3. Менеджер процесів звертається до обчислювальних вузлів за списком та передає запущеним на них менеджерам процесів вказівки по запуску MPI-програми.
4. Менеджери процесів запускають на обчислювальних вузлах декілька копій MPI-програми (можливо, по декілька копій на кожному вузлі) та передають програмам необхідну інформацію для зв'язку один з одним.

Дуже важливим моментом тут є те, що перед запуском MPI-програма не копіюється автоматично на обчислювальні вузли кластера. Замість цього менеджер процесів передає вузлам шлях до файлу програми, що виконується, точно у тому вигляді, у якому користувач вказав цей шлях програмі Mpirun. Це означає, що якщо ви, наприклад, запускаєте програму C:\proga.exe, то всі менеджери процесів на обчислювальних вузлах будуть намагатися запустити файл C:\proga.exe. Якщо хоча б на одному з вузлів такого не буде, то станеться помилка MPI-програми.

Щоб кожен раз не копіювати вручну програму і всі необхідні для її роботи файли на обчислювальні кластери, зазвичай використовують *спільний мережевий ресурс*. У цьому випадку користувач копіює програму і додаткові файли на мережевий ресурс, який видно всім вузлам кластера, і вказує шлях до файлу програми на цьому ресурсі. Додатковою зручністю такого підхода є те, що при наявності можливості запису на спільний мережевий ресурс запущені копії програми можуть записувати туди результати своєї роботи.

Робота MPI-програми проходить наступним чином:

1. Програма запускається та ініціалізує бібліотеку часу виконання MPICH шляхом виклику функції MPI_Init.
2. Бібліотека отримує від менеджера процесів інформацію про кількість і місце знаходження процесів програми, та встановлює з ними зв’язок.
3. Після цього запущені копії програми обмінюються одна з іншою інформацією за допомогою бібліотеки MPICH. З точки зору операційної системи бібліотека є частиною програми, тому можна вважати, що запущені копії MPI-програми обмінюються даними напряму одна з одною, як будь які інші додатки, що передають дані по мережі.
4. Консоль вводу-виводу всіх процесів MPI-програм перенаправляється на консоль, на якій запущена Mpirun.
5. Перед завершенням усі процеси визивають функцію MPI_Finalize, яка коректно завершує передачу і прийом усіх повідомлень, і відключає MPICH.

Встановлення MPICH у Windows

Спочатку потрібно скачати останню версію MPICH2 з http://www.mcs.anl.gov/research/projects/mpich2/. Обираємо операційну систему, потім переходимо за посиланням і скачуємо msi -файл, наприклад mpich2-1.2.1-win-ia32.

Завантажений дистрибутив необхідно запустити з правами адміністратора на всіх комп’ютерах, на яких планується запуск MPI-програми.

Під час встановлення потрібно буде ввести пароль для доступу до менеджера процесів SMPD. За замовчуванням це: behappy. Потрібно ввести однаковий пароль на всіх комп’ютерах(рис. 3.2).

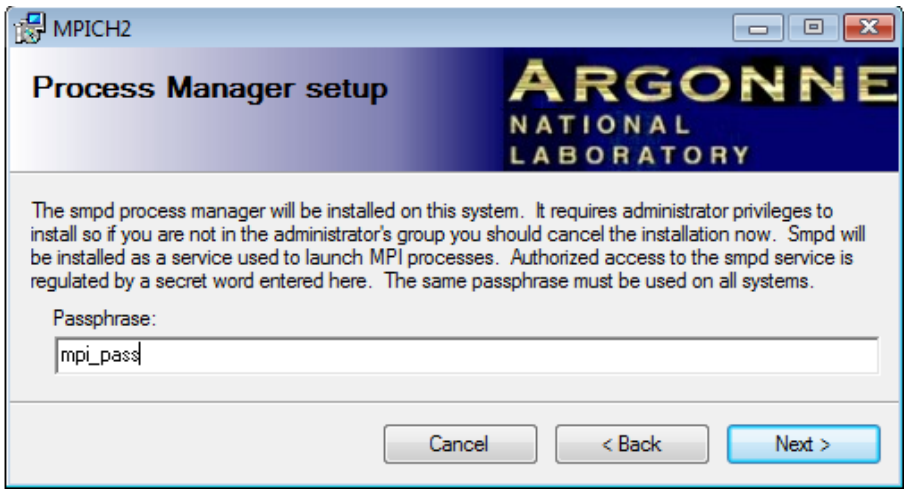


Рис. 3.2. Вказівка пароля для доступу до менеджера процесів

У вікно зазначення шляху встановлення рекомендується залишити каталог за замовченням. Крім того потрібно поставити точку у пункті “Everyone”(рис. 3.3).

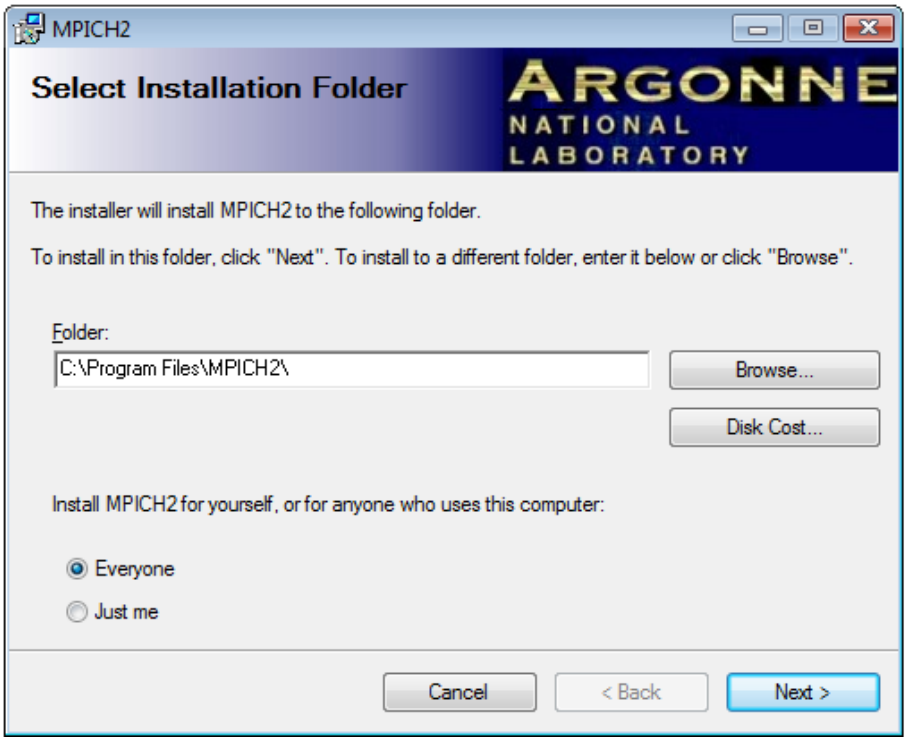


Рис. 3.3. Вказівка шляху встановлення

Якщо Windows запитає, чи дозволити доступ у мережу програмі smd.exe, то потрібно натиснути «Дозволити».

Перед тим як переходити до налаштування MPICH, потрібно обов’язково перевірити дві речі: чи запущена служба ”MPICH2 Process Manager” і чи дозволений цій службі доступ до мережі.

Натисніть Пуск → Настройка → Панель управления → Администрирование → Службы. Потрібно побачити ”MPICH2 Process Manager” у списку служб (рис. 3.4). Ця служба повинна працювати. Якщо

служба у списку відсутня, то, мабуть, інсталятор був запущений не від імені Адміністратора.

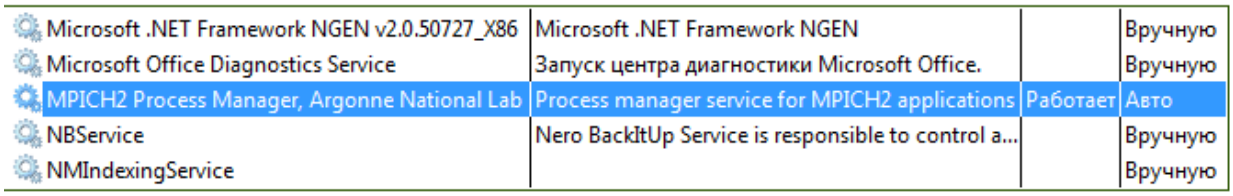


Рис. 3.4. Служба "MPICH2 Process Manager" у списку служб

Далі потрібно перевірити чи є доступ у мережі для MPICH. Для цього потрібно зайти в Пуск → Настройка → Панель управления →Брадмауэр Windows. Там натисніть «Разрешение запуска программы через брадмауэр Windows». Потрібно побачити у списку дозволених програм "Process launcher for MPICH2 applications" і "Process manager service for MPICH2 applications"(рис. 3.5).

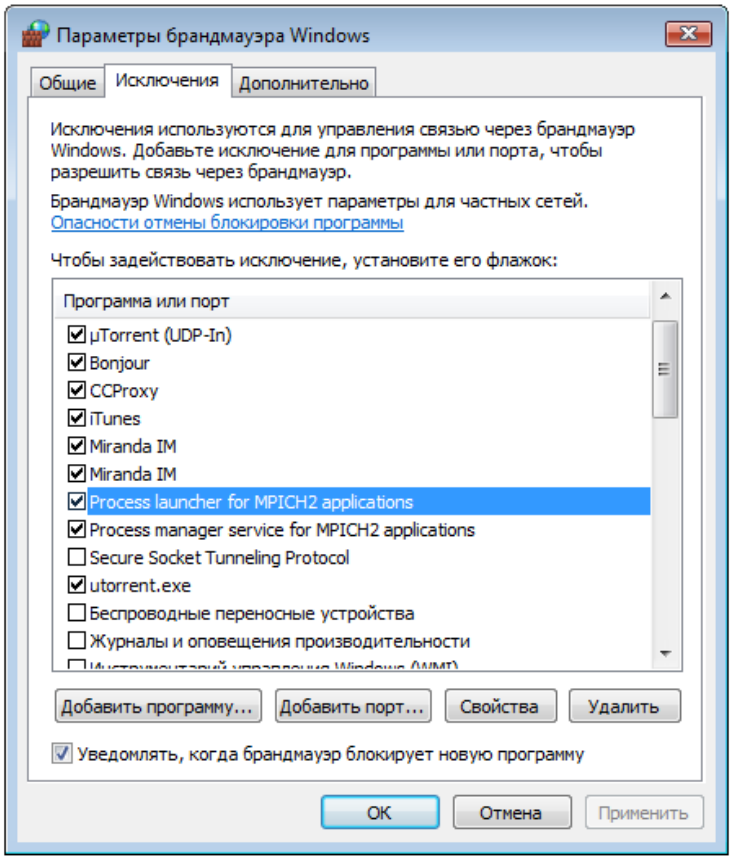


Рис. 3.5. Програми MPICH у списку виключень брадмауера

Якщо якась з перерахованих програм відсутня у списку дозволених программ, то ми можемо добавити її вручну. Для цього потрібно натиснути на кнопку «Добавить программу…», і добавити C:\Program files\mpich2\bin\mpiexec.exe, якщо відсутня "Process launcher for MPICH2 applications", та C:\Program files\mpich2\bin\smpd.exe, якщо відсутня "Process manager service for MPICH2 applications".

Налаштування MPICH

Розглянемо налаштування MPICH на прикладі конфігурації з двох комп'ютерів, об'єднаних у локальну мережу (Wi-Fi): один комп'ютер має мережеве ім'я MrBig і IP-адресу 192.168.1.4, інший - ім'я Small та адресу 192.168.1.3. Припустимо, що MPI-програми ми хочемо запускати з комп'ютера MrBIG. На обох комп'ютерах встановлені російськомовні версії Windows. На MrBIG встановлена Windows Vista, на Small - Windows XP. Кожен комп'ютер має двоядерний процесор.

Перш за все потрібно створити на всіх комп'ютерах користувача з однаковим ім'ям і паролем; від імені цього користувача будуть запускатися MPI-програми (якщо у нас один комп'ютер, то цей крок можна пропустити). Найпростіше це зробити, встановивши однаковий пароль користувачам Адміністратор.

У коментарях підказують, що краще створити для MPI-програм обліковий запис з обмеженими правами, а не віддавати адміністраторську обліковий запис

Як вже було сказано раніше, будь-яку дію система MPICH виконує від вказаного імені користувача. Для того, щоб запитувати ім'я користувача і пароль, використовується програма Wmpiregister(рис. 3.6). Проблема в тому, що ім'я користувача та пароль запитуються досить часто, що може викликати роздратування. Для того, щоб цього уникнути, Wmpiregister може зберігати ім'я користувача і пароль у реєстрі Windows.

Запустіть Wmpiregister на тому комп'ютері, з якого ви збираєтеся запускати MPI-програми. Для цього натисніть Пуск → Програми → MPICH2 → wmpiregister.exe. Вікно програми виглядає наступним чином:

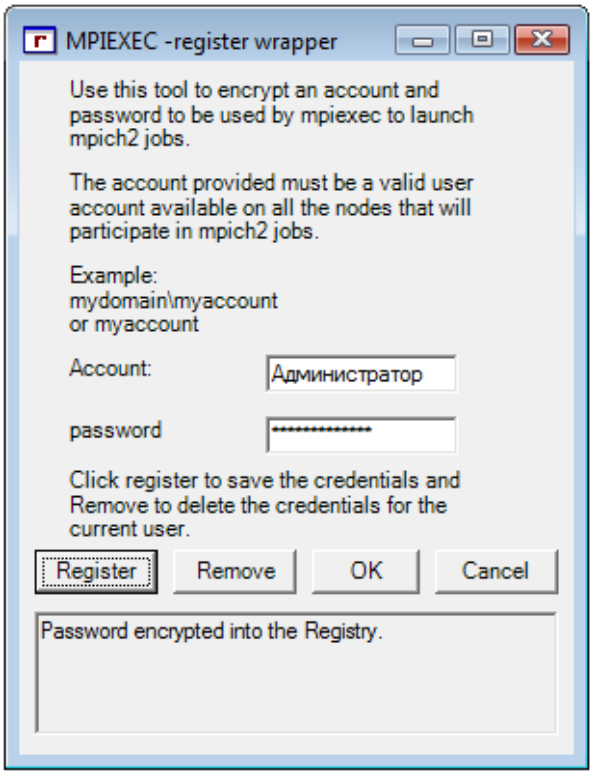


Рис. 3.6. Програма Wmpiregister

Призначення кнопок (зправа-наліво):

- «Cancel» - закрити програму без виконання якої-небудь дії.
- «OK» - передати введені ім'я користувача та пароль,програмі, що викликала. Якщо Wmpiregister запущена нами як окремий додаток, то натискання кнопки OK еквівалентно натисканню кнопки Cancel.
- «Remove» - натискання цієї кнопки видаляє збережені раніше ім'я користувача і пароль з реєстру Windows.
- «Register» - зберігає ім'я користувача та пароль у реєстрі.

Введіть ім'я користувача і пароль у вікні програми і натисніть кнопку «Register». Повинна з'явитися напис «Password encrypted into the Registry». Після цього вікно програми більше не буде з'являтися при виконанні будь-яких дій MPICH. Якщо ви захочете згодом видалити ім'я користувача і пароль з реєстру, то вам потрібно буде знову запустити цю програму, і натиснути кнопку «Remove».

Тепер нам потрібно налаштувати менеджери процесів MPICH. Для цього запустіть на всіх комп'ютерах програму Wmpiconfig. Якщо всі попередні кроки зроблені правильно, то в полі «version» у лівій колонці таблиці ви повинні побачити версію встановленого менеджера процесів (рис. 3.7).

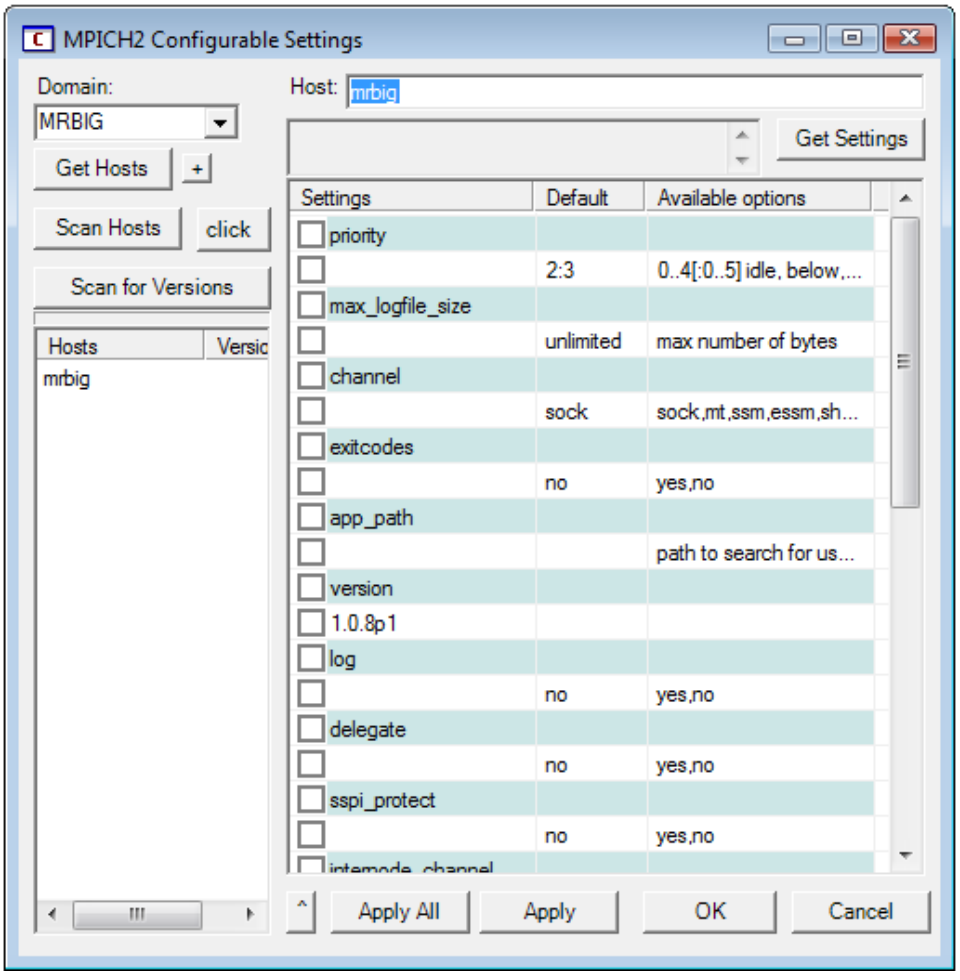


Рис. 3.7. Програма Wmpconfig

Якщо менеджер процесів не встановлений, або йому закритий доступ в мережу, то ви побачите напис «MPICH2 not installed or unable to query the host» в одному з полів лівого стовпця. У цьому випадку потрібно перевірити чи правильно встановлена MPICH (рис. 3.8).

Wmpiconfig призначена для налаштування менеджерів процесів на поточному комп'ютері та інших комп'ютерах мережі. Для цього вона під'єднується до менеджерів процесів на обраних комп'ютерах, читає наявні у них настройки, і повідомляє їм нові налаштування, якщо потрібно.

Елементи керування програми Wmpiconfig виконують наступні дії:

- Зліва-внизу є список комп'ютерів, з якими працює програма налаштування. Ім'я комп'ютера на білому фоні означає, що не було спроб зв'язатися з цим комп'ютером; зелений фон означає, що зв'язок проведений успішно; сірий фон означає, що при встановленні зв'язку виникла помилка. Видалити комп'ютер зі списку можна клавішею Del. Слід мати на увазі, що цей список призначений тільки для зручності настройки, і не має ніякого відношення до списку комп'ютерів, на яких буде запущена MPI-програма.
- Кнопка «Get Hosts» отримує список комп'ютерів в заданому домені або робочій групі (у випадаючому списку «Domain»). Отриманий список замінює наявний список комп'ютерів або, якщо натиснута кнопка «+», додає комп'ютери до поточного списку.
- Кнопка «Scan Hosts» отримує налаштування з усіх комп'ютерів списку; кнопка «Scan for Versions» отримує тільки номери версій.
- Кнопка «Get Settings» отримує поточні налаштування того комп'ютера, ім'я якого введено в полі введення «Host». При виборі комп'ютера у списку комп'ютерів його ім'я автоматично вводиться в поле «Host». Якщо натиснута кнопка «Click», то настройки будуть отримані автоматично при виборі комп'ютера зі списку.
- Праворуч у вікні розташована таблиця налаштувань. Якщо ви хочете змінити будь-які настройки, то потрібно двічі клацнути на відповідному полі у першому стовпці таблиці. Порожнє поле означає, що використовується настройка за замовчуванням, зазначена у другому стовпці. Налаштування, призначені до зміни, слід відзначати установкою галочки зліва.
- Кнопка «Apply» застосовує виділені галочкою налаштування до того комп'ютера, ім'я якого знаходиться в полі «Host». Кнопка «Apply All» застосовує настройки до всіх комп'ютерів списку.
- Кнопка «Cancel» закриває програму.

На тому комп'ютері, з якого планується запуск програм, потрібно вказати список доступних обчислювальних вузлів. Цей список потрібно ввести (через пробіл) в полі hosts лівого стовпця таблиці (рис. 3.10), і натиснути кнопку «Apply». На малюнку показаний приклад, коли сам комп'ютер, з якого здійснюється запуск MPI-програм, є одним з обчислювальних вузлів.

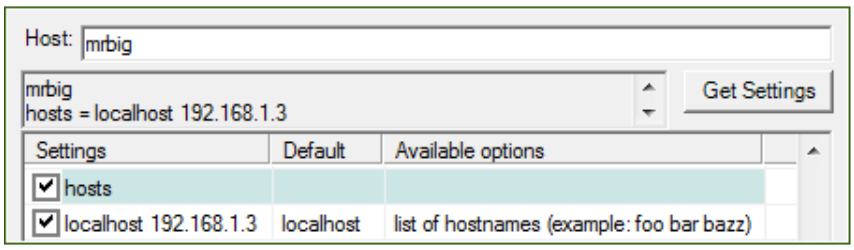


Рис. 3.8. Вказуємо список доступних обчислювальних вузлів

Тепер перевіримо, чи бачать менеджери процесів один одного по мережі. Для цього в програмі Wmpiconfig на «головному» комп'ютері потрібно ввести в поле «Host» адресу комп'ютера, що перевіряється і натиснути «Get Settings». Ми повинні побачити версію встановленого менеджера процесів на обраному комп'ютері. Якщо зв'язок встановити не вдається - буде пауза в кілька секунд, після чого в останньому рядку таблиці з'явиться повідомлення про помилку. Якщо до цих пір у нас все йшло добре, то слід перевірити мережу: переконатися, що комп'ютери "бачать" один одного, спробувати відключити бранмауери, тощо. Також перевірте, чи співпадає контрольна фраза (поле phrase) на всіх комп'ютерах.

Запуск MPI-програм

Для запуску MPI-програм у комплект MPICH2 входить програма з графічним інтерфейсом Wmpiexec(рис. 3.9), яка являє собою оболонку навколо відповідної утиліти командного рядка Mpiexec.

Вікно програми Wmpiexec показано на малюнку 9 (зверніть увагу, що включений прапорець «more options»).

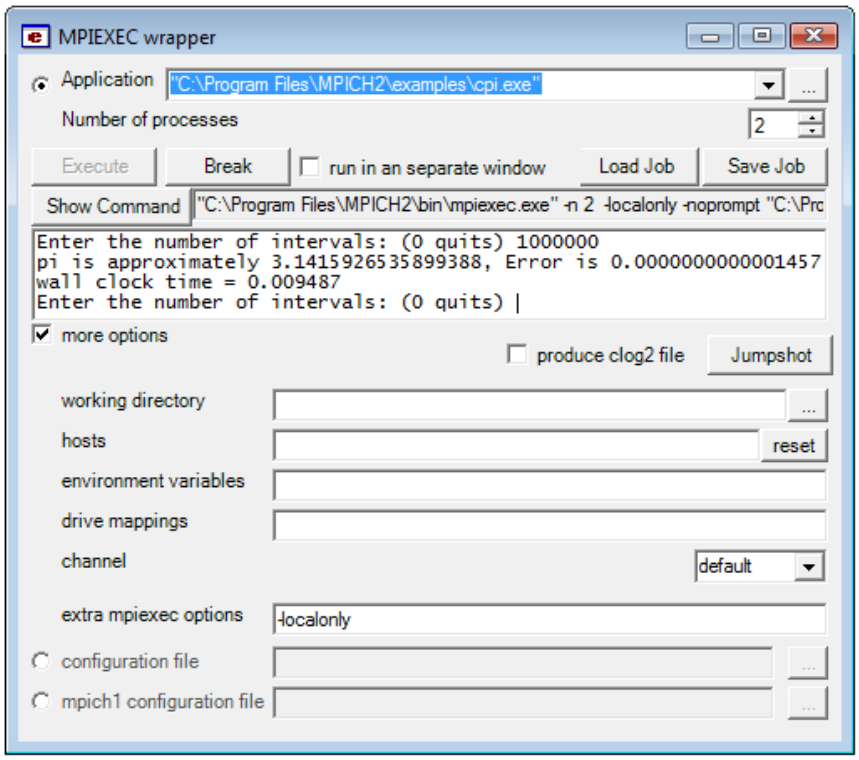


Рис. 3.9. Програма Wmpiexec

Елементи керування вікна мають наступні значення:

- Поле введення «Application»: сюди вводиться шлях до MPI-програми. Як вже було сказано раніше, шлях передається в незмінному вигляді на всі комп'ютери мережі, тому бажано, щоб програма розташовувалася у загальній мережевий папці.
- «Number of processes»: число процесів, що запускаються. За замовчуванням процеси розподіляються порівну між комп'ютерами мережі, однак це можна змінити за допомогою конфігураційного файлу.
- Кнопка «Execute» запускає програму; кнопка «Break» примусово завершує всі запущені екземпляри.
- Прапорець «run in a separate window» перенаправляє вивід усіх примірників MPI-програми в окреме консольне вікно.
- Кнопка «Show Command» показує у полі праворуч командний рядок, яка використовується для запуску MPI-програми (Wmpiexec - всього лише оболонка над Mpiexec). Командний рядок збирається з усіх налаштувань, введених в інших полях вікна.
- Далі йде велике поле, в яке потрапляє введення-виведення всіх примірників MPI-програми, якщо не встановлено прапорець «run in a separate window».
- Прапорець «more options» показує додаткові параметри.
- «Working directory»: сюди можна ввести робочий каталог програми. Знову ж таки, цей шлях повинен бути вірний на всіх обчислювальних вузлах. Якщо шлях не зазначений, то в якості робочого каталогу буде використовуватися місце знаходження MPI-програми.
- «Hosts»: тут можна вказати через пропуск список обчислювальних вузлів, що використовуються для запуску MPI-програми. Якщо це поле порожнє, то використовується список, що зберігається в налаштуваннях менеджера процесів поточного вузла.
- «Environment variables»: у цьому полі можна вказати значення додаткових змінних оточення, що встановлюються на всіх вузлах на час запуску MPI-програми. Синтаксис наступний: імя1 = значення1, імя2 = значення2.
- «Drive mappings»: тут можна вказати мережевий диск, що підключається на кожному обчислювальному вузлі на час роботи MPI-програми. Синтаксис: Z:\\winsrv\wdir.
- «Channel»: дозволяє вибрати спосіб передачі даних між екземплярами MPI-програми.
- «Extra mpiexec options»: в це поле можна ввести додаткові ключі для командного рядка Mpiexec.

II. Практична частина

Інструкції до виконання

1. Опрацювати теоретичні відомості.

2. Скачати дистрибутив з папки обміну.

3. Встановити та налаштувати MPICH2 на своєму комп’ютері.

4. Разом з MPICH поставляється зразок MPI-програми: C:\Program Files\MPICH2\examples\cpi.exe. Це проста програма, наближено обчислює значення числа Пі шляхом чисельного обчислення наступного інтегралу:

$$\pi = \int_0^1 \frac{4}{1+x^2}\,dx.$$

.

По-черзі запустіть один, два, і чотири процеси на одному комп'ютері. Для цього у полі «Application» уведіть шлях до програми cpi.exe. Так як програма запускається на одному комп'ютері, шлях можна вводити локальний.

«Number of processes» введіть рівним 1, і натисніть кнопку «Execute». Програма запуститься і запитає число інтервалів для чисельного інтегрування. Введіть 100000000, і натисніть Ctrl + Enter. Програма порахує число Пі, і виведе час, витрачений на обчислення. Проведіть обчислення зі 100 мільйонами інтервалів ще кілька разів, щоб визначити мінімальний час. Після цього введіть 0, і натисніть Ctrl + Enter. Програма завершиться.

Далі запустіть програму ще раз, на цей раз вибравши 2 процеси (потім 4). Якщо не вийде - перезапустіть Wmpiexec.

5. Зробіть висновок про час та похибку виконання програми у кожному з випадків.

6. Показати результати викладачу.

7. Розбитися на пари. Вибрати один з двох комп’ютерів за «головний». На ньому створити спільну мережеву папку і туди скинути cpi.exe. (згідно прикладу це папка за адресою \\192.168.1.4\MPI\). Переконайтеся, що програму видно з усіх комп'ютерів за однією і тією ж мережевою адресою.

8. Так як екземпляри MPI-програми працюють та обмінюються даними по мережі незалежно від менеджера процесів, їх теж потрібно внести в список виключень брандмауера Windows. Додайте cpi.exe (прямо з мережевого ресурсу) в список виключень брандмауера на всіх комп'ютерах мережі.

9. На «головному» комп’ютері у програмі Wmpconfig вказати список доступних обчислювальних вузлів (інший з двох).

10. Вкажіть мережеву адресу програми cpi.exe (у прикладі \\192.168.1.4\MPI\cpi.exe)

11. Запустіть програму cpi.exe на «головному» комп’ютері (по-черзі встановивши один, два, і чотири процеси).

12. Зробити висновки про час та похибку виконання програми у кожному з випадків.

13. Показати результати викладачу.

14.Видалити ім’я користувача та пароль з реєстру.

Поточні контрольні питання:

1. Що таке MPI?
2. Які основні принципи роботи MPICH?
3. Як встановлення MPICH у Windows?
4. Які основні принципи налаштування?
5. Як запустити MPI-програму?

Рекомендована література:

1. Антонов А.С. Параллельное программирование с использованием технологии MPI.

2. Аксак Н. Г. Паралельні та розподілені обчислення: підруч. / Н. Г. Аксак, О. Г. Руденко, А.М. Гуржій. – Х.: Компанія СМІТ, 2009. – 480 с.

3. Паралельні та розподілені обчислення: навчальний посібник для вищих закладів освіти / К.Т. Кузьма, О.В. Мельник. – Миколаїв: ФОП Швець В.М., 2020. – 172 с.

4. Семеренко В. П. Теорія циклічних кодів на основі автоматних моделей : монографія / Семеренко В. П. – Вінниця : ВНТУ, 2015. – 444 с.

5. Шеховцов В. А. Операційні системи / Шеховцов В. А. – Київ : Видавнича група BHV, 2005. – 576 с.

Лабораторна робота №3. Знайомство з MPI-програмуванням

Мета: ознайомитися з MPI-програмуванням.

Основні поняття: будова MPI-програми, повідомлення, складові повідомлення, види обмінів повідомлення.

План

1. Загальна будова MPI-програм.
2. Повідомлення.
3. Визначення кількості та рангу процесів.
4. Попарна передача даних.
5. Попарне приймання повідомлень.

I. Теоретичні відомості

Загальна будова MPI-програм

До першого виклику будь-якої MPI-функції необхідно викликати **функцію MPI_Init**. Її прототип:

int MPI_Init(int *argc, char **argv[]);

де argc, argv – вказівники на кількість аргументів програми і на вектор аргументів відповідно (це адреси аргументів функції main програми).

Після закінчення роботи з MPI-функціями необхідно визвати **функцію MPI_Finalize.** Її прототип:

int MPI_Finalize(void);

Загальна схема MPI-програми на мові С виглядає так:

```
#include "mpi.h"
/*Інші описи*/
int main( int argc, char** argv[])
{
/*Опис локальних змінних*/
      MPI_Init( &argc, &argv );
      /*Тіло програми*/
      MPI_Finalize();
      return 0;
}
```

Приклад №1 «Hello World»

```
#include <stdio.h>
#include "mpi.h"
int main( int argc, char** argv[])
{
      MPI_Init( &argc, &argv );
      printf("Hello World\n");
      MPI_Finalize();
      return 0;
}
```

Визначення кількості та рангу процесів

Визначення кількості процесів у паралельній програмі, що виконується, здійснюється за допомогою функцій:

int MPI_Comm_size (MPI_Comm comm, int * size).

Для визначення рангу процесу використовується функція:

int MPI_Comm_rank (MPI_Comm comm, int * rank).

Як правило, виклик функцій MPI_Comm_size і MPI_Comm_rank виконується відразу після MPI_Init:

```
# include "mpi.h"
int main (int argc, char * argv []) {
int ProcNum, ProcRank;
   <програмний код без використання MPI функцій>
   MPI_Init (& agrc, & argv);
   MPI_Comm_size (MPI_COMM_WORLD, & ProcNum);
   MPI_Comm_rank (MPI_COMM_WORLD, & ProcRank);
     <програмний код з використанням MPI функцій>
   MPI_Finalize ();
   <програмний код без використання MPI функцій>
   return 0;
}
```

Приклад №2 «Hello World2»

```
#include <stdio.h>
#include "mpi.h"
int main( int argc, char** argv )
{    int rank, size;
     double a;
    MPI_Init( &argc, &argv ); //ініціалізація паралельної частини
рпограми
    MPI_Comm_size( MPI_COMM_WORLD/*стандартне системне імя для групи
віртуальних компютерів*/, &size );//кількість процесів у групі,
    MPI_Comm_rank( MPI_COMM_WORLD, &rank ); //номер процеса в групі
     if (rank==0)
     {printf( "Hello from process %d of %d\n",  rank, size );
          }
    MPI_Finalize();//Завершення паралельної частини програми
    return 0;
}
```

Передача повідомлень

Для передачі повідомлення процес-відправник повинен виконати функцію:

int MPI_Send (void * buf, int count, MPI_Datatype type, int dest, int tag, MPI_Comm comm), де

- Buf - адреса буфера пам'яті, в якому розташовуються дані повідомлення, що відправляється,
- Count - кількість елементів даних в повідомленні,
- Type - тип елементів даних, що пересилаються повідомленям,
- Dest - ранг процесу, якому відправляється повідомлення,
- Tag - значення-тег, що використовуєть для ідентифікації повідомлень,
- Comm - комунікатор, в рамках якого виконується передача даних.

Приймання повідомлень

Для прийому повідомлення процес-одержувач повинен виконати функцію:

int MPI_Recv (void * buf, int count, MPI_Datatype type, int source, int tag, MPI_Comm comm, MPI_Status * status), де

- Buf, count, type - буфер пам'яті для прийому повідомлення, призначення кожного окремого параметра відповідає опису в MPI_Send,
- Source - ранг процесу, від якого повинен бути виконаний прийом повідомлення,
- Tag - тег повідомлення, яке має бути прийняте для процесу,
- Comm - комунікатор, в рамках якого виконується передача даних,
- Status - покажчик на структури даних з інформацією про результат виконання операції прийому даних.

Функція MPI_Get_count (MPI_Status * status, MPI_Datatype type, int * count) повертає у змінній count кількість елементів типу type в прийнятому повідомленні.

Приклад №3 «Hello World3»

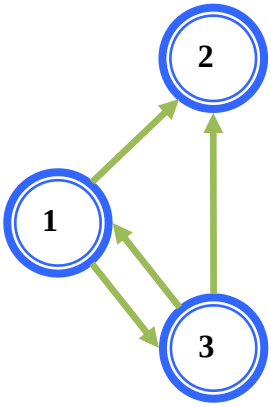
```
#include <stdio.h>
#include <string.h>
#include "mpi.h"

#define BUF_LEN 256 /*довжина буфера обміну*/
#define MSG_TAG 100 /*тег повідомлення*/

int main(int argc, char *argv[])
{
	int my_rank;
	int numprocs;
	int source;
	int dest;
	char message[BUF_LEN];
	MPI_Status status;

	MPI_Init(&argc, &argv);
	MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
	MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
	if (my_rank != 0) {
		sprintf(message, "Hello from process %d", my_rank);
		dest=0;
		MPI_Send(message, strlen(message)+1, MPI_CHAR, dest,
MSG_TAG, MPI_COMM_WORLD);//відправка повідомлення
		//адреса буфера з даними, кількість елементів у буфері,тип
даних кожного із елементів, ранг отрмувача, тег повідомлення,
комунікатор
	}
	else{
		for (source=1; source<numprocs; source++) {
		MPI_Recv(message, BUF_LEN, MPI_CHAR, source, MSG_TAG,
MPI_COMM_WORLD, &status);//одержання повідомлення
		printf("%s\n",message);
		}
	}
	MPI_Finalize();
```

```
    return 0;
}
```

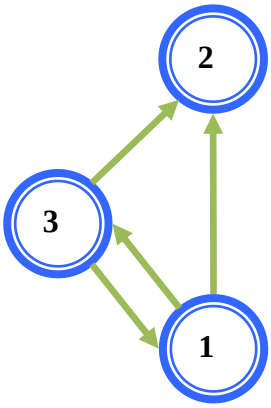
II. Практична частина

Інструкції до виконання

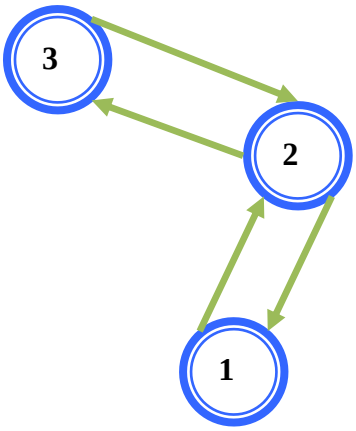
1. Опрацювати теорію.
2. Розібрати роботу програм з теоретичної частини.
3. Написати MPI-програму, яка б виконувала обмін між процесами за схемою відповідно до варіанту.



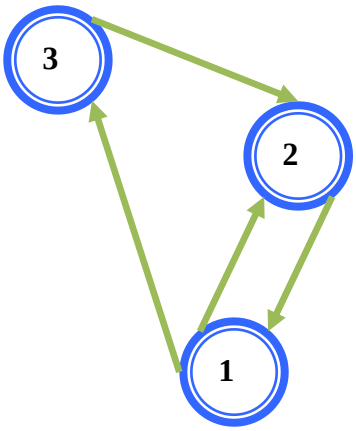
Варіант №1,7,13, 19



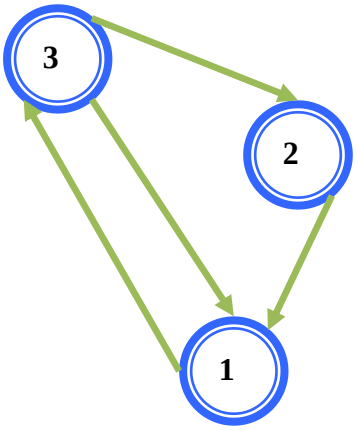
Варіант №2,8,14,20



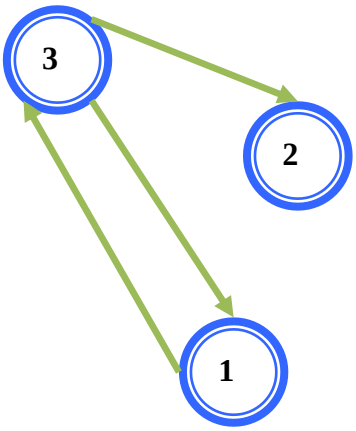
Варіант №3,9,15,21



Варіант №4,10,16,22



Варіант №5, 11,17,23



Варіант №6, 12, 18,24

Поточні контрольні питання:

1. Яка загальна будова MPI-програм?
2. Які виділяють види обмінів повідомленнями?
3. Які основні складові повідомлення?
4. Яка функція дає можливість визначити ранг (кількість) процесів?
5. Яка функція дає можливість передати повідомлення від одного процесу іншому?
6. Яка функція дає можливість отримати повідомлення від одного процесу до іншого?

Рекомендована література:

1. Семеренко В. П. Теорія циклічних кодів на основі автоматних моделей : монографія / Семеренко В. П. – Вінниця : ВНТУ, 2015. – 444 с.

2. Шеховцов В. А. Операційні системи / Шеховцов В. А. – Київ : Видавнича група BHV, 2005. – 576 с.

3. Кулаков А.Ю., Клименко І.А. Спосіб формування структури віртуальної GRID системи // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – К.: Век+, 2009. – № 50. - С. 97-100.

4. Кулаков О.Ю., Бролінський С.М., Ашаєв Ю.М. Динамічне створення віртуальних GRID систем для вирішення розподілених задач на основі менеджера ресурсів // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, 2009. – № 51. - С. 125-129.

5. Кулаков О.Ю., Бролінський С.М., Ашаєв Ю.М. Динамічне створення віртуальних GRID систем для вирішення розподілених задач на основі менеджера ресурсів // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, 2009. – № 51. - С. 125-129.

Тема 4. Колективні операції передачі даних

Лабораторна робота №4. Технологія паралельного програмування MPI. Використання паралельного алгоритму сумування елементів

Мета: ознайомити студентів з паралельним алгоритмом сумування елементів. Отримати навички його використання.

План

1. Порівняння часу виконання програми при різних характеристиках.
2. Розробка програми для обчислення визначеного інтеграла методом трапеції.

І. Теоретичні відомості

Визначення часу виконання MPI-програми

Отримати час поточного моменту виконання програми можна за допомогою функції:

```
double MPI_Wtime (void),
```

результатом виклику якої є кількість секунд, що минули від деякого певного моменту часу в минулому.

Можлива схема застосування функції MPI_Wtime може полягати в наступному:

```
double t1, t2, dt;
   t1 = MPI_Wtime();
   …
   t2 = MPI_Wtime();
   dt = t2 – t1;
```

Точність вимірювання часу також може залежати від середовища виконання паралельної програми. Для визначення поточного значення точності може бути використана функція:

```
double MPI_Wtick (void),
```

що дозволяє визначити час у секундах між двома послідовними показниками часу апаратного таймера комп'ютерної системи, що використовується.

Передача даних від одного процесу всім процесам програми

Досягнення ефективного виконання операції передачі даних від одного процесу всім процесам програми може бути забезпечено за допомогою функції MPI:

int MPI_Bcast (void * buf, int count, MPI_Datatype type, int root, MPI_Comm comm).

Функція MPI_Bcast здійснює розсилку даних з буфера buf, що містить count елементів типу type з процесу, що має номер root, всім процесам, що входять в комунікатор comm.

Передача даних від всіх процесів одному процесу. Операції редукції

Для найкращого виконання дій, пов'язаних з редукцією даних, в MPI передбачена функція:

int MPI_Reduce (void * sendbuf, void * recvbuf, int count, MPI_Datatype type, MPI_Op op, int root, MPI_Comm comm),

де

- Sendbuf - буфер пам'яті з повідомленням, що відправляється,
- Recvbuf - буфер пам'яті для результуючого повідомлення (тільки для процесу з рангом root),
- Count - кількість елементів у повідомленнях,
- Type - тип елементів повідомлень,
- Op - операція, яка повинна бути виконана над даними,
- Root - ранг процесу, на якому повинен бути отриманий результат,
- Comm - комунікатор, в рамках якого виконується операція.

Операція	Опис
MPI_MAX	Визначення максимального значення
MPI_MIN	Визначення мінімального значення
MPI_SUM	Визначення суми значень
MPI_PROD	Визначення добуток значень
MPI_LAND	Виконання логічної операції "І" над значеннями повідомлень
MPI_BAND	Виконання бітової операції "І" над значеннями повідомлень
MPI_LOR	Виконання логічної операції "АБО" над значеннями повідомлень
MPI_BOR	Виконання бітової операції "АБО" над значеннями повідомлень
MPI_LXOR	Виконання логічної операції виключає "АБО" над значеннями повідомлень
MPI_BXOR	Виконання бітової операції виключає "АБО" над значеннями повідомлень
MPI_MAXLOC	Визначення максимальних значень і їх індексів
MPI_MINLOC	Визначення мінімальних значень і їх індексів

Синхронізація обчислень

Синхронізація процесів, тобто одночасне досягнення процесами тих чи інших точок процесу обчислень, забезпечується за допомогою функції MPI:

```
int MPI_Barrier (MPI_Comm comm);
```

Функція MPI_Barrier визначає колективну операцію і, тим самим, при використанні повинна викликатися всіма процесами комунікатора, що використовується. При виклику функції MPI_Barrier виконання процесу блокується, продовження обчислень процесу відбудеться тільки після виклику функції MPI_Barrier всіма процесами комунікатора.

Приклад №4. Знайти суму елементів: $S = \sum_{i=1}^{n} x_i$.

Розробка паралельного алгоритму для рішення даної задачі не викликає труднощів – необхідно розділити дані на рівні блоки, передати ці блоки процесам, виконати в процесах підсумовування отриманих даних, зібрати значення обчислених часткових сум на одному з процесів і скласти значення часткових сум для отримання загального результату розв'язуваної задачі.

```c
#include <stdio.h>
#include "mpi.h"
#include <stdlib.h>


int main(int argc, char *argv[])
{
      int my_rank;
      int numprocs;
      int n,i,sum;
      int a;
      int result;
      double wtime;

      MPI_Init(&argc, &argv);
      MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
      MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
      if (my_rank == 0) {
            printf("Input n:");
            fflush( stdout );
            scanf("%d",&n);
            }

            MPI_Bcast(&n,1,MPI_INT,0,MPI_COMM_WORLD);
            MPI_Barrier(MPI_COMM_WORLD);
            wtime=MPI_Wtime();

            sum=0;
            for (i= my_rank; i<=n; i+=numprocs)
            {     a=rand()%n+1;
                  sum=sum+a;
            }

            MPI_Reduce(&sum,
&result,1,MPI_INT,MPI_SUM,0,MPI_COMM_WORLD);
            MPI_Barrier(MPI_COMM_WORLD);
            wtime=MPI_Wtime()-wtime;
            if (my_rank == 0) {

            printf("suma=%d",result);
            printf("Working time: %.8lf seconds\n",wtime);

      }
      MPI_Finalize();
      return 0;
}
```

Математичні функції на С++:

Функція	Опис	Приклад
abs(a)	модуль або абсолютне значення від a	abs(-3.0)= 3.0 abs(5.0)= 5.0
sqrt(a)	корінь квадратний з a, причому a не від'ємне	sqrt(9.0)=3.0
pow(a, b)	піднесення a у степінь b	pow(2,3)=8
ceil(a)	округлення a до найменшого цілого, але не менше ніж a	ceil(2.3)=3.0 ceil(-2.3)=-2.0
floor(a)	округлення a до найбільшого цілого, але не більше ніж a	floor(12.4)=12 floor(-2.9)=-3
fmod(a, b)	обчислення остачі від a/b	fmod(4.4, 7.5) = 4.4 fmod(7.5, 4.4) = 3.1
exp(a)	обчислення експоненти e^a	exp(0)=1
sin(a)	a задається у радіанах	
cos(a)	a задається у радіанах	
log(a)	натуральний логарифм a(основою є експонента)	log(1.0)=0.0
log10(a)	десятковий логарифм a	Log10(10)=1
asin(a)	арксинус a, де -1.0 < a < 1.0	asin(1)=1.5708

Таблиця 1 — Математичні функції у С++

II. Практична частина

Інструкції до виконання

1. Застосовуючи технологію паралельного програмування MPI, написати програму для обчислення значення e^x при $x=1$. Для знаходження значення e^x застосувати ряд Тейлора. Заповнити порівняльну таблицю часу виконання програми при різних характеристиках

	T	$n=10$	$n=100$	$n=1000$
Один комп'ютер	1 процес			
	2 процеси			
	3 процеси			
Два комп'ютер	1 процес			
	2 процеси			
	3 процеси			

2. Застосовуючи технологію паралельного програмування MPI, написати програму для обчислення значення визначеного інтеграла, використовуючи метод трапеції. Завдання згідно варіанту *(Додаток №1).*

Скласти порівняльну таблицю часу виконання програми при різних характеристиках.

3. Для роботи з програмою на двох комп'ютерах потрібно створити спільний ресурс.

Додаток №1

1. $\int_{-2}^{0}(x^2+5x+6)\cos 2x dx.$
2. $\int_{-2}^{0}(x^2-4)\cos 3x dx.$
3. $\int_{-1}^{0}(x^2+4x+3)\cos x dx.$
4. $\int_{-2}^{0}(x+2)^2\cos 3x dx.$
5. $\int_{-4}^{0}(x^2+7x+12)\cos x dx.$
6. $\int_{0}^{\pi}(2x^2+4x+7)\cos 2x dx.$
7. $\int_{0}^{\pi}(9x^2+9x+11)\cos 3x dx.$
8. $\int_{0}^{\pi}(8x^2+16x+17)\cos 4x dx.$
17. $\int_{-3}^{0}(x^2+6x+9)\sin 2x dx.$
18. $\int_{0}^{\pi/4}(x^2+17{,}5)\sin 2x dx.$
19. $\int_{0}^{\pi/2}(1-5x^2)\sin x dx.$
20. $\int_{\pi/4}^{3}(3x-x^2)\sin 2x dx.$
21. $\int_{1}^{2}x\ln^2 x dx.$
22. $\int_{1}^{e^2}\frac{\ln^2 x dx}{\sqrt{x}}.$
23. $\int_{1}^{8}\frac{\ln^2 x dx}{3\sqrt{x^2}}.$
24. $\int_{0}^{1}(x+1)\ln^2(x+1)dx.$

9. $\int_0^{2\pi}(3x^2+5)\cos 2x dx.$

10. $\int_0^{2\pi}(2x^2-15)\cos 3x dx.$

11. $\int_0^{2\pi}(3-7x^2)\cos 2x dx.$

12. $\int_0^{2\pi}(1-8x^2)\cos 4x dx.$

13. $\int_{-1}^{0}(x^2+2x+1)\sin 3x dx.$

14. $\int_0^{3}(x^2-2x)\sin 2x dx.$

15. $\int_0^{\pi}(x^2-3x+2)\sin x dx.$

16. $\int_0^{\pi/2}(x^2-5x+6)\sin 3x dx.$

25. $\int_2^{3}(x-1)^3\ln^2(x-1)dx.$

26. $\int_{-1}^{0}(x+2)^3\ln^2(x+2)dx.$

27. $\int_0^{2}(x+1)^2\ln^2(x+1)dx.$

28. $\int_1^{e}\sqrt{x}\ln^2 x dx.$

29. $\int_{-1}^{1}x^2e^{-x/2}dx.$

30. $\int_0^{1}x^2e^{3x}dx.$

31. $\int_{-2}^{0}(x^2+2)e^{x/2}dx.$

Поточні контрольні питання:

1. Яка загальна будова MPI-програм?
2. Які виділяють види обмінів повідомленнями?
3. Які основні складові повідомлення?
4. Яка функція дає можливість визначити ранг (кількість) процесів?
5. Яка функція дає можливість передати повідомлення від одного процесу іншому?
6. Яка функція дає можливість отримати повідомлення від одного процесу до іншого?

Рекомендована література:

1. Семеренко В. П. Теорія циклічних кодів на основі автоматних моделей : монографія / Семеренко В. П. – Вінниця : ВНТУ, 2015. – 444 с.
2. Шеховцов В. А. Операційні системи / Шеховцов В. А. – Київ : Видавнича група BHV, 2005. – 576 с.
3. Кулаков А.Ю., Клименко І.А. Спосіб формування структури віртуальної GRID системи // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – К.: Век+, 2009. – № 50. - С. 97-100.
4. Кузьменко Б.В., Чайковська О.А. Технологія розподілених систем та паралельних обчислень. (конспект лекцій, частина 1. Розподілені об’єктні системи, паралельні обчислювальні системи та паралельні обчислення, паралельне програмування на основі MPI) Навчальний посібник. – К.: Видавничий центр КНУКІМ, 2011 – 126 с.
5. Семеренко В. П. Технології паралельних обчислень : навчальний посібник / Семеренко В. П. – Вінниця : ВНТУ, 2018. – 104 с.
6. Петренко А. І. Практикум з грід-технологій : навчальний посібник / А. І. Петренко, С. Я. Свістунов, Г. Д. Кисельов. – К. : НТУУ «КПІ», 2011. – 580 с.
7. Петренко А.І. Вступ до Grid-технологій в науці і освіті (навчальний посібник). Київ: Політехніка. – 2008. – 124 с.

Тема 5. Найпростіші паралельні алгоритми

Лабораторна робота №5. Паралельний алгоритм Флойда_Уоршела

Мета: ознайомити студентів з паралельним алгоритмом Флойда-Уоршела.

План

1. Застосування технології паралельного програмування MPI для написання програми знаходження найкоротших шляхів на графі.
2. Порівняння швидкості виконання задачі при різній кількості комп'ютерів та процесів.

І. Теоретичні відомості

1. Колективні операції передачі даних

1.1. Узагальнена передача даних від одного процесу всім процесам

Виконання даної операції може бути забезпечено за допомогою функції:

int MPI_Scatter (void * sbuf, int scount, MPI_Datatype stype, void * rbuf, int rcount, MPI_Datatype rtype, int root, MPI_Comm comm), де

- sbuf, scount, stype - параметри повідомлення, що передається (scount визначає кількість елементів, що передаються на кожен процес),
- rbuf, rcount, rtype - параметри повідомлення, що приймається в процесах,
- root - ранг процесу, що виконує розсилку даних,
- comm - комунікатор, в рамках якого виконується передача даних.

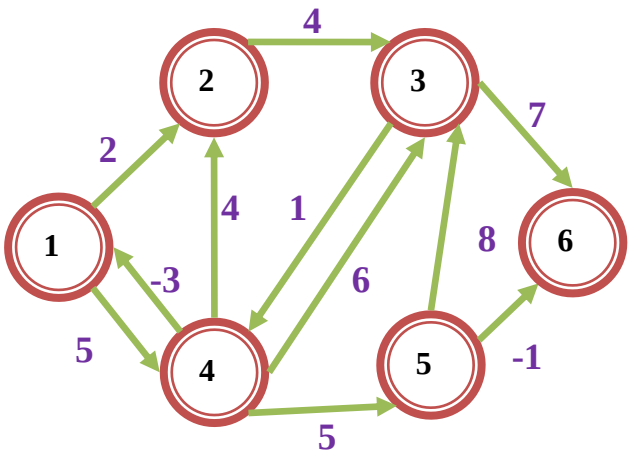
1.2. Узагальнена передача даних від всіх процесів одному процесу

Для виконання цієї операції в MPI призначена функція:

int MPI_Gather (void * sbuf, int scount, MPI_Datatype stype, void * rbuf, int rcount, MPI_Datatype rtype, int root, MPI_Comm comm), де

- sbuf, scount, stype - параметри повідомлення, що передається,
- rbuf, rcount, rtype - параметри повідомлення, що приймається,
- root - ранг процесу, що виконує збір даних,
- comm - комунікатор, у рамках якого виконується передача даних.

2. Паралельний алгоритм Флойда-Уоршелла



$k = 0$

$my\_rank = 0$

0	1	2	3	4	5	6
1	0	2	∞	5	∞	∞
2	∞	0	4	∞	∞	∞
3	∞	∞	0	1	∞	7
4	-3	4	6	0	5	∞
5	∞	∞	8	∞	0	-1
6	∞	∞	∞	∞	∞	0

При $k = 1...n,\ n = 6$

- головний процес відсилає k-й рядок та k-й стовпчик решті процесам

$my\_rank = 0$

	1	2	3	4	5	6
1	0	2	∞	5	∞	∞
2	∞					
3	∞					
4	-3					
5	∞					
6	∞					

$my\_rank = 1$

$my\_rank = 2$

- далі головний процес відправляє всім процесам по n/numprocs рядків

$my\_rank = 0$

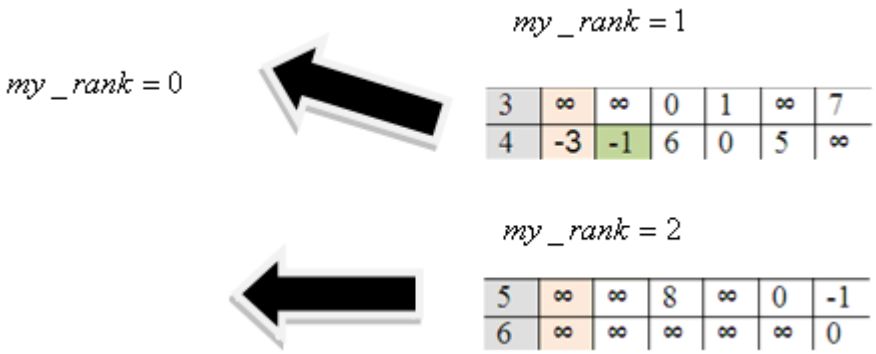
$my\_rank = 1$

3	∞	∞	0	1	∞	7
4	-3	4	6	0	5	∞

$my\_rank = 2$

5	∞	∞	8	∞	0	-1
6	∞	∞	∞	∞	∞	0

- потім кожен процес обчислює свої рядки (включаючи 1-й) і відсилає їх на головний (у нашому випадку 0).



- головний процес збираю матрицю:

1	1	2	3	4	5	6
1	0	2	∞	5	∞	∞
2	∞	0	4	∞	∞	∞
3	∞	∞	0	1	∞	7
4	-3	-1	6	0	5	∞
5	∞	∞	8	∞	0	-1
6	∞	∞	∞	∞	∞	0

далі $k = 2$ і т. д.

3. Передача масиву з головного процесу на інші

```
#include <stdio.h>
#include "mpi.h"
int main(int argc, char *argv[])
{
int my_rank;
int numprocs;
int n,i;
int a[100];
int temp[100];
int local_n;
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
if (my_rank == 0) {
printf("Input n:");
fflush( stdout );
scanf("%d",&n);
for (i=0; i<n; i++){
printf("Input a[%d]",i);
fflush( stdout );
scanf("%d",&a[i]);
}
}
MPI_Bcast(&n,1,MPI_INT,0,MPI_COMM_WORLD);
local_n=n/numprocs;
MPI_Scatter(a,local_n,MPI_INT,temp,local_n,MPI_INT,0,MPI_COMM_WOR
LD);
for (i=0; i<local_n; i++){
```

```
printf("%d",temp[i]);
}
printf("\n");
MPI_Finalize();
return 0;
}
```

4. Передача масиву з усіх процесів на головний

```
#include <stdio.h>
#include "mpi.h"
int main(int argc, char *argv[])
{
int my_rank;
int numprocs;
int n,i;
int a[100];
int temp[100];
int local_n;
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
if (my_rank == 0) {
printf("Input n:");
fflush( stdout );
scanf("%d",&n);
}
MPI_Bcast(&n,1,MPI_INT,0,MPI_COMM_WORLD);
local_n=n/numprocs;
for (i=my_rank; i<n; i+=numprocs){
printf("Input a[%d]",i);
fflush( stdout );
scanf("%d",&a[i]);
}
MPI_Gather(a,local_n,MPI_INT,temp,local_n,MPI_INT,0,MPI_COMM_WORL
D);
if (my_rank==0){
for (i=0; i<n; i++){
printf("%d",temp[i]);
}
}
MPI_Finalize();
return 0;
}
```

ІІ. Практична частина

Інструкції до виконання

1. Застосовуючи технологію паралельного програмування MPI, написати програму для знаходження найкоротших шляхів на заданому графі

(варіанти згідно Додатку №3 лабораторної роботи №2) для будь-якої пари вершин графу, використовуючи паралельний алгоритм Флойда – Уоршелла.

2. Зробити висновки про швидкість виконання задачі при різній кількості комп'ютерів та процесів.

Поточні контрольні питання:

1. За допомогою якої функції можна здійснити передачу даних від одного процесу всім іншим?
2. За допомогою якої функції можна здійснити передачу даних від всіх процесів до одного процесу?
3. Опишіть паралельний алгоритм Флойда-Уоршела.

Рекомендована література:

1. Паралельні та розподілені обчислення [Текст] : навч. підруч. для студентів вищ. навч. закл. / А. Луцків, С. Луценко, В. Пасічник. – Львів : Магнолія 2006, 2017. – 565, [1] с. : схеми.
2. Аксак Н. Г. Паралельні та розподілені обчислення: підруч. / Н. Г. Аксак, О. Г. Руденко, А.М. Гуржій. – Х.: Компанія СМІТ, 2009. – 480 с.
3. Паралельні та розподілені обчислення: навчальний посібник для вищих закладів освіти / К.Т. Кузьма, О.В. Мельник. – Миколаїв: ФОП Швець В.М., 2020. – 172 с.
4. Шеховцов В. А. Операційні системи / Шеховцов В. А. – Київ : Видавнича група BHV, 2005. – 576 с.
5. Кулаков А.Ю., Клименко І.А. Спосіб формування структури віртуальної GRID системи // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – К.: Век+, 2009. – № 50. - С. 97-100.
6. Кузьменко Б.В., Чайковська О.А. Технологія розподілених систем та паралельних обчислень. (конспект лекцій, частина 1. Розподілені об'єктні системи, паралельні обчислювальні системи та паралельні обчислення, паралельне програмування на основі MPI) Навчальний посібник. – К.: Видавничий центр КНУКІМ, 2011 – 126 с.
7. Семеренко В. П. Технології паралельних обчислень : навчальний посібник / Семеренко В. П. – Вінниця : ВНТУ, 2018. – 104 с.
8. Петренко А. І. Практикум з грід-технологій : навчальний посібник / А. І. Петренко, С. Я. Свістунов, Г. Д. Кисельов. – К. : НТУУ «КПІ», 2011. – 580 с.
9. Згуровський М.З., Петренко А.І. Е-наука на шляху до семантичного Грід. Частина 1: Об'єднання Web- і Грід- технологій // Системні дослідження та інформаційні технології. – Київ, №1, 2010. – С. 26-38.
10. Згуровський М.З., Петренко А.І. Е-наука на шляху до семантичного Грід. Частина 2: Семантичний Web- і семантичний Грід // Системні дослідження та інформаційні технології. – Київ, №2, 2010. – С. 7-25.

Лабораторна робота №6. Алгоритми паралельного матрично-векторного множення

Мета: ознайомитися з паралельними алгоритмами матрично-векторного множення.

План

1. Написання програми для множення матриці на вектор з використанням технології паралельного програмування MPI.
2. Написання програми для множення матриці на матрицю з використанням технології паралельного програмування MPI.

I. Теоретичні відомості

Послідовне множення матриці на вектор

$$A \times B = C.$$

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \dots & .. & \dots & \dots & \dots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{pmatrix}, \; B = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \dots \\ b_n \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \dots & .. & \dots & \dots & \dots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \dots \\ b_n \end{pmatrix} = \begin{pmatrix} a_{11}b_1 + a_{12}b_2 + a_{13}b_3 \dots a_{1n}b_n \\ a_{21}b_1 + a_{22}b_2 + a_{23}b_3 \dots a_{2n}b_n \\ a_{31}b_1 + a_{32}b_2 + a_{33}b_3 \dots a_{3n}b_n \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ a_{m1}b_1 + a_{m2}b_2 + a_{m3}b_3 \dots a_{mn}b_n \end{pmatrix}$$

$$C = \begin{pmatrix} a_{11}b_1 + a_{12}b_2 + a_{13}b_3 \dots a_{1n}b_n \\ a_{21}b_1 + a_{22}b_2 + a_{23}b_3 \dots a_{2n}b_n \\ a_{31}b_1 + a_{32}b_2 + a_{33}b_3 \dots a_{3n}b_n \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ a_{m1}b_1 + a_{m2}b_2 + a_{m3}b_3 \dots a_{mn}b_n \end{pmatrix}.$$

Приклад

$$\begin{pmatrix} 1 & -2 & 3 & 4 \\ -4 & 5 & 6 & 8 \\ -1 & -3 & 0 & 2 \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \\ 2 \\ 3 \end{pmatrix}.$$

Розв’язання

$$\begin{pmatrix} 1 & -2 & 3 & 4 \\ -4 & 5 & 6 & 8 \\ -1 & -3 & 0 & 2 \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 1\cdot 1 + (-2)\cdot 0 + 3\cdot 2 + 4\cdot 3 \\ -4\cdot 1 + 5\cdot 0 + 6\cdot 2 + 8\cdot 3 \\ -1\cdot 1 + (-3)\cdot 0 + 0\cdot 2 + 2\cdot 3 \end{pmatrix} = \begin{pmatrix} 19 \\ 32 \\ 5 \end{pmatrix}.$$

Відповідь. $C = \begin{pmatrix} 19 \\ 32 \\ 5 \end{pmatrix}$

Алгоритм паралельного множення матриці на вектор

1) Визначаємо кількість процесів.

2) На головному процесі розділяємо матрицю A на смужки по рядках та розділяємо їх між процесами.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{pmatrix}$$

3) З головного процесу надсилаємо на кожен процес вектор B $\begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \dots \\ b_n \end{pmatrix}$

4) На кожному процесі виконуємо множення відповідного рядка на вектор:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \dots \\ b_n \end{pmatrix} = \begin{pmatrix} a_{11}b_1 + a_{12}b_2 + a_{13}b_3 \dots a_{1n}b_n \end{pmatrix}$$

.

5) Результат множення з кожного процесу надсилається на головний процес і на ньому виконується збір результуючої матриці.

6) $$C = \begin{pmatrix} a_{11}b_1 + a_{12}b_2 + a_{13}b_3 \dots a_{1n}b_n \\ a_{21}b_1 + a_{22}b_2 + a_{23}b_3 \dots a_{2n}b_n \\ a_{31}b_1 + a_{32}b_2 + a_{33}b_3 \dots a_{3n}b_n \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ a_{m1}b_1 + a_{m2}b_2 + a_{m3}b_3 \dots a_{mn}b_n \end{pmatrix}$$

Приклад

$$\begin{pmatrix} 1 & -2 & 3 & 4 \\ -4 & 5 & 6 & 8 \\ -1 & -3 & 0 & 2 \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \\ 2 \\ 3 \end{pmatrix}.$$

Алгоритм розв’язання:

1) Нехай ми маємо 3 процеси.

2) Розділяємо матрицю на смужки і надсилаємо:

на процесі з рангом 0 (головний) залишиться смужка $\begin{pmatrix} 1 & -2 & 3 & 4 \end{pmatrix}$; на процес з рангом 1 відправиться смужка $\begin{vmatrix} -4 & 5 & 6 & 8 \end{vmatrix}$, тоді на процес з рангом 2 -- $\begin{pmatrix} -1 & -3 & 0 & 2 \end{pmatrix}$.

3) З головного процесу надсилаємо на кожен процес вектор B $\begin{pmatrix} 1 \\ 0 \\ 2 \\ 3 \end{pmatrix}$

4) На кожному процесі виконуємо множення відповідного рядка на вектор:

на процесі з рангом 0:

$$\begin{pmatrix} 1 & -2 & 3 & 4 \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 \cdot 1 + (-2) \cdot 0 + 3 \cdot 2 + 4 \cdot 3 \end{pmatrix}$$

на процесі з рангом 1:

$$\begin{vmatrix} -4 & 5 & 6 & 8 \end{vmatrix} \times \begin{pmatrix} 1 \\ 0 \\ 2 \\ 3 \end{pmatrix} = \begin{vmatrix} -4 \cdot 1 + 5 \cdot 0 + 6 \cdot 2 + 8 \cdot 3 \end{vmatrix}$$

на процесі з рангом 2:

$$\begin{pmatrix} -1 & -3 & 0 & 2 \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} -1 \cdot 1 + (-3) \cdot 0 + 0 \cdot 2 + 2 \cdot 3 \end{pmatrix}$$

5) Результат множення з кожного процесу надсилається на головний процес і на ньому виконується збір результуючої матриці.

$$C = \begin{pmatrix} 19 \\ 32 \\ 5 \end{pmatrix}.$$

Послідовне множення матриці на матрицю

$$A \times B = C.$$

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \dots & .. & \dots & \dots & \dots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{pmatrix}, \quad B = \begin{pmatrix} b_{11} & b_{12} & b_{13} & \dots & b_{1k} \\ b_{21} & b_{22} & b_{23} & \dots & b_{2k} \\ b_{31} & b_{32} & b_{33} & \dots & b_{3k} \\ \dots & .. & \dots & \dots & \dots \\ b_{n1} & b_{n2} & b_{n3} & \dots & b_{nk} \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} & b_{13} & \dots & b_{1k} \\ b_{21} & b_{22} & b_{23} & \dots & b_{2k} \\ b_{31} & b_{32} & b_{33} & \dots & b_{3k} \\ \dots & \dots & \dots & \dots & \dots \\ b_{n1} & b_{n2} & b_{n3} & \dots & b_{nk} \end{pmatrix} =$$

$$= \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} \dots a_{1n}b_{n1} & \dots & \dots & \dots & a_{11}b_{1k} + a_{12}b_{2k} + a_{13}b_{3k} \dots a_{1n}b_{nk} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} \dots a_{2n}b_{n1} & \dots & \dots & \dots & a_{21}b_{1k} + a_{22}b_{2k} + a_{23}b_{3k} \dots a_{2n}b_{nk} \\ a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} \dots a_{3n}b_{n1} & \dots & \dots & \dots & a_{31}b_{1k} + a_{32}b_{2k} + a_{33}b_{3k} \dots a_{3n}b_{nk} \\ \dots & \dots & \dots & \dots & \dots \\ a_{m1}b_{11} + a_{m2}b_{21} + a_{m3}b_{31} \dots a_{mn}b_{n1} & \dots & \dots & \dots & a_{m1}b_{1k} + a_{m2}b_{2k} + a_{m3}b_{3k} \dots a_{mn}b_{nk} \end{pmatrix}.$$

$$C = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} \dots a_{1n}b_{n1} & \dots & \dots & \dots & a_{11}b_{1k} + a_{12}b_{2k} + a_{13}b_{3k} \dots a_{1n}b_{nk} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} \dots a_{2n}b_{n1} & \dots & \dots & \dots & a_{21}b_{1k} + a_{22}b_{2k} + a_{23}b_{3k} \dots a_{2n}b_{nk} \\ a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} \dots a_{3n}b_{n1} & \dots & \dots & \dots & a_{31}b_{1k} + a_{32}b_{2k} + a_{33}b_{3k} \dots a_{3n}b_{nk} \\ \dots & \dots & \dots & \dots & \dots \\ a_{m1}b_{11} + a_{m2}b_{21} + a_{m3}b_{31} \dots a_{mn}b_{n1} & \dots & \dots & \dots & a_{m1}b_{1k} + a_{m2}b_{2k} + a_{m3}b_{3k} \dots a_{mn}b_{nk} \end{pmatrix}.$$

Приклад

$$\begin{pmatrix} 1 & -2 & 3 & 4 \\ -4 & 5 & 6 & 8 \\ -1 & -3 & 0 & 2 \end{pmatrix} \times \begin{pmatrix} 1 & 3 \\ 0 & 4 \\ 2 & 0 \\ 0 & 5 \end{pmatrix}.$$

Розв’язання

$$\begin{pmatrix} 1 & -2 & 3 & 4 \\ -4 & 5 & 6 & 8 \\ -1 & -3 & 0 & 2 \end{pmatrix} \times \begin{pmatrix} 1 & 3 \\ 0 & 4 \\ 2 & 0 \\ 0 & 5 \end{pmatrix} = \begin{pmatrix} 1\cdot 1 + (-2)\cdot 0 + 3\cdot 2 + 4\cdot 0 & 1\cdot 3 + (-2)\cdot 4 + 3\cdot 0 + 4\cdot 5 \\ -4\cdot 1 + 5\cdot 0 + 6\cdot 2 + 8\cdot 0 & -4\cdot 3 + 5\cdot 4 + 6\cdot 0 + 8\cdot 5 \\ -1\cdot 1 + (-3)\cdot 0 + 0\cdot 2 + 2\cdot 0 & -1\cdot 3 + (-3)\cdot 4 + 0\cdot 0 + 2\cdot 5 \end{pmatrix} = \begin{pmatrix} 7 & 15 \\ 8 & 48 \\ -1 & -5 \end{pmatrix}$$

Відповідь. $C = \begin{pmatrix} 7 & 15 \\ 8 & 48 \\ -1 & -5 \end{pmatrix}$

Алгоритм паралельного множення матриці на матрицю

1) Визначаємо кількість процесів.

2) На головному процесі розділяємо матрицю *A* на смужки по рядках та розділяємо їх між процесами.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \end{pmatrix}$$

$$\begin{pmatrix} a_{21} & a_{22} & a_{23} & \dots & a_{2n} \end{pmatrix}$$

$$\begin{pmatrix} a_{31} & a_{32} & a_{33} & \dots & a_{3n} \end{pmatrix}$$

$$\begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix}$$

$$\begin{pmatrix} a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{pmatrix}$$

3) ***K*** дорівнює кількості стовпців у матриці B.

4) З головного процесу надсилаємо на кожен процес і-й стовпець матриці *B* $\begin{pmatrix} b_{11} \\ b_{21} \\ b_{31} \\ \cdots \\ b_{n1} \end{pmatrix}$

5) На кожному процесі виконуємо множення відповідного рядка на і-й стовпець матриці B

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \end{pmatrix} \times \begin{pmatrix} b_{11} \\ b_{21} \\ b_{31} \\ \cdots \\ b_{n1} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} \cdots a_{1n}b_{n1} \end{pmatrix}.$$

6) Результат множення з кожного процесу надсилається на головний процес і на ньому виконується збір відповідного стовпця результуючої матриці.

$$\begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} \cdots a_{1n}b_{n1} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} \cdots a_{2n}b_{n1} \\ a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} \cdots a_{3n}b_{n1} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_{m1}b_{11} + a_{m2}b_{21} + a_{m3}b_{31} \cdots a_{mn}b_{n1} \end{pmatrix}$$

7) ***i*** збільшуємо на 1 і починаємо з 4) поки i<=k.

Приклад

$$\begin{pmatrix} 1 & -2 & 3 & 4 \\ -4 & 5 & 6 & 8 \\ -1 & -3 & 0 & 2 \end{pmatrix} \times \begin{pmatrix} 1 & 3 \\ 0 & 4 \\ 2 & 0 \\ 0 & 5 \end{pmatrix}.$$

Алгоритм розв’язання:

1) Нехай ми маємо 3 процеси.

2) Розділяємо матрицю на смужки і надсилаємо:

на процесі з рангом 0 (головний) залишиться смужка $\begin{pmatrix} 1 & -2 & 3 & 4 \end{pmatrix}$; на процес з рангом 1 відправиться смужка $\begin{vmatrix} -4 & 5 & 6 & 8 \end{vmatrix}$, тоді на процес з рангом 2 -- $\begin{pmatrix} -1 & -3 & 0 & 2 \end{pmatrix}$.

3) $k = 2$.

4) З головного процесу надсилаємо на кожен процес ($i = 1$) і-й стовпець матриці B $\begin{pmatrix} 1 \\ 0 \\ 2 \\ 0 \end{pmatrix}$

5) На кожному процесі виконуємо множення відповідного рядка на і-й стовпець матриці B:

на процесі з рангом 0:

$$\begin{pmatrix} 1 & -2 & 3 & 4 \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \\ 2 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \cdot 1 + (-2) \cdot 0 + 3 \cdot 2 + 4 \cdot 0 \end{pmatrix}$$

на процесі з рангом 1:

$$\begin{vmatrix} -4 & 5 & 6 & 8 \end{vmatrix} \times \begin{pmatrix} 1 \\ 0 \\ 2 \\ 0 \end{pmatrix} = \begin{vmatrix} -4 \cdot 1 + 5 \cdot 0 + 6 \cdot 2 + 8 \cdot 0 \end{vmatrix}$$

на процесі з рангом 2:

$$\begin{pmatrix} -1 & -3 & 0 & 2 \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \\ 2 \\ 0 \end{pmatrix} = \begin{pmatrix} -1 \cdot 1 + (-3) \cdot 0 + 0 \cdot 2 + 2 \cdot 0 \end{pmatrix}$$

6) Результат множення з кожного процесу надсилається на головний процес і на ньому виконується збір відповідного стовпця результуючої матриці.

$$\begin{pmatrix} 1 \cdot 1 + (-2) \cdot 0 + 3 \cdot 2 + 4 \cdot 0 \\ -4 \cdot 1 + 5 \cdot 0 + 6 \cdot 2 + 8 \cdot 0 \\ -1 \cdot 1 + (-3) \cdot 0 + 0 \cdot 2 + 2 \cdot 0 \end{pmatrix}$$

.

7) ***i*** збільшуємо на 1 ($i = 2 \le k$) і починаємо з 4).

8) З головного процесу надсилаємо на кожен процес ($i = 2$) і-й стовпець матриці B $\begin{pmatrix} 3 \\ 4 \\ 0 \\ 5 \end{pmatrix}$

9) На кожному процесі виконуємо множення відповідного рядка на і-й стовпець матриці *B*:

на процесі з рангом 0:

$$\begin{pmatrix} 1 & -2 & 3 & 4 \end{pmatrix} \times \begin{pmatrix} 3 \\ 4 \\ 0 \\ 5 \end{pmatrix} = \begin{pmatrix} 1\cdot 3+(-2)\cdot 4+3\cdot 0+4\cdot 5 \end{pmatrix}$$

на процесі з рангом 1:

$$\begin{pmatrix} -4 & 5 & 6 & 8 \end{pmatrix} \times \begin{pmatrix} 3 \\ 4 \\ 0 \\ 5 \end{pmatrix} = \begin{pmatrix} -4\cdot 3+5\cdot 4+6\cdot 0+8\cdot 5 \end{pmatrix}$$

на процесі з рангом 2:

$$\begin{pmatrix} -1 & -3 & 0 & 2 \end{pmatrix} \times \begin{pmatrix} 3 \\ 4 \\ 0 \\ 5 \end{pmatrix} = \begin{pmatrix} -1\cdot 3+(-3)\cdot 4+0\cdot 0+2\cdot 5 \end{pmatrix}$$

10) Результат множення з кожного процесу надсилається на головний процес і на ньому виконується збір відповідного стовпця результуючої матриці.

$$\begin{pmatrix} 1\cdot 3+(-2)\cdot 4+3\cdot 0+4\cdot 5 \\ -4\cdot 3+5\cdot 4+6\cdot 0+8\cdot 5 \\ -1\cdot 3+(-3)\cdot 4+0\cdot 0+2\cdot 5 \end{pmatrix}.$$

11) В результаті на головному процесі ми матимемо матрицю:

$$C = \begin{pmatrix} 1\cdot 1+(-2)\cdot 0+3\cdot 2+4\cdot 0 & 1\cdot 3+(-2)\cdot 4+3\cdot 0+4\cdot 5 \\ -4\cdot 1+5\cdot 0+6\cdot 2+8\cdot 0 & -4\cdot 3+5\cdot 4+6\cdot 0+8\cdot 5 \\ -1\cdot 1+(-3)\cdot 0+0\cdot 2+2\cdot 0 & -1\cdot 3+(-3)\cdot 4+0\cdot 0+2\cdot 5 \end{pmatrix} = \begin{pmatrix} 7 & 15 \\ 8 & 48 \\ -1 & -5 \end{pmatrix}.$$

II. Практична частина

Інструкції до виконання

1. Виконати множення матриці на вектор.
2. Написати програму для множення матриці на вектор, використовуючи технологію паралельного програмування MPI.
3. Виконати множення матриці на матрицю.
4. Написати програму для множення матриці на матрицю, використовуючи технологію паралельного програмування MPI.

Для 1-го та 2-го завдання варіанти згідно *додатка №1*, для 3-го та 4-го завдання варіанти згідно *додатка №2.*

Поточні контрольні питання:

1. Опишіть алгоритм послідовного множення матриці на вектор.
2. Опишіть алгоритм паралельного множення матриці на вектор.
3. Опишіть алгоритм послідовного множення матриці на матрицю.
4. Опишіть алгоритм паралельного множення матриці на матрицю.

Рекомендована література:

1. Паралельні та розподілені обчислення [Текст] : навч. підруч. для студентів вищ. навч. закл. / А. Луцків, С. Луценко, В. Пасічник. – Львів : Магнолія 2006, 2017. – 565, [1] с. : схеми.

2. Паралельні та розподілені обчислення: навчальний посібник для вищих закладів освіти / К.Т. Кузьма, О.В. Мельник. – Миколаїв: ФОП Швець В.М., 2020. – 172 с.

3. Шеховцов В. А. Операційні системи / Шеховцов В. А. – Київ : Видавнича група BHV, 2005. – 576 с.

4. Кулаков А.Ю., Клименко І.А. Спосіб формування структури віртуальної GRID системи // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – К.: Век+, 2009. – № 50. - С. 97-100.

5. Семеренко В. П. Технології паралельних обчислень : навчальний посібник / Семеренко В. П. – Вінниця : ВНТУ, 2018. – 104 с.

6. Петренко А. І. Практикум з грід-технологій : навчальний посібник / А. І. Петренко, С. Я. Свістунов, Г. Д. Кисельов. – К. : НТУУ «КПІ», 2011. – 580 с.

7. Згуровський М.З., Петренко А.І. Е-наука на шляху до семантичного Грід. Частина 1: Об’єднання Web- і Грід- технологій // Системні дослідження та інформаційні технології. – Київ, №1, 2010. – С. 26-38.

Додаток №1

1. Варіант

$$\begin{pmatrix} 0 & 1 & 5 & 9 \\ 3 & 2 & -4 & -7 \\ 0 & 8 & -1 & 3 \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \\ -2 \\ 3 \end{pmatrix}$$

2. Варіант

$$\begin{pmatrix} 1 & -3 & 0 & 5 \\ -3 & 4 & -4 & -2 \\ 2 & -3 & -1 & 9 \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \\ -2 \\ 1 \end{pmatrix}$$

3. Варіант

$$\begin{pmatrix} 0 & 3 & 0 & 9 \\ -2 & 4 & -4 & 0 \\ 10 & -5 & 0 & 5 \end{pmatrix} \times \begin{pmatrix} 1 \\ 3 \\ 0 \\ 2 \end{pmatrix}$$

4. Варіант

$$\begin{pmatrix} 1 & 1 & 2 & 9 \\ -7 & 0 & 4 & 0 \\ 6 & 8 & 0 & -5 \end{pmatrix} \times \begin{pmatrix} 2 \\ -3 \\ 0 \\ 1 \end{pmatrix}$$

5. Варіант

$$\begin{pmatrix} 1 & 1 & 2 & 9 \\ -7 & 0 & 4 & 0 \\ 6 & 8 & 0 & -5 \end{pmatrix} \times \begin{pmatrix} 2 \\ -3 \\ 0 \\ 1 \end{pmatrix}$$

6. Варіант

$$\begin{pmatrix} 0 & 1 & 3 & -6 \\ 3 & 0 & 2 & 0 \\ 0 & 8 & -1 & -3 \end{pmatrix} \times \begin{pmatrix} 4 \\ 0 \\ 0 \\ -3 \end{pmatrix}$$

7. Варіант

$$\begin{pmatrix} 1 & 0 & -6 & 0 \\ 3 & 1 & 1 & -7 \\ 0 & 8 & -1 & 3 \end{pmatrix} \times \begin{pmatrix} 1 \\ 2 \\ -2 \\ 0 \end{pmatrix}$$

8. Варіант

$$\begin{pmatrix} -8 & 1 & 3 & 0 \\ 9 & 0 & -10 & -7 \\ 0 & 8 & -1 & 0 \end{pmatrix} \times \begin{pmatrix} -1 \\ 0 \\ -2 \\ -3 \end{pmatrix}$$

9. Варіант

$$\begin{pmatrix} 3 & -1 & 0 & -9 \\ 2 & 0 & 4 & 7 \\ 4 & 6 & -6 & 0 \end{pmatrix} \times \begin{pmatrix} 3 \\ 0 \\ 0 \\ 3 \end{pmatrix}$$

10. Варіант

$$\begin{pmatrix} 11 & -3 & 0 & 5 \\ -3 & 4 & -4 & -2 \\ 0 & -3 & -1 & 9 \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \\ -2 \\ 1 \end{pmatrix}$$

11. Варіант

$$\begin{pmatrix} 0 & 2 & 0 & -9 \\ -2 & -4 & -6 & 5 \\ 10 & 0 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 7 \\ 3 \\ 1 \\ 4 \end{pmatrix}$$

12. Варіант

$$\begin{pmatrix} 0 & 1 & 0 & 3 \\ -2 & 0 & 4 & 0 \\ 0 & 7 & 0 & -5 \end{pmatrix} \times \begin{pmatrix} -2 \\ -3 \\ 0 \\ -1 \end{pmatrix}$$

13. Варіант

$$\begin{pmatrix} 1 & 0 & 2 & 6 \\ 0 & 6 & 0 & 7 \\ -3 & 8 & -4 & 0 \end{pmatrix} \times \begin{pmatrix} 4 \\ -5 \\ 1 \\ -2 \end{pmatrix}$$

14. Варіант

$$\begin{pmatrix} 3 & 1 & 3 & 0 \\ 0 & 2 & 0 & 4 \\ -4 & 0 & -1 & -5 \end{pmatrix} \times \begin{pmatrix} 0 \\ 3 \\ 2 \\ -7 \end{pmatrix}$$

15. Варіант

$$\begin{pmatrix} 2 & 1 & -6 & 0 \\ 3 & 0 & -1 & 5 \\ 0 & 8 & -2 & 9 \end{pmatrix} \times \begin{pmatrix} 8 \\ 0 \\ -2 \\ 0 \end{pmatrix}$$

16. Варіант

$$\begin{pmatrix} -3 & 2 & 3 & 0 \\ 1 & 0 & 5 & -7 \\ 0 & -4 & -6 & 0 \end{pmatrix} \times \begin{pmatrix} -1 \\ 0 \\ 0 \\ -3 \end{pmatrix}$$

17. Варіант

$$\begin{pmatrix} 3 & 1 & 0 & 9 \\ -2 & 0 & 4 & 0 \\ 0 & 8 & -1 & 3 \end{pmatrix} \times \begin{pmatrix} -1 \\ -2 \\ -3 \\ -4 \end{pmatrix}$$

18. Варіант

$$\begin{pmatrix} 0 & -2 & 7 & 5 \\ -1 & 4 & -5 & -6 \\ 2 & -3 & -1 & 10 \end{pmatrix} \times \begin{pmatrix} 3 \\ 4 \\ -5 \\ -6 \end{pmatrix}$$

19. Варіант

$$\begin{pmatrix} 1 & 3 & 0 & 9 \\ 0 & 0 & -7 & 0 \\ 10 & -4 & 0 & 5 \end{pmatrix} \times \begin{pmatrix} 4 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

20. Варіант

$$\begin{pmatrix} 0 & 1 & -2 & -9 \\ -3 & 0 & 8 & 0 \\ 2 & 8 & 0 & -5 \end{pmatrix} \times \begin{pmatrix} 1 \\ -2 \\ 3 \\ -4 \end{pmatrix}$$

21. Варіант

$$\begin{pmatrix} 1 & 1 & 2 & 3 \\ 3 & 1 & 3 & 4 \\ 2 & 3 & 1 & 2 \end{pmatrix} \times \begin{pmatrix} 4 \\ -5 \\ 0 \\ 6 \end{pmatrix}$$

22. Варіант

$$\begin{pmatrix} 3 & 1 & -3 & -2 \\ 4 & 0 & 2 & 0 \\ 5 & 4 & -5 & -3 \end{pmatrix} \times \begin{pmatrix} 5 \\ 1 \\ 2 \\ -5 \end{pmatrix}$$

23. Варіант

$$\begin{pmatrix} -1 & 2 & -2 & 3 \\ 1 & -1 & 2 & -3 \\ -2 & 1 & -2 & 3 \end{pmatrix} \times \begin{pmatrix} -4 \\ 5 \\ -6 \\ 7 \end{pmatrix}$$

24. Варіант

$$\begin{pmatrix} 1 & 2 & 0 & 4 \\ 0 & 2 & 3 & 0 \\ 1 & 0 & 3 & 4 \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \\ 2 \\ 3 \end{pmatrix}$$

Додаток №2

1. Варіант

$$\begin{pmatrix} 0 & 1 & 5 & 9 \\ 3 & 2 & -4 & -7 \\ 0 & 8 & -1 & 3 \end{pmatrix} \times \begin{pmatrix} 1 & 8 \\ 7 & 2 \\ 3 & 6 \\ 5 & 4 \end{pmatrix}$$

2. Варіант

$$\begin{pmatrix} 1 & -3 & 0 & 5 \\ -3 & 4 & -4 & -2 \\ 2 & -3 & -1 & 9 \end{pmatrix} \times \begin{pmatrix} 1 & 6 \\ 0 & 2 \\ 3 & 0 \\ 5 & 4 \end{pmatrix}$$

3. Варіант

$$\begin{pmatrix} 0 & 3 & 0 & 9 \\ -2 & 4 & -4 & 0 \\ 10 & -5 & 0 & 5 \end{pmatrix} \times \begin{pmatrix} 0 & 6 \\ 3 & 2 \\ 0 & 5 \\ 1 & 4 \end{pmatrix}$$

4. Варіант

$$\begin{pmatrix} 1 & 1 & 2 & 9 \\ -7 & 0 & 4 & 0 \\ 6 & 8 & 0 & -5 \end{pmatrix} \times \begin{pmatrix} 1 & 4 \\ 0 & 2 \\ 3 & 6 \\ 5 & 0 \end{pmatrix}$$

5. Варіант

$$\begin{pmatrix} 1 & 1 & 2 & 9 \\ -7 & 0 & 4 & 0 \\ 6 & 8 & 0 & -5 \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ 6 & 2 \\ 3 & 5 \\ 0 & 4 \end{pmatrix}$$

6. Варіант

$$\begin{pmatrix} 0 & 1 & 3 & -6 \\ 3 & 0 & 2 & 0 \\ 0 & 8 & -1 & -3 \end{pmatrix} \times \begin{pmatrix} 0 & 6 \\ 1 & 2 \\ 3 & 4 \\ 5 & 0 \end{pmatrix}$$

7. Варіант

$$\begin{pmatrix} 1 & 0 & -6 & 0 \\ 3 & 1 & 1 & -7 \\ 0 & 8 & -1 & 3 \end{pmatrix} \times \begin{pmatrix} 1 & 3 \\ 6 & 0 \\ 0 & 5 \\ 4 & 2 \end{pmatrix}$$

8. Варіант

$$\begin{pmatrix} -3 & 1 & 2 & 0 \\ 9 & 0 & -10 & -7 \\ 0 & 8 & -1 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 7 \\ 1 & 2 \\ 3 & 6 \\ 5 & 4 \end{pmatrix}$$

9. Варіант

$$\begin{pmatrix} 3 & -1 & 0 & -9 \\ 2 & 0 & 4 & 7 \\ 4 & 6 & -6 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 3 \\ 1 & 0 \\ 0 & 2 \\ 5 & 4 \end{pmatrix}$$

10. Варіант

$$\begin{pmatrix} 11 & -3 & 0 & 5 \\ -3 & 4 & -4 & -2 \\ 0 & -3 & -1 & 9 \end{pmatrix} \times \begin{pmatrix} 0 & 5 \\ 1 & 2 \\ 3 & 6 \\ 0 & 4 \end{pmatrix}$$

11. Варіант

$$\begin{pmatrix} 0 & 2 & 0 & -9 \\ -2 & -4 & -6 & 5 \\ 10 & 0 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 7 \\ 8 & 2 \\ 3 & 6 \\ 5 & 4 \end{pmatrix}$$

12. Варіант

$$\begin{pmatrix} 0 & 1 & 0 & 3 \\ -2 & 0 & 4 & 0 \\ 0 & 7 & 0 & -5 \end{pmatrix} \times \begin{pmatrix} 8 & 6 \\ 3 & 2 \\ 1 & 4 \\ 5 & 7 \end{pmatrix}$$

13. Варіант

$$\begin{pmatrix} 1 & 0 & 2 & 6 \\ 0 & 6 & 0 & 7 \\ -3 & 8 & -4 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 6 \\ 0 & 2 \\ 3 & 0 \\ 5 & 4 \end{pmatrix}$$

14. Варіант

$$\begin{pmatrix} 3 & 1 & 3 & 0 \\ 0 & 2 & 0 & 4 \\ -4 & 0 & -1 & -5 \end{pmatrix} \times \begin{pmatrix} 0 & 4 \\ 1 & 0 \\ 0 & 2 \\ 3 & 0 \end{pmatrix}$$

15. Варіант

$$\begin{pmatrix} 2 & 1 & -6 & 0 \\ 3 & 0 & -1 & 5 \\ 0 & 8 & -2 & 9 \end{pmatrix} \times \begin{pmatrix} 0 & 8 \\ 2 & 7 \\ 3 & 0 \\ 4 & 5 \end{pmatrix}$$

16. Варіант

$$\begin{pmatrix} -3 & 2 & 3 & 0 \\ 1 & 0 & 5 & -7 \\ 0 & -4 & -6 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 4 \\ 1 & 4 \\ 3 & 2 \\ 3 & 2 \end{pmatrix}$$

17. Варіант

$$\begin{pmatrix} 3 & 1 & 0 & 9 \\ -2 & 0 & 4 & 0 \\ 0 & 8 & -1 & 3 \end{pmatrix} \times \begin{pmatrix} 0 & 1 \\ 1 & 2 \\ 2 & 3 \\ 3 & 4 \end{pmatrix}$$

18. Варіант

$$\begin{pmatrix} 0 & -2 & 7 & 5 \\ -1 & 4 & -5 & -6 \\ 2 & -3 & -1 & 10 \end{pmatrix} \times \begin{pmatrix} 0 & 1 \\ 1 & 2 \\ 2 & 3 \\ 3 & 4 \end{pmatrix}$$

19. Варіант

$$\begin{pmatrix} 1 & 3 & 0 & 9 \\ 0 & 0 & -7 & 0 \\ 10 & -4 & 0 & 5 \end{pmatrix} \times \begin{pmatrix} 4 & 3 \\ 3 & 2 \\ 2 & 1 \\ 1 & 0 \end{pmatrix}$$

20. Варіант

$$\begin{pmatrix} 0 & 1 & -2 & -9 \\ -3 & 0 & 8 & 0 \\ 2 & 8 & 0 & -5 \end{pmatrix} \times \begin{pmatrix} 1 & 3 \\ 2 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$$

21. Варіант

$$\begin{pmatrix} 1 & 1 & 2 & 3 \\ 3 & 1 & 3 & 4 \\ 2 & 3 & 1 & 2 \end{pmatrix} \times \begin{pmatrix} 2 & 3 \\ 1 & 2 \\ 3 & 6 \\ 5 & 4 \end{pmatrix}$$

22. Варіант

$$\begin{pmatrix} 3 & 1 & -3 & -2 \\ 4 & 0 & 2 & 0 \\ 5 & 4 & -5 & -3 \end{pmatrix} \times \begin{pmatrix} 1 & 2 \\ 1 & 2 \\ 2 & 2 \\ 3 & 3 \end{pmatrix}$$

23. Варіант

$$\begin{pmatrix} -1 & 2 & -2 & 3 \\ 1 & -1 & 2 & -3 \\ -2 & 1 & -2 & 3 \end{pmatrix} \times \begin{pmatrix} 3 & 3 \\ 3 & 1 \\ 1 & 1 \\ 2 & 2 \end{pmatrix}$$

24. Варіант

$$\begin{pmatrix} 1 & 2 & 0 & 4 \\ 0 & 2 & 3 & 0 \\ 1 & 0 & 3 & 4 \end{pmatrix} \times \begin{pmatrix} 3 & 4 \\ 1 & 2 \\ 1 & 2 \\ 3 & 4 \end{pmatrix}$$

Лабораторна робота №7. Паралельне розв'язування систем лінійних рівнянь. Метод Якобі

Мета: ознайомитися з паралельним алгоритмом розв'язування систем лінійних рівнянь.

План

1. Розв'язування системи лінійних рівнянь методом Якобі.
2. Написання програми для розв'язування системи методом Якобі з використанням технології паралельного програмування MPI.

І. Теоретичні відомості

Розв'язування систем лінійних рівнянь. Метод Якобі

Розглянемо систему *n* лінійних рівнянь з невідомими $x_1, x_2, ..., x_n$:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n = b_2, \\ \qquad\qquad ... \\ a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n = b_n, \end{cases},$$

де коефіцієнти $a_{ij}(i, j = 1,...,n)$ -- дійсні числа.

Запишемо систему в матричній формі:

$Ax = b$,

де A – матриця розмірності $n \times n$, $x = (x_1,...,x_n)$ -- шуканий вектор, $b = (b_1,...,b_n)$ -- заданий вектор.

Розв'язування систем лінійних рівнянь є однією з важливих обчислювальних задач, що часто зустрічаються у прикладній математиці. До розв'язування систем лінійних рівнянь зводиться ряд задач аналізу, пов'язаних з наближенням функції, розв'язуванням диференціальних рівнянь і т. д. Вельми поширеними методами розв'язку систем лінійних рівнянь є ітераційні методи, які полягають у тому, що розв'язок *x* системи знаходиться як границя при $k \to \infty$ послідовних наближень x^k, де $k = 0,1,...$ -- номер ітерації.

Ми розглянемо найпростіший з таких алгоритмів – *метод Якобі*. Згідно йому, послідовні наближення, починаючи з деякого початкового x^0, обчислюються з використанням наступної формули:

$$x_i^{k+1} = \frac{1}{a_{ii}}\left(-\sum_{j \neq i} a_{ij}x_j^k + b_i\right),\ i = 1,...,n.$$

Метод Якобі не є прийнятним для більшості задач, оскільки він збігається не завжди, а тільки у випадку діагональної переваги матриці А.

Якість отриманого приближення x^k характеризує норма вектора $r^k = x^k - x^{k-1}$ -- так звана нев'язки. Зазвичай обчислення у процесі продовжуються до тих пір, поки норма нев'язки не стане достатньо малою:

$\left\|x^k - x^{k-1}\right\| \leq \varepsilon$, де $\varepsilon > 0$ є деяке наперед задане мале число.

Приклад

$$\begin{cases} 4{,}1x_1 + 2{,}5x_2 + 1{,}3x_3 = 2{,}1; \\ 3{,}5x_1 - 7{,}7x_2 + 2{,}8x_3 = 1{,}7; \\ 3{,}1x_1 + 1{,}2x_2 + 5{,}8x_3 = 0{,}8. \end{cases}$$

Розв'язання

$$\begin{cases} 4{,}1x_1 + 2{,}5x_2 + 1{,}3x_3 = 2{,}1; \\ 3{,}5x_1 - 7{,}7x_2 + 2{,}8x_3 = 1{,}7; \\ 3{,}1x_1 + 1{,}2x_2 + 5{,}8x_3 = 0{,}8. \end{cases}$$

Перевіряємо три умови:

1. $\begin{vmatrix} 4{,}1 & 2{,}5 & 1{,}3 \\ 3{,}5 & -7{,}7 & 2{,}8 \\ 3{,}1 & 1{,}2 & 5{,}8 \end{vmatrix} = -189{,}44$;

2. $a_{11} = 4{,}1 \neq 0$

   $a_{22} = -7{,}7 \neq 0$,

   $a_{33} = 5{,}8 \neq 0$

3. Діагональна перевага:

$$|a_{11}| = |4{,}1| > |2{,}5| + |1{,}3| = 3{,}8;$$
$$|a_{22}| = |-7{,}7| > |3{,}5| + |2{,}8| = 6{,}3;$$
$$|a_{33}| = |5{,}8| > |3{,}1| + |1{,}2| = 4{,}3.$$

$$\begin{cases} x_1 = \dfrac{1}{4{,}1}(2{,}1 - 2{,}5x_2 - 1{,}3x_3), \\ x_2 = \dfrac{1}{-7{,}7}(1{,}7 - 3{,}5x_1 - 2{,}8x_3), \\ x_3 = \dfrac{1}{5{,}8}(0{,}8 - 3{,}1x_1 - 1{,}2x_2). \end{cases}$$

4,1	2,5	1,3	2,1
3,5	-7,7	2,8	1,7
3,1	1,2	5,8	0,8

x1	1	0,609756	0,317073	0,512195		
x2	-0,45455	1	-0,36364	-0,22078		
x3	0,534483	0,206897	1	0,137931	ε=	0,001

Процес ітерації:

	x1	x2	x3	ΔX1	ΔX2	ΔX3	
	0	0	0				max
1	0,512195	-0,22078	0,137931	0,5122	0,2208	0,1379	0,5122
2	0,603082	0,062193	-0,09015	0,0909	0,2830	0,2281	0,2830
3	0,502856	0,020567	-0,19727	0,1002	0,0416	0,1071	0,1071
4	0,562204	-0,06394	-0,13509	0,0593	0,0845	0,0622	0,0845
5	0,59402	-0,01436	-0,14933	0,0318	0,0496	0,0142	0,0496
6	0,568297	-0,00507	-0,17659	0,0257	0,0093	0,0273	0,0273

7	0,57128	-0,02668	-0,16476	0,0030	0,0216	0,0118	0,0216
8	0,580704	-0,02102	-0,16189	0,0094	0,0057	0,0029	0,0094
9	0,576343	-0,01569	-0,1681	0,0044	0,0053	0,0062	0,0062
10	0,575062	-0,01993	-0,16687	0,0013	0,0042	0,0012	0,0042
11	0,577258	-0,02007	-0,16531	0,0022	0,0001	0,0016	0,0022
12	0,576845	-0,0185	-0,16645	0,0004	0,0016	0,0011	0,0016
13	0,576253	-0,0191	-0,16655	0,0006	0,0006	0,0001	0,0006

$$X = \begin{pmatrix} 0{,}576253 \\ -0{,}0191 \\ -0{,}16655 \end{pmatrix}.$$

Відповідь. $X = \begin{pmatrix} 0{,}576253 \\ -0{,}0191 \\ -0{,}16655 \end{pmatrix}$

Паралельне розв'язання системи лінійних рівнянь

7) Визначаємо кількість процесів.

8) На головному процесі:

a. ділимо кожен рядок матриці A на діагональний елемент і отримаємо матрицю A':

$$A = \begin{pmatrix} 4{,}1 & 2{,}5 & 1{,}3 \\ 3{,}5 & -7{,}7 & 2{,}8 \\ 3{,}1 & 1{,}2 & 5{,}8 \end{pmatrix},\ A' = \begin{pmatrix} 1 & \frac{2{,}5}{4{,}1} & \frac{1{,}3}{4{,}1} \\ \frac{3{,}5}{-7{,}7} & 1 & \frac{2{,}8}{-7{,}7} \\ \frac{3{,}1}{5{,}8} & \frac{1{,}2}{5{,}8} & 1 \end{pmatrix};$$

b. У матриці A' 1 заміняємо на нулі а решту елементів замінюємо на протилежні:

$$A'' = \begin{pmatrix} 0 & -\frac{2{,}5}{4{,}1} & -\frac{1{,}3}{4{,}1} \\ -\frac{3{,}5}{-7{,}7} & 0 & -\frac{2{,}8}{-7{,}7} \\ -\frac{3{,}1}{5{,}8} & -\frac{1{,}2}{5{,}8} & 0 \end{pmatrix}.$$

c. ділимо відповідний елемент вектора B на діагональний елемент матриці A і отримаємо вектор B':

$$B = \begin{pmatrix} 2,1 \\ 1,7 \\ 0,8 \end{pmatrix},\ B' = \begin{pmatrix} \frac{2,1}{4,1} \\ \frac{1,7}{-7,7} \\ \frac{0,8}{5,8} \end{pmatrix}.$$

9) Далі для розв'язку задачі на паралельній системі матрицю A'' "розрізають" на p горизонтальних смуг, де p – кількість процесів у системі. Аналогічно, горизонтальними смугами, розділяють вектор B' і вектор поточного наближення $X(0,0,0)$. Смуги розподіляють по відповідним комп'ютерам і реалізують паралельний алгоритм множення матриці A'' на вектор X. До результату додаємо вектор B'.

10) $k = 0$,

a. $$\begin{pmatrix} 0 & -\frac{2,5}{4,1} & -\frac{1,3}{4,1} \\ -\frac{3,5}{-7,7} & 0 & -\frac{2,8}{-7,7} \\ -\frac{3,1}{5,8} & -\frac{1,2}{5,8} & 0 \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} \frac{2,1}{4,1} \\ \frac{1,7}{-7,7} \\ \frac{0,8}{5,8} \end{pmatrix} = \begin{pmatrix} 0,5122 \\ -0,2207 \\ 0,1379 \end{pmatrix} = \begin{pmatrix} x_1^1 \\ x_2^1 \\ x_3^1 \end{pmatrix},$$

b. Перевіряємо значення норми невязки:

c. $$\max(|0,5122-0|;|-0,2207-0|;|0,1379-0|) = 0,5122 > \varepsilon = 0,001;$$

11) $k = 1$,

a. $$\begin{pmatrix} 0 & -\frac{2,5}{4,1} & -\frac{1,3}{4,1} \\ -\frac{3,5}{-7,7} & 0 & -\frac{2,8}{-7,7} \\ -\frac{3,1}{5,8} & -\frac{1,2}{5,8} & 0 \end{pmatrix} \times \begin{pmatrix} 0,5122 \\ -0,2207 \\ 0,1379 \end{pmatrix} + \begin{pmatrix} \frac{2,1}{4,1} \\ \frac{1,7}{-7,7} \\ \frac{0,8}{5,8} \end{pmatrix} = \begin{pmatrix} 0,6030 \\ 0,0622 \\ -0,0902 \end{pmatrix} = \begin{pmatrix} x_1^2 \\ x_2^2 \\ x_3^2 \end{pmatrix},$$

b. Перевіряємо значення норми нев'язки:

c. $$\max(|0,6030-0,5122|;|0,0622-(-0,2207)|;|-0,0902-0,1379|) =$$
$$= 0,2830 > \varepsilon = 0,001,$$

$k = 2$ і т. д. доти, доки нев'язки не стане достатньо малою: $\|x^k - x^{k-1}\| \le \varepsilon$.

II. Практична частина

Інструкції до виконання

1. Розв'язати систему методом Якобі.

2. Написати програму для розв’язання системи методом Якобі, використовуючи технологію паралельного програмування MPI.

Варіанти згідно *Додатка №1.*

Поточні контрольні питання:

1. Опишіть метод Якобі для розв’язування системи лінійних рівнянь.
2. Опишіть паралельний алгоритм для розв’язування системи лінійних рівнянь.

Рекомендована література:

1. Паралельні та розподілені обчислення [Текст] : навч. підруч. для студентів вищ. навч. закл. / А. Луцків, С. Луценко, В. Пасічник. – Львів : Магнолія 2006, 2017. – 565, [1] с. : схеми.

2. Аксак Н. Г. Паралельні та розподілені обчислення: підруч. / Н. Г. Аксак, О. Г. Руденко, А.М. Гуржій. – Х.: Компанія СМІТ, 2009. – 480 с.

3. Паралельні та розподілені обчислення: навчальний посібник для вищих закладів освіти / К.Т. Кузьма, О.В. Мельник. – Миколаїв: ФОП Швець В.М., 2020. – 172 с.

4. Шеховцов В. А. Операційні системи / Шеховцов В. А. – Київ : Видавнича група BHV, 2005. – 576 с.

5. Кулаков А.Ю., Клименко І.А. Спосіб формування структури віртуальної GRID системи // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – К.: Век+, 2009. – № 50. - С. 97-100.

6. Семеренко В. П. Технології паралельних обчислень : навчальний посібник / Семеренко В. П. – Вінниця : ВНТУ, 2018. – 104 с.

7. Згуровський М.З., Петренко А.І. Е-наука на шляху до семантичного Грід. Частина 2: Семантичний Web- і семантичний Грід // Системні дослідження та інформаційні технології. - Київ, №2, 2010. - С. 7-25.

8. Петренко А. І. Практикум з грід-технологій : навчальний посібник / А. І. Петренко, С. Я. Свістунов, Г. Д. Кисельов. – К. : НТУУ «КПІ», 2011. – 580 с.

9. Згуровський М.З., Петренко А.І. Е-наука на шляху до семантичного Грід. Частина 1: Об’єднання Web- і Грід- технологій // Системні дослідження та інформаційні технології. – Київ, №1, 2010. – С. 26-38.

Додаток №1

1. Варіант
$$\begin{cases} 9x_1+5x_2+x_3=1, \\ 2x_1-7x_2-4x_3=-2, \\ 3x_1-x_2+8x_3=3. \end{cases}$$
2. Варіант
$$\begin{cases} 5x_1-3x_2+1x_3=1, \\ -3x_1+8x_2-4x_3=-2, \\ 2x_1-1x_2-5x_3=1. \end{cases}$$
3. Варіант
$$\begin{cases} 9x_1+2x_2+3x_3=1, \\ -2x_1+6x_2-4x_3=3, \\ 3x_1-5x_2+10x_3=2. \end{cases}$$
4. Варіант
$$\begin{cases} 4x_1+x_2+2x_3=2, \\ -x_1+6x_2+4x_3=-3, \\ 6x_1+x_2+8x_3=1. \end{cases}$$
5. Варіант
$$\begin{cases} 5x_1+x_2+2x_3=2, \\ -2x_1+7x_2+4x_3=-3, \\ 6x_1+2x_2+9x_3=1. \end{cases}$$
6. Варіант
$$\begin{cases} -6x_1+3x_2-x_3=4, \\ 3x_1+6x_2+2x_3=0, \\ 3x_1-x_2-8x_3=-3. \end{cases}$$
7. Варіант
$$\begin{cases} 6x_1+4x_2-1x_3=1, \\ x_1+3x_2+x_3=2, \\ 3x_1-x_2+8x_3=-2. \end{cases}$$
8. Варіант
$$\begin{cases} -8x_1+x_2+3x_3=-1, \\ x_1+10x_2-8x_3=-2, \\ x_1-x_2+8x_3=-3. \end{cases}$$
9. Варіант
$$\begin{cases} 9x_1-x_2-3x_3=3, \\ 2x_1+7x_2+4x_3=0, \\ x_1+3x_2-6x_3=3. \end{cases}$$
10. Варіант
$$\begin{cases} 11x_1-3x_2+5x_3=1, \\ -3x_1+8x_2-4x_3=-2, \\ -3x_1-x_2+9x_3=1. \end{cases}$$
11. Варіант
$$\begin{cases} 9x_1+2x_2-2x_3=7, \\ -2x_1-7x_2-4x_3=3, \\ 2x_1+5x_2+10x_3=1. \end{cases}$$
12. Варіант
$$\begin{cases} 3x_1+x_2+x_3=-2, \\ -2x_1+4x_2+x_3=-3, \\ 2x_1+3x_2-7x_3=-1. \end{cases}$$
13. Варіант
$$\begin{cases} 6x_1+2x_2+1x_3=4, \\ 2x_1+7x_2+3x_3=-5, \\ -3x_1+4x_2-8x_3=1. \end{cases}$$
14. Варіант
$$\begin{cases} 5x_1+x_2+3x_3=3, \\ 2x_1+7x_2+4x_3=2, \\ -4x_1-x_2-6x_3=-7. \end{cases}$$
15. Варіант
$$\begin{cases} 6x_1+x_2-2x_3=8, \\ 3x_1-5x_2+x_3=2, \\ 6x_1-2x_2+9x_3=-2. \end{cases}$$
16. Варіант
$$\begin{cases} -4x_1+2x_2+x_3=-1, \\ x_1+7x_2-5x_3=2, \\ -4x_1-6x_2+11x_3=-3. \end{cases}$$
17. Варіант
$$\begin{cases} 9x_1+x_2+3x_3=-1, \\ -2x_1+8x_2+4x_3=-2, \\ 3x_1-x_2+8x_3=-3. \end{cases}$$
18. Варіант
$$\begin{cases} -7x_1+x_2+5x_3=3, \\ -x_1+5x_2-3x_3=4, \\ 2x_1-3x_2-6x_3=-5. \end{cases}$$
19. Варіант
$$\begin{cases} 9x_1+3x_2+x_3=4, \\ 2x_1-9x_2-6x_3=5, \\ 5x_1-4x_2+10x_3=1. \end{cases}$$
20. Варіант
$$\begin{cases} 9_1-2x_2-5x_3=1, \\ -3x_1+8x_2+2x_3=-2, \\ 2x_1+5x_2-8x_3=3. \end{cases}$$
21. Варіант
$$\begin{cases} 6x_1+x_2+2x_3=4, \\ 3x_1+7x_2+3x_3=-5, \\ 2x_1+3x_2+6x_3=6. \end{cases}$$
22. Варіант
$$\begin{cases} 5x_1+x_2-3x_3=5, \\ 4x_1+7x_2+2x_3=1, \\ 5x_1+4x_2-10x_3=2. \end{cases}$$
23. Варіант
$$\begin{cases} -5x_1+2x_2-2x_3=-4, \\ x_1-4x_2+2x_3=5, \\ -2x_1+x_2-4x_3=-6. \end{cases}$$
24. Варіант
$$\begin{cases} 4x_1+2x_2+x_3=1, \\ 2x_1+7x_2+4x_3=2, \\ x_1+3x_2+5x_3=3. \end{cases}$$