МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДВНЗ «ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»
КАФЕДРА ПРИКЛАДНОЇ МАТЕМАТИКИ ТА ІНФОРМАТИКИ

МЕТОДИЧНІ ВКАЗІВКИ І ЗАВДАННЯ

до виконання практичних робіт з дисципліни

«Теорія та організація розподіленої обробки даних»

для студентів освітнього ступеня «Магістр»

спеціальності 121 Інженерія програмного забезпечення

всіх форм навчання

Дрогобич – 2025

УДК 004.4(072)

М 54

Методичні вказівки і завдання до виконання практичних робіт з дисципліни «Теорія та організація розподіленої обробки даних» для студентів освітнього ступеня «Магістр» спеціальності 121 Інженерія програмного забезпечення всіх форм навчання / уклад. І.А. Назарова. - Дрогобич: ДВНЗ «ДонНТУ», 2025. - 39 с.

Укладачі: Ірина НАЗАРОВА, к.т.н., доцент кафедри ПМІ

Рецензент: Віталій ШАМАЄВ, к.т.н., доцент кафедри ЕТіКІ

Відповідальний за випуск: Ярослав ДОРОГИЙ, д.т.н., в. о. завідувача кафедри прикладної математики та інформатики

Затверджено навчально-методичним відділом ДВНЗ ДонНТУ, протокол № 5 від «25» березня 2025р.

Розглянуто на засіданні кафедри ПМІ, протокол №3 від 21 лютого 2025р.

ЗМІСТ

ВСТУП

Методичні вказівки призначені до виконання практичних робіт за курсом «Теорія та організація розподіленої обробки даних» для магістрів, що навчаються за спеціальністю 121 Інженерія програмного забезпечення.

Основною метою виконання практичних робіт є отримання теоретичних знань та практичних навичок у розробці та аналізу ефективності паралельних алгоритмів та їх програмних додатків.

Розробка алгоритмічного та програмного забезпечення для обчислювальних систем з розподіленою пам’яттю – це актуальна і нетривіальна задача в галузі комп’ютерних наук. Особлива складність виконання таких завдань пов’язана з малою кількістю друкованих джерел та інтернет видань, що викладені українською мовою. З другого боку, паралельні обчислення для розподіленої пам’яті характерні для кластерних систем, тому є актуальними для студентів усіх спеціальностей галузі знань 12 Інформаційні технології, і, насамперед, для спеціальності 121 Інженерія програмного забезпечення.

За структурою методичні вказівки містять тему, завдання, індивідуальні варіанти (додаток А) та рекомендації для обрання варіанту (додаток Б), а також приклад виконання практичних робіт за одним з варіантів.

Таким чином, методичні вказівки містять у повному обсязі всю необхідну інформацію для ефективної організації роботи магістрів в процесі виконання практичних робіт за курсом «Теорія та організація розподіленої обробки даних» і не дублюють відповідний лекційний курс.

РОЗДІЛ 1
ВКАЗІВКИ ТА ЗАВДАННЯ ДО ВИКОНАННЯ ПРАКТИЧНИХ РОБІТ

Практична робота №1
Розробка та оцінка ефективності послідовних алгоритмів розв’язання обчислювального завдання

Мета практичної роботи: набуття та закріплення теоретичних знань та практичних навичок в розробці та аналізі ефективності послідовних алгоритмів розв’язання реальних обчислювальних завдань.

Завдання до практичної роботи

Для обраного індивідуального варіанта (додатки А, Б) реального обчислювального завдання:

- описати постановку індивідуального завдання розв'язуваної задачі в змістовному та математичному варіантах;
- провести аналітичний огляд існуючих методів розв’язання індивідуального завдання, обґрунтувати вибір методу розв'язання;
- розробити послідовний алгоритм обраного методу розв’язання завдання;
- розробити систему тестів для перевірки та обґрунтування коректності роботи послідовного алгоритму при розв’язанні завдання для довільних вхідних даних та постановок завдання;
- визначити та обґрунтувати розмір (розмірність вхідних даних) входу для обраного завдання, m;

- визначити та обґрунтувати алгоритм отримання значення оцінки часу виконання однієї операції з плаваючою точкою (флоп), t_{op} або таблицю оцінок різних операцій з плаваючою точкою окремо (довгих та швидких…);

- вивести аналітичні оцінки ефективності або якості послідовного алгоритму, тимчасову: $T_1(m, t_{op})$ та ємнісну складність (за пам'яттю): $S_1(m)$;

- оцінити клас асимптотичної складності послідовного алгоритму за часом, провести дослідження залежності аналітичних оцінок ефективності від розміру входу.

Зміст звіту для практичної роботи

1. Титульний лист для практичної роботи (назва, студент, дата тощо).

2. Змістовний та математичний опис варіанта індивідуального завдання.

3. Огляд найбільш відомих методів розв'язання індивідуального завдання (область застосування, недоліки та переваги, обмеження на вхідні дані, якщо такі існують, оцінка ефективності), обґрунтування обраного методу розв'язання.

4. Опис обраного послідовного алгоритму розв'язання індивідуального завдання (наприклад, блок-схема). Якщо алгоритм є достатньо складним, доцільно розбити його на частки та навести опис кожного підалгоритму окремо і описати їх зв'язок між собою.

5. Навести обґрунтування, що є розміром вхідних даних для даного послідовного алгоритму, m.

6. Довести коректність роботи послідовного алгоритму на розробленій системі тестів (покрокове виконання для кожного тесту).

7. Навести алгоритм та програмний додаток експерименту для отримання тимчасових оцінок однієї/різних операцій, що використовуються в послідовному алгоритмі.

8. Навести аналітичні вирази тимчасової та ємнісної складності та покроковий опис їх отримання. Провести аналіз залежності аналітичних оцінок ефективності від розміру входу для заданого завдання, побудувати графіки аналітичної залежності $T_1(m), S_1(m)$. За тимчасовою складністю визначити клас, до якого належить описаний алгоритм (лінійний $O(m)$, поліноміальний $O(m^c)$, експоненційний $O(c^m)$, тощо).

9. Висновки.

10. Список використаних джерел або перелік посилань.

Зауваження: позначення в математичній постановці та в описі алгоритму повинні співпадати, для відповідних змінних в ПД або потрібно навести таблицю позначень, або ідентифікатори змінних повинні бути очевидно-відповідними до своїх математичних/алгоритмічних аналогів.

Контрольні питання

1. Як оцінити складність послідовного алгоритму?
2. Що таке розмірність вхідних даних (вхідна довжина) або розмір входу? Навести приклади для різних класів алгоритмів.
3. Асимптотична складність послідовного алгоритму.
4. Тимчасова або часова складність алгоритму.
5. Ємнісна або просторова складність алгоритму.
6. Класифікація алгоритмів за часовою складністю.
7. Лінійні, поліноміальні та експоненційні алгоритми.
8. Поняття ефективного алгоритму.
9. NP-повні завдання. Приклади.

Практична робота №2
Реалізація програмних додатків послідовного розв'язання обчислювального завдання

Мета практичної роботи: набуття та закріплення теоретичних знань та практичних навичок в розробці та аналізі ефективності програмних додатків для послідовних алгоритмів розв'язання обчислювальних завдань.

Завдання до практичної роботи

Для обраного індивідуального варіанта (додаток А) реального обчислювального завдання:

- розробити програмний додаток (ПД) послідовного алгоритму розв'язання обчислювального завдання (довільна мова програмування/середовище), отриманого в практичній роботі 1;
- обґрунтувати обрання мови програмування або середовища для розв'язання конкретного обраного обчислювального завдання;
- розробити і описати повну систему тестів для перевірки функціонування програмного додатку (інтерфейс, вхідні/вихідні дані тощо);
- навести та проаналізувати програмні результати тестування коректності роботи послідовного алгоритму для розробленого ПД при довільних постановках завдання (система тестів з практичної роботи 1);
- провести експериментальні дослідження тимчасової складності послідовного алгоритму для різних обсягів вхідних даних;
- проаналізувати та порівняти теоретичні вирази та експериментальні результати оцінки тимчасової складності послідовного алгоритму та програмного додатку.

Зміст звіту для практичної роботи

1. Титульний лист для практичної роботи (назва, студент, дата, тощо).

2. Опис програмного додатку повинен бути виконаний за всіма вимогами програмної інженерії. Вхідні та вихідні структури даних та обґрунтування їх вибору, особливо для завдань з теорії графів. Основні функції. Структурна схема та екранні форми основних функцій програмної системи тощо. Текст програми основних модулів ПЗ з достатньою кількістю коментарів може бути наведений в додатку.

3. Опис системи тестових завдань, обґрунтування її повноти. Приклади роботи ПД за кожним тестом.

4. Результати експериментальних досліджень для часової складності ПД та їх порівняння з теоретичним аналогом. Графіки залежності аналітичної та експериментальної тимчасової складності від розміру входу.

5. Скриншоти роботи ПД.

6. Висновки.

7. Лістинги програмного додатку.

8. Перелік посилань.

Зауваження: позначення в математичній постановці/описі алгоритму і ПД повинні або співпадати, або для відповідних змінних в ПД потрібно навести таблицю позначень, або ідентифікатори змінних повинні бути очевидно-відповідними до своїх математичних/алгоритмічних аналогів.

Контрольні питання

1. Класифікація мов програмування. Послідовні мови програмування високого та низького рівнів та області їх застосування.

2. Машинні та машинно-орієнтовані мови. Асемблери. Мови символьного кодування. Функціональне та логічне програмування.

3. Машино-незалежні мови. Проблемно-, процедурно-орієнтовані та декларативні мови. Обробка даних. Мови моделювання.

4. Мови для розв’язування обчислювальних задач.
5. Тестування програмних додатків. Основні методи.

Практична робота №3
Розробка та оцінка ефективності паралельних алгоритмів розв'язання обчислювального завдання

Мета практичної роботи: набуття та закріплення теоретичних знань та практичних навичок в розробці та аналізі ефективності паралельних алгоритмів розв'язання реальних обчислювальних завдань.

Завдання до практичної роботи

Для обраного індивідуального варіанта (додаток А) реального обчислювального завдання:

1) розробити паралельний алгоритм (ПА) розв'язання індивідуального завдання відповідно до ієрархічної декомпозиційної методики, викладеної в навчальних матеріалах курсу «Паралельні інформаційні системи», в умовах ідеї необмеженого паралелізму;

2) розробити реальне відображення отриманого паралельного алгоритму на одну з заданих топологій: кільце або 1D-тор, сітка або 2D- тор та гіперкуб;

3) провести теоретичний аналіз ефективності розробленого/розроблених паралельних алгоритмів (динамічні характеристики: час паралельних обчислень та обмінів, прискорення та ефективність, накладні затрати на паралелізм, ступінь паралелізму);

4) оцінити масштабованість паралельного алгоритму, побудувати та проаналізувати функцію ізоефективності, визначити пріоритетні області використання розробленого паралельного алгоритму.

Зміст звіту для практичної роботи

1. Титульний лист для практичної роботи (назва, студент, дата тощо).

2. Опис паралельного алгоритму розв’язання індивідуального завдання відповідно до ієрархічної декомпозиційної методики із застосуванням графів впливу в умовах ідеї необмеженого паралелізму (довільна кількість паралельних пристроїв/процесорів та повнозв’язна комунікаційна топологія):

- початковий та поточний розподіл даних між процесорами;
- перелік та опис макрооперацій, якщо такі будуть мати місце;
- граф впливу всього алгоритму або кожної з макрооперацій разом зі схемою їх з’єднання.

3. Обґрунтування і схема відображення паралельного алгоритму на топологію з'єднання обчислювальних пристроїв. Схема відображення ПА виконується на таку з заданих топологій, на яку конкретно даний обчислювальний алгоритм відображається природним чином. Відображення на інші топології виконується за рахунок логічного перетворення однієї топології в іншу (наприклад, за рахунок кодів Грея), наводиться тільки таблиця відповідності номерів процесорів для застосованих топологій.

4. Навести аналітичні вирази для динамічних характеристик паралельних алгоритмів, такі як:

1) $T_{p,comp}$ − час виконання розрахунків в ПА;
2) $T_{p,comm}$ − час на виконання міжпроцесорного обміну в ПА;
3) T_p − загальний час на реалізацію ПА: $T_p = T_{p,comp} + T_{p,comm}$;
4) Dop −максимальний ступінь паралелізму ПА;
5) $S_p,\ E_p$ − коефіцієнти прискорення та ефективності ПА.

Аналітичні вирази динамічних характеристик ПА обов’язково будуть функціями від розміру входу, m та машино-залежних констант: t_{op} та t_s − латентність комунікаційної мережі та t_w − час на передачу одного слова в мережі. Окрім цього, в якості параметрів в виразах можуть бути характеристик обчислювального завдання та методу його розв’язання.

Зауваження: для отримання аналітичних виразів для оцінки часу комунікацій рекомендується використовувати лінійну модель Хокні.

6. Оцінити масштабованість ПА на базі ізоефективного аналізу. Навести аналітичний вираз для накладних витрати на паралелізм, T_0 та для функції ізоефективності f_E. Класифікувати масштабованість отриманого ПА. Побудувати аналітичні вирази та графіки областей пріоритетного використання отриманого ПА.

7. Провести теоретичний аналіз якості отриманого ПА на основі динамічних характеристик. Побудувати графіки аналітичної залежності динамічних характеристик та характеристик масштабованості від розміру входу, кількості процесорів обчислювальної системи, параметрів методу та машино-залежних констант. Зробити висновки.

8. Висновки.

9. Перелік посилань.

Контрольні питання

1. Паралельні обчислювальні системи (ОС) з розподіленою пам’яттю. Топології з’єднання паралельних обчислювальних пристроїв в паралельну розподілену систему.

2. Моделі передачі даних в розподілених ОС. Лінійна модель Хокні.

3. Методи отримання паралельних алгоритмів. Ієрархічна декомпозиційна методика розробки ПА. Агрегаційний метод розробки ПА.

4. Графи алгоритмів та особливості їх застосування. Графи впливу.

5. Задача відображення ПА на реальну комунікаційну мережу. Коди Грея. Задача балансування навантаження паралельного обладнання.

6. Динамічні характеристики ПА. Прискорення. Ефективність. Максимальний ступінь паралелізму. Машинно-залежні константи.

7. Масштабованість паралельних алгоритмів. Накладні витрати на паралелізм. Засади ізоефективного аналізу. Функція ізоефективності.

Практична робота №4
Реалізація програмних MPI-додатків з відображенням на довільну топологію паралельної обчислювальної системи

Мета практичної роботи: набуття та закріплення теоретичних знань та практичних навичок в розробці та аналізі ефективності паралельних програмних додатків для алгоритмів розв'язання реальних обчислювальних завдань на базі інтерфейсу передачі повідомлень.

Завдання до практичної роботи

Для обраного індивідуального варіанта (додаток А) реального обчислювального завдання:

- розробити програмний додаток паралельного алгоритму розв'язання обчислювального завдання (довільна мова програмування +інтерфейс передачі повідомлень, MPI), отриманого в практичній роботі 3;
- розробити і описати повну систему тестів для перевірки функціонування програмного паралельного MPI-додатку (інтерфейс, вхідні та вихідні дані тощо);
- навести та проаналізувати програмні результати тестування коректності роботи паралельного алгоритму для розробленого MPI-додатку при довільних постановках завдання (система тестів з практичної роботи 1), порівняти їх послідовних алгоритмом;
- провести експериментальні дослідження динамічних характеристик паралельного способу вирішення завдання для різних обсягів вхідних даних, кількості процесорів, швидких та повільних мереж з'єднання процесорів у ОС тощо;

- проаналізувати та порівняти теоретичні вирази та експериментальні результати для динамічних показників та характеристик масштабованості ПА.

Зміст звіту для практичної роботи

1. Титульний лист для практичної роботи (назва, студент, дата тощо).

2. Опис та лістинг програмного MPI-додатку повинен бути виконаний за всіма вимогами програмної інженерії. Особливості опису паралельного додатку – показати початкові та поточні розподіли вхідних та вихідних даних. Текст програми основних модулів MPI-додатку з достатньою кількістю коментарів на українській мові може бути наведений в додатку.

3. Опис системи тестових завдань, обґрунтування її повноти. Приклади роботи MPI-додатку за кожним тестом.

4. Результати експериментальних досліджень для динамічних характеристик та характеристик масштабованості ПА за MPI-додатком та їх порівняння з теоретичними аналогами. Графіки залежності динамічних характеристик від параметрів задачі, ОС та методу: для різних обсягів вхідних даних, кількості процесорів, різних топологій з’єднання, швидких та повільних мереж з’єднання процесорів у ОС тощо;

Зауваження. Для коректного визначення експериментальних значень динамічних характеристик ПА використовувати елементи бар’єрної синхронізації MPI та MPI-функції керування часом.

5. Скриншоти роботи MPI-додатку.

6. Висновки.

6. Лістинги MPI-додатку.

7. Перелік посилань.

Контрольні питання

1. Інтерфейс передачі повідомлень MPI.
2. Стандарти MPI та його реалізації: MPIH, MS MPI тощо.

3. Атрибути процесів в MPI.

4. Основні процедури та функції.

5. На які групи можна розділити операції взаємодії між 2 процесами? Які типи операцій передачі повідомлень існують? У чому суть кожної з них?

6. Які існують режими передачі повідомлень з блокуванням і без блокування? У чому їх відмінності?

7. Назвіть відомі вам функції передачі і прийому повідомлень.

8. Як визначити точне число елементів у прийнятому повідомленні?

9. Що таке параметри-джокери? Назвіть відомі вам.

10. Як (за допомогою якої функції, структури) можна отримати інформацію про очікуване повідомленні? Які саме дані про структуру можна отримати?

11. Які допоміжні функції і з якою метою використовують при асинхронній передачі даних?

12. Яким чином можна реалізувати передачу повідомлення від 1 процесу – всім іншим?

13. Що таке операція редукції? Як вона виконується? Які колективні операції можна застосовувати в редукції? Які функції за це відповідають?

14. Як реалізувати операцію взаємодії процесів «all-to-all»?

15. За допомогою яких функцій реалізується розподіл і збір даних?

16. Яка функція дозволяє реалізувати синхронізацію процесів при колективній розсилці? Як вона працює?

17. Які існують способи роботи з підмножинами процесів?

18. Що таке група? Які наперед визначені групи існують?

19. Що таке комунікатор? Які наперед визначені комунікатори ви знаєте? Які функції реалізують роботу з групами і комунікаторами?

20. Які типи конструкторів груп вам відомі?

21. Що відбувається при виклику деструктора групи/комунікатора?

Практична робота 5
Розробка програмного візуалізатора покрокового виконання паралельного обчислювального алгоритму

Мета практичної роботи: набуття практичних навичок в розробці програмних візуалізаторів паралельних алгоритмів розв'язання реальних обчислювальних завдань для розподілених обчислювальних систем.

Завдання до практичної роботи

Для обраного індивідуального варіанта (додаток А) реального обчислювального завдання: розробити програмний додаток-візуалізатор покрокової реалізації виконання паралельного алгоритма на архітектурі з розподіленою пам'яттю, для того щоб перевести складний процес виконання паралельного алгоритму в інтуїтивно зрозумілу візуальну форму.

Програмний візуалізатор має своєю метою демонстрацію різних варіативних гілок обраного алгоритму для різних вхідних даних при його паралельній реалізації та націлений на спрощення опануванням методами підрахунку оцінок динамічних тимчасових показників для різних варіантів вхідних даних, що впливають на логіку алгоритму.

Для досягнення мети потрібно:

- обрати та обґрунтувати вибір декількох прикладів вхідних даних, щоб продемонструвати всі різні гілки виконання алгоритму (для алгоритмів сортування − впорядкована/не впорядкована послідовність елементів тощо);
- ретельно підібрати співвідношення розміру входу, m та кількості процесорів, p, для наочності розрахунків без втрати загальності;
- обрати та обґрунтувати відповідну найпростішу для відображення даного алгоритму топологію з'єднання процесорів серед заданих (для

матричних операцій це може бути 2D-тор, для методів сортування – 1D-тор тощо);

- візуалізатор повинен не тільки показувати кожен крок виконання паралельного алгоритму, а ще й видавати часову складність цього етапу;

- ПД не повинен працювати на довільних вхідних даних, тільки на демонстраційних прикладах.

Зміст звіту для практичної роботи

1. Титульний лист для практичної роботи (назва, студент, дата тощо).

2. Математичний опис постановки та розрахунків демонстраційних прикладів розв'язання обраного завдання: опис вхідних/вихідних даних, демонстрація коректності виконання алгоритму над цими вхідними даними. Якщо демонстраційні приклади було описано в практичній роботі 1, то дати тільки посилання. Обґрунтування обраних прикладів.

2. Опис застосування паралельного алгоритму для описаних прикладів, починаючи з початкового розбиття даних за процесорами, поточні комунікації та розрахунки, і на кінець– розбиття вихідних даних.

3. Опис візуалізатора як програмного додатку з точки зору вимог програмної інженерії. Текст програми-візуалізатора для основних модулів ПЗ з достатньою кількістю коментарів може бути наведений в додатку.

4. Опис системи тестів ПД-візуалізатор та результати тестування.

5. Скриншоти покрокової роботи програми для кожного з демонстраційних прикладів.

6. Висновки.

7. Перелік посилань.

8. Лістинги програмного додатку

Контрольні питання

1. Що таке програма візуалізатор? Мета програмного візуалізатора складних динамічних процесів?
2. Основні задачі візуалізації паралельних процесів?

РОЗДІЛ 2
ЗРАЗКИ ВИКОНАННЯ ТА ОФОРМЛЕННЯ ПРАКТИЧНИХ РОБІТ

2.1 Постановка задачі та огляд методів сортування масивів

Сортування є однією з типових проблем обробки даних і зазвичай сприймається як завдання розміщення елементів невпорядкованого набору значень:

$$S = a_1, a_2, \ldots, a_n, \tag{2.1}$$

у порядку монотонного зростання чи спадання:

$$S \sim S' = (a'_1, a'_2, \ldots, a'_n) : a'_1 \leq a'_2 \leq \ldots \leq a'_n. \tag{2.2}$$

Обчислювальна трудомісткість процедури впорядкування є досить високою. Так, для ряду відомих простих методів (сортування бульбашкою, сортування включенням та ін.) кількість необхідних операцій визначається квадратичною залежністю від числа даних, що впорядковуються:

$$T = n^2. \tag{2.3}$$

Для ефективніших алгоритмів (сортування злиттям, сортування Шелла, швидке сортування) трудомісткість визначається величиною:

$$T = n \; log \; log \; n \; . \tag{2.4}$$

Даний вираз дає також нижню оцінку необхідної кількості операцій для впорядкування набору n значень; алгоритми з меншою трудомісткістю можуть бути отримані лише для окремих варіантів задачі.

Прискорення сортування може бути забезпечене за допомогою кількох $(p > 1)$ процесорів. Вихідний впорядкований набір у разі розділяється між процесорами; в ході сортування дані пересилаються між процесорами та порівнюються між собою. Результуючий (упорядкований) набір, як правило, розділений між процесорами; при цьому для систематизації такого поділу для процесорів вводиться та чи інша система послідовної нумерації і зазвичай потрібно, щоб при завершенні сортування значення, які розміщуються на процесорах з меншими номерами, не перевищували значення процесорів з

більшими номерами. Залишаючи докладний аналіз проблеми сортування для окремого розгляду, тут основну увагу приділяється вивченню паралельних способів виконання ряду широко відомих методів внутрішнього сортування, коли всі дані, що впорядковуються, можуть бути розміщені повністю в оперативній пам'яті.

При уважному розгляді способів упорядкування даних, що застосовуються в алгоритмах сортування, можна помітити, що багато методів засновані на застосуванні однієї і тієї ж базової операції "порівняти і переставити", що складається в порівнянні тій чи іншій парі значень сортованого набору даних та перестановці цих значень, якщо їх порядок не відповідає умовам сортування.

```
if ( A[i] > A[j] ) {
  temp = A [i];
  A[i] = A[j];
  A[j] = temp;
}
```

Рисунок 2.1 – Псевдокод операції "порівняти та переставити"

Цілеспрямоване застосування цієї операції дозволяє впорядкувати дані; в методах вибору пар значень для порівняння, власне, і проявляється відмінність алгоритмів сортування.

В якості завдання для виконання практичних робіт було обрано алгоритм сортування «парна-непарна перестановка». Парне-непарне сортування також відоме як «odd-even sort». Це відносно простий алгоритм сортування, розроблений для використання на паралельних процесорах. Даний алгоритм сортування порівнюється з сортуванням бульбашкою, з яким він поділяє багато спільного. Алгоритм бульбашкового сортування в прямому вигляді досить складний для розпаралелювання, адже порівняння пар значень

впорядковуємого набору даних відбувається строго послідовно. У зв'язку з цим для паралельного застосування у звичайних випадках використовується не сам цей алгоритм, а його модифікація, відома в літературі як метод парно-непарної перестановки. Головною перевагою даного сортування є те, що його дуже легко реалізувати на паралельній системі, що послужило тому, що його було обрано на виконання практичних робіт. Недоліком парно-непарної перестановки у класичній реалізації, є його низька швидкість.

2.2 Опис та оцінка складності послідовного алгоритму методу сортування «парна-непрана перестановка»

Послідовний алгоритм сортування парної-непарної перестановки діє наступним чином: порівнюються всі парні (even) / непарні (odd) пари проіндексованих суміжних елементів в списку, і якщо пара знаходиться в неправильному порядку (перший більше, ніж другий) елементи міняються місцями. Наступним кроком повторює це для парних/непарних індексованих пар (суміжних елементів). Чергуються парні/непарні та непарні/парні кроки, поки список не буде відсортований.

Далі на рисунку 2.2 наведено приклад сортування масиву чисел парно-непарною перестановкою.

Рисунок 2.2 – Результат виконання алгоритму сортування на 8 елементах

Всього за 8 фаз, та 28 операцій порівняння масив було відсортовано.

Всього алгоритм використовує два цикли: один зовнішній цикл, та один внутрішній. Зовнішній цикл буде містити $\frac{n}{2}$ парних фаз, і $\frac{n}{2}$ непарних фаз, тому загальна складність зовнішнього циклу буде:

$$T\left(\frac{n}{2}+\frac{n}{2}\right). \tag{2.5}$$

На парній фазі, внутрішній цикл алгоритму буде становити $\frac{n}{2}$ порівнянь та $\frac{n}{2}$ обмінів, тому загальна складність $\frac{n}{2}$ парних фаз буде:

$$T\left(\frac{n}{2}\cdot\left(\frac{n}{2}+\frac{n}{2}\right)\right)=T\left(\frac{n}{2}\cdot n\right)=T\left(\frac{n^2}{2}\right). \tag{2.6}$$

На непарній фазі, внутрішній цикл алгоритму буде становити $\frac{n}{2}-1$ порівнянь та $\frac{n}{2}-1$ обмінів. Звідси, загальна складність $\frac{n}{2}$ непарних фаз:

$$T\left(\frac{n}{2}\cdot\left(\frac{n}{2}-1+\frac{n}{2}-1\right)\right)=T\left(\frac{n}{2}\cdot(n-2)\right). \tag{2.7}$$

Тоді, загальний час на виконання послідовного алгоритму сортування парно-непарного сортування буде становити:

$$T_1=\left(\frac{n^2}{2}+\frac{n}{2}\cdot(n-2)\right)\cdot t_{op},. \tag{2.8}$$

де t_{op}– час виконання однієї базової обчислювальної операції ($t_{op} = \frac{1}{3{,}6\ \text{Ггц}};\ 3{,}6\ \text{ГГц} = 3600000000\ \text{Гц}$).

Сортування парно-непарною перестановкою використовує n додаткової пам'яті, а його послідовна складність в найгіршому випадку буде становити $O\left(\frac{n^2}{2} + \frac{n}{2} \cdot (n-2)\right)$ часу, що приблизно дорівнює $O(n^2)$.

2.3 Розробка та тестування програмного додатку послідовного алгоритму

Блок-схема послідовного алгоритму сортування парно-непарної перестановки показана на рисунку 2.3.

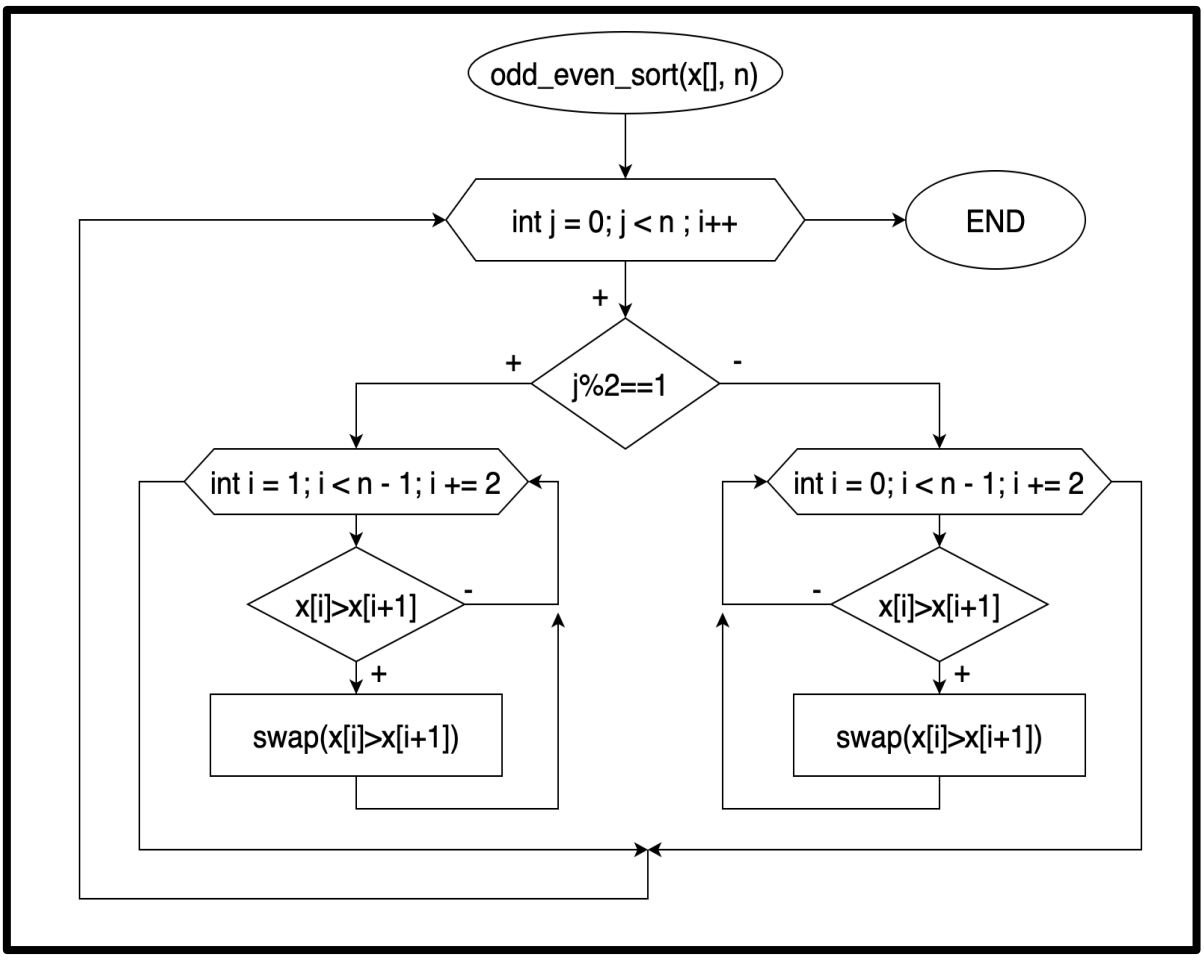

Рисунок 2.3 – Блок-схема послідовного алгоритму сортування парно-непарної перестановки

Програмний додаток для виконання послідовного алгоритму методу сортування на основі парно-непарної перестановки створено на мові C++.

Тестування роботи програми на 8 елементах показано на рисунку 2.4.

Рисунок 2.4 – Результат виконання програми на 8 елементах

Бачимо, що алгоритм відсортував масив з 8 елементів за 41 операцію порівнянь та обмінів.

Тестування роботи програмного додатку на 30 елементах показано на рисунку 2.5.

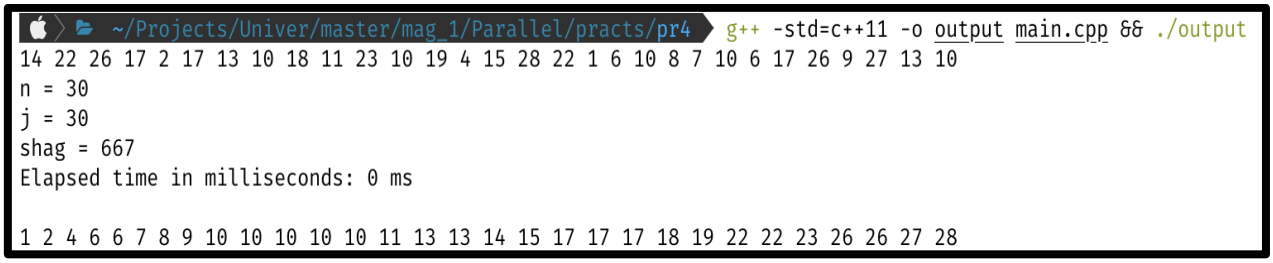

Рисунок 2.5 – Результат виконання програми на 30 елементах

Алгоритм відсортував масив з 30 елементів за 667 операцій порівнянь та обмінів. Розрахунок теоретичної тимчасової складності алгоритму сортування масиву наведено в табл. 2.1.

Таблиця 2.1 – Залежність часу виконання послідовного алгоритму сортування від розміру масиву

Номер тесту	Розмір масиву	Експериментальний час, сек.	Теоретичний час, сек.
1	5000	0,053	0,007

2	10000	0,191	0,028
3	15000	0,441	0,062
4	20000	0,81	0,111
5	25000	1,272	0,174
6	30000	1,87	0,250
7	35000	2,547	0,340
8	40000	3,284	0,444
9	45000	4,199	0,562
10	50000	5,185	0,694

Графік залежності експериментальної та теоретичної часової складності алгоритму сортування показано на рисунку 2.6.

Рисунок 2.6 – Залежність теоретичного та експериментального часу послідовного алгоритму сортування від розміру масиву (реальний час, сек.)

Таблиця 2.2 – Залежність кількості виконаних операцій послідовного алгоритму сортування від розміру масиву

Номер тесту	Розмір масиву	Експериментальна кількість операцій	Теоретична кількість операцій
1	5000	18 693 287	24 995 000

2	10000	74 823 557	99 990 000
3	15000	168 017 298	224 985 000
4	20000	300 013 884	399 980 000
5	25000	467 342 375	624 975 000
6	30000	674 423 748	899 970 000
7	35000	919 554 624	1 224 965 000
8	40000	1 200 280 085	1 599 960 000
9	45000	1 519 797 036	2 024 955 000
10	50000	1 876 984 335	2 499 950 000

Графік залежності експериментальної та теоретичної алгоритмічної складності алгоритму сортування на основі парно-непарної перестановки показано на рисунку 2.7.

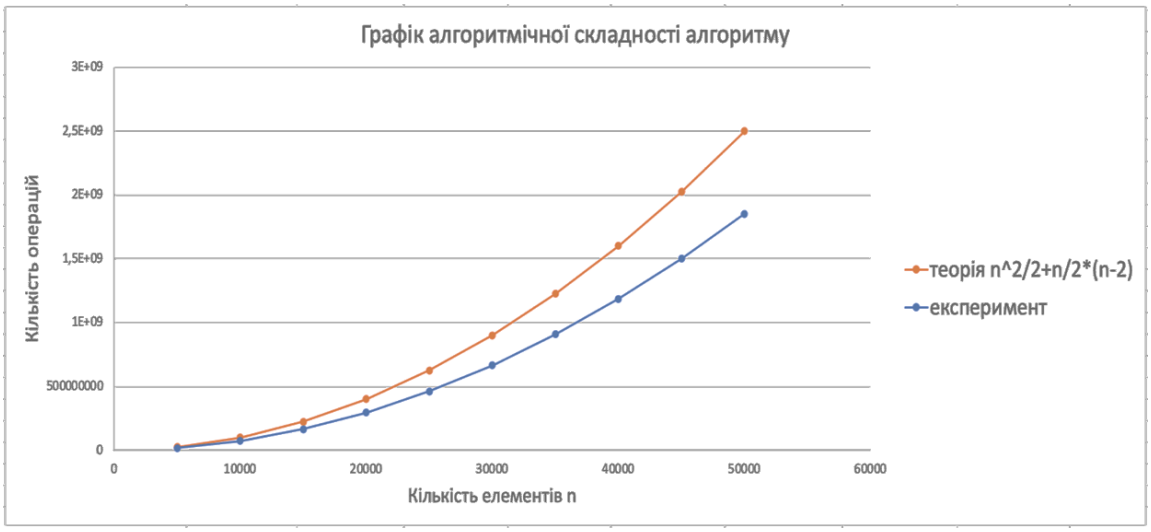

Рисунок 2.7 – Залежність теоретичної та експериментальної кількості виконаних операцій послідовного алгоритму сортування від розміру масиву

2.4 Розробка та оцінка ефективності паралельного алгоритму методу сортування «парна-непрана перестановка»

Для паралельного узагальнення базової операції сортування розглянемо спочатку ситуацію, коли кількість процесорів збігається з числом значень, що сортуються (тобто $p = n$). Тоді порівняння значень a_i і a_j, що

розташовуються, наприклад, на процесорах P_i і P_j можна організувати наступним чином:

- виконати взаємний обмін наявних на процесорах P_i і P_j значень (зі збереженням на цих процесорах вхідних елементів);

- порівняти на кожному процесорі P_i і P_j однакові пари значень (a_i, a_j); результати порівняння використовуються для поділу даних між процесорами – на одному процесорі (наприклад, P_i) залишається менший елемент, інший процесор (тобто P_j) запам'ятовує для дальньої обробки більше значення пари:

$$a_i' = min(a_i, a_j),\ a_j' = max(a_i, a_j). \qquad (2.9)$$

Розглянуте паралельне узагальнення базової операції сортування може бути належним чином адаптоване і для випадку $p < n$, коли кількість процесорів є меншою від числа значень, що впорядковуються. У цій ситуації кожен процесор міститиме вже не єдине значення, а частину (блок розміру $\frac{n}{p}$) набору даних, що сортується. Ці блоки зазвичай упорядковуються на самому початку сортування на кожному процесорі окремо за допомогою якого-небудь швидкого алгоритму (попередня стадія паралельного сортування). Далі, дотримуючись схеми одноелементного порівняння, взаємодія пари процесорів P_i та P_j для спільного впорядкування вмісту блоків A_i та A_j може бути здійснено наступним чином:

- виконати взаємний обмін блоків між процесорами P_i і P_j;

- об'єднати блоки A_i та A_j на кожному процесорі в один відсортований подвійний блок розміру (при вихідній упорядкованості блоків A_i та A_j процедура їхнього об'єднання зводиться до швидкої операції злиття впорядкованих наборів даних);

- розділити отриманий подвійний блок на дві рівні частини і залишити одну з цих частин (наприклад, з меншими значеннями даних) на

процесорі P_i, а іншу частину (з великими значеннями відповідно) – на процесорі P_j.

- розглянута процедура називається, як операція "порівняти і розділити", слід зазначити, що сформовані в результаті такої процедури блоки на процесорах P_i і P_j збігаються за розміром вхідних блоків A_i і A_j, і всі значення процесору P_i, є меншими ніж значення процесору P_j.

Паралельний варіант методу парної-непарної перестановки:

- розбиття даних масиву розміру n на p процесорах, при чому, у разі $p < n$, процесори містять блоки даних розміру $\frac{n}{p}$;

- сортування даних на кожному процесорі окремо одним із швидких методів сортування;

- потім процесори здійснюють $\frac{p}{2}$ непарних і $\frac{p}{2}$ парних ітерацій, у кожній з яких відбувається наступне: суміжні процесори обмінюються своїми елементами, в результаті чого кожен процесор потім сортує не тільки свої елементи, але і елементи сусіднього процесора. Після того, як кожен процесор зробив внутрішнє сортування (на кожній парі процесорів отримуємо однакові масиви), подвійний масив ділиться на 2 частини: лівий процесор оброблює далі лише ліву частину (з меншими значеннями даних), а правий – тільки праву (з великими значеннями даних). Далі здійснюється чергова ітерація. Після здійснення останньої ітерації відбувається збір даних, в результаті чого маємо відсортований масив.

Далі наведено граф впливу для паралельного алгоритму сортування парної-непарної перестановки (рис. 2.8).

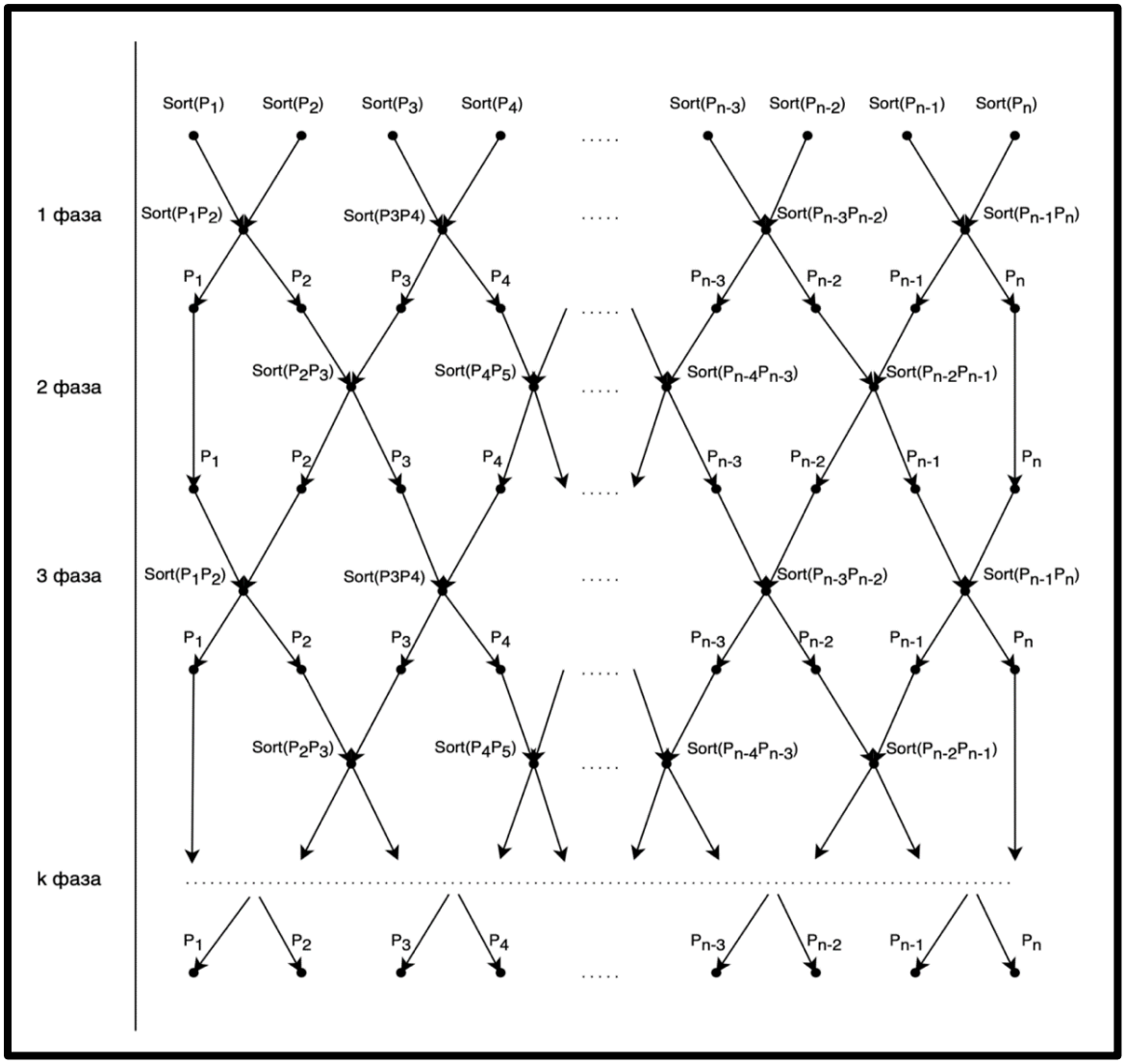

Рисунок 2.8 – Граф впливу паралельного алгоритму сортування

Таблиця 2.3 – Виконання паралельного алгоритму сортування парно-непарної перестановки

Початкові дані	Масив															
	55	13	88	59	29	85	71	43	75	40	18	2	22	43	4	14
№ та тип операції	Процесори															
	1				2				3				4			
Сортування на кожному із процесорів	13	55	59	88	29	43	71	85	2	18	40	75	4	14	22	43
1 фаза непарна (1,2)(3,4)	13	55	59	88	29	43	71	85	2	18	40	75	4	14	22	43
	13	29	43	55	59	71	85	88	2	4	14	18	22	40	43	75
2 фаза парна (2,3)	13	29	43	55	59	71	85	88	2	4	14	18	22	40	43	75
	13	29	43	55	2	4	14	18	59	71	85	88	22	40	43	75
3 фаза непарна (1,2)(3,4)	13	29	43	55	2	4	14	18	59	71	85	88	22	40	43	75
	2	4	13	14	18	29	43	55	22	40	43	59	71	75	85	88
4 фаза парна (2,3)	2	4	13	14	18	29	43	55	22	40	43	59	71	75	85	88
	2	4	13	14	18	22	29	40	43	43	55	59	71	75	85	88
Кінцеві дані	Масив															
	2	4	13	14	18	22	29	40	43	43	55	59	71	75	85	88

В таблиці 2.3 наведено приклад використання паралельного алгоритму парно-непарної перестановки при $n = 16,\ p = 4$. За такої умови, на кожному процесорі буде знаходитись $\frac{n}{p} = 4$ елемента. Бачимо, що для сортування даних паралельним методом знадобилось p ітерацій.

В якості топології будемо використовувати топологію 1D-тор, так як вона найбільш підходить до даного алгоритму сортування, а також тому що обмінюватись дані будуть лише між сусідніми процесами.

Нехай:

- n – кількість розмір масиву;
- p – кількість процесорів (кількість блоків даних);
- $n > p$;
- $\frac{n}{p}$ – кількість даних в блоках;

тоді масив розбивається на блоки A_i, де *i=1,p*.

Перший крок: сортування блоків на кожному із процесорів окремо деяким швидким алгоритмом сортування (рис. 2.9).

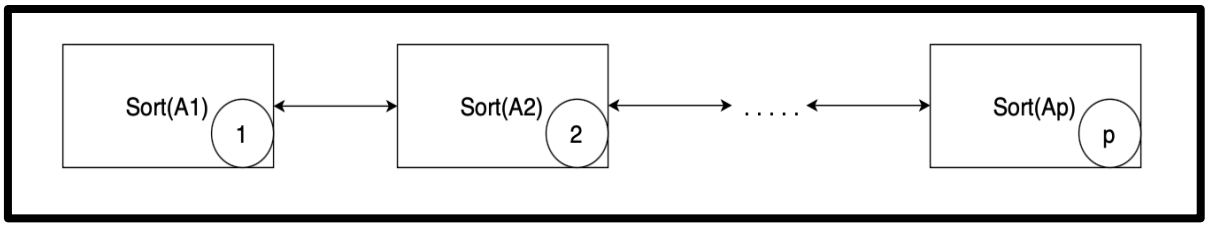

Рисунок 2.9 – Відображення сортування на кожному процесорі на топології 1D-тор

Другий крок: на кожній ітерації паралельної сортування взаємодіючі пари процесорів здійснюють передачу блоків між собою, після чого одержувані на кожному процесорі пари блоків об'єднуються за допомогою процедури злиття.

Послідовність схем топологій на парній та непарній ітерації може бути представлено у в рисунку 2.10.

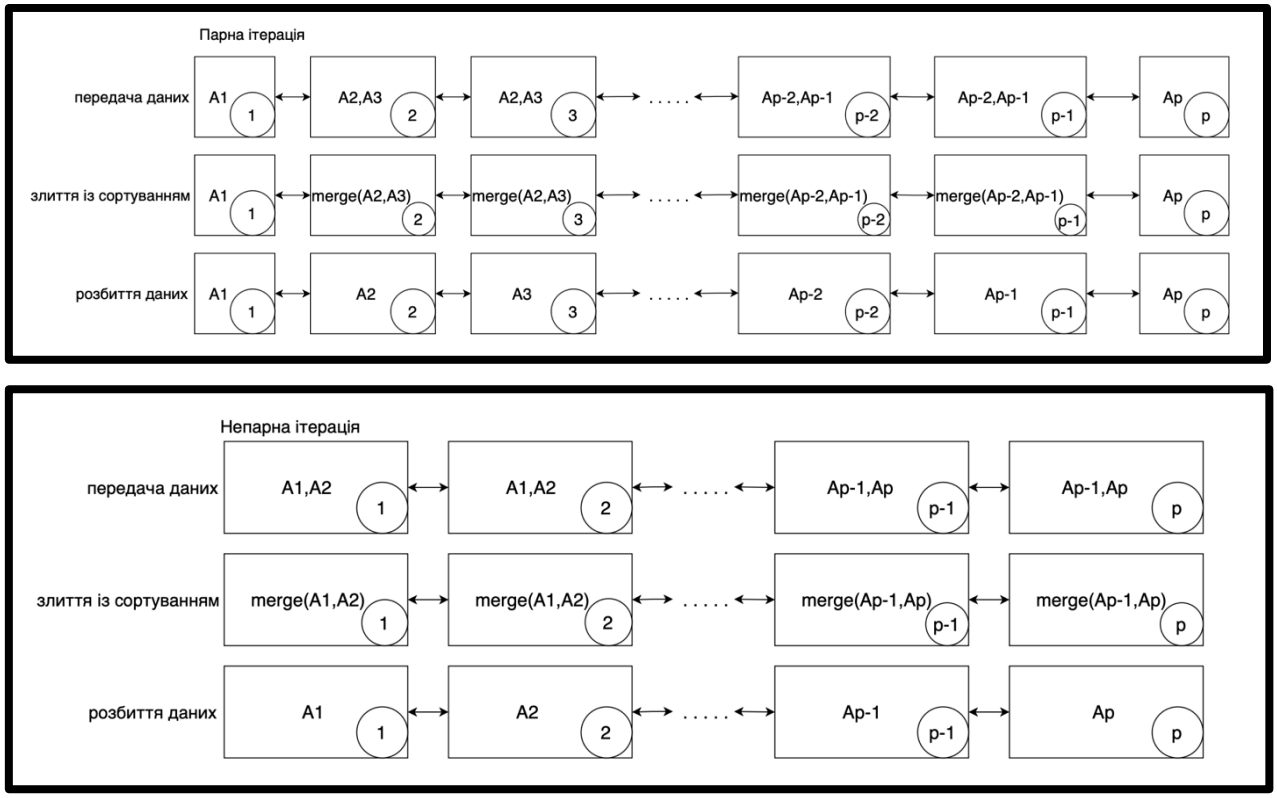

Рисунок 2.10 – Відображення виконання алгоритму на топології 1D-тор

а) парна ітерація

б) непарна ітерація

Загальна схема відображення паралельного алгоритму парно-непарної перестановки на топологію «лінійка» представлена в рисунку 2.11.

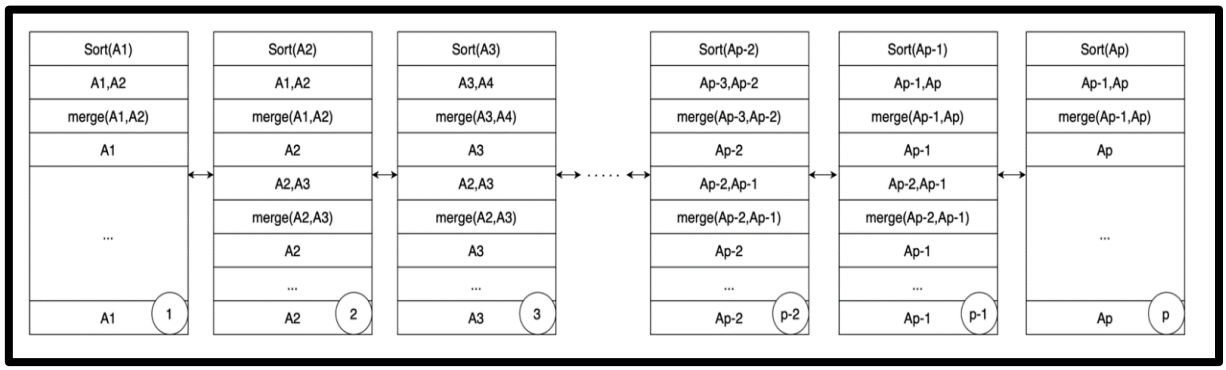

Рисунок 2.11 – Відображення паралельного алгоритму сортування парно-непарної перестановки на топології 1D-тор

В даному паралельному алгоритмі сортування масиву даних використовується швидкий послідовний алгоритм для внутрішнього сортування на кожному із процесів окремо, послідовна складність якого:

$$T_1 = n\ log\ log\ n\ \cdot t_{op}, \tag{2.10}$$

де t_{op} – час виконання однієї базової обчислювальної операції $\left(t_{op} = \frac{1}{3{,}6\ \text{Ггц}}\right)$.

Визначимо складність паралельного алгоритму парно-непарної перестановки для упорядкування даних. Як зазначалося раніше в описі алгоритму, на початковій стадії роботи методу кожен процесор проводить упорядкування своїх блоків даних (розмір блоків при рівномірному розподілі даних дорівнює n/p). Припустимо, що це початкове сортування може бути виконано за допомогою швидкодіючих алгоритмів упорядкування даних, тоді трудомісткість початкової стадії обчислень можна визначити виразом виду:

$$T_{p,comp}^{odd-even,1} = \frac{n}{p}\ log\ log\ \frac{n}{p} \cdot t_{op}\ . \tag{2.11}$$

Далі, на кожній ітерації паралельної сортування взаємодіючі пари процесорів здійснюють передачу блоків між собою, після чого одержувані на кожному процесорі пари блоків об'єднуються за допомогою процедури злиття. Загальна кількість ітерацій не перевищує величини p, а злиття двох блоків даних потребує $2 \cdot n/p$, тому загальна кількість операцій цієї частини паралельних обчислень виявляється рівною:

$$T_{p,comp}^{odd-even,2} = 2 \cdot \frac{n}{p} \cdot p \cdot t_{op} = 2n \cdot t_{op}. \tag{2.12}$$

Загальна складність паралельного алгоритму упорядкування даних буде визначатись виразом:

$$T_{p,comp}^{odd-even} = T_{p,comp}^{odd-even,1} + T_{p,comp}^{odd-even,2} = \left(\frac{n}{p}\ log\ log\ \frac{n}{p} + 2n\right) \cdot t_{op}. \tag{2.13}$$

Врахуємо тривалість виконуваних обчислювальних операцій та оцінимо трудомісткість операції передачі блоків між процесорами.

За простою моделлю Хокні для режиму передачі повідомлень при довжині маршруту $l = 1$ (дані передаються «сусіду»), максимально p раз, маємо:

$$T_{p,comm}^{odd-even} = (t_s + V \cdot t_w) \cdot p, \tag{2.14}$$

де V – обсяг даних, що переміщуються, в даному випадку це кількість елементів одного блоку масиву або блок обсягу:

$$V = \frac{n}{p}, \tag{2.15}$$

t_s – латентність (час на підготовку повідомлення для передачі);

t_w – час передачі одного байту;

При використанні моделі Хокні загальний час всіх виконуваних під час сортування операцій передачі блоків можна оцінити за допомогою співвідношення виду:

$$T_{p,comm}^{odd-even} = (t_s + \frac{n}{p} \cdot t_w) \cdot p. \tag{2.16}$$

З урахуванням трудомісткості комунікаційних процесів загальний час виконання паралельного алгоритму сортування визначається наступним виразом:

$$T_p^{odd-even} = T_{p,comp}^{odd-even} + T_{p,comm}^{odd-even} =$$
$$= \left(\frac{n}{p} log\ log\ \frac{n}{p} + 2n\right) \cdot t_{op} + (t_s + \frac{n}{p} \cdot t_w) \cdot p. \tag{2.17}$$

Коефіцієнти прискорення та ефективності, відповідно, обчислюються за наступними виразами:

$$S_p^{odd-even} = \frac{T_1}{T_p^{odd-even}} = \frac{n\ log\ log\ n\ \cdot t_{op}}{\left(\frac{n}{p} log\ log\ \frac{n}{p} + 2n\right) \cdot t_{op} + (t_s + \frac{n}{p} \cdot t_w) \cdot p} =$$
$$\frac{p\ log\ log\ n\ \cdot t_{op}}{log\ log\ \frac{n}{p} + 2p \cdot t_{op} + (t_s + \frac{n}{p} \cdot t_w) \cdot p},$$
$$E_p^{odd-even} = \frac{S_p}{p} = \frac{n\ log\ log\ n\ \cdot t_{op}}{p \times \left(\frac{n}{p} log\ log\ \frac{n}{p} + 2n\right) \cdot t_{op} + (t_s + \frac{n}{p} \cdot t_w) \cdot p} =$$

$$= \frac{log\ log\ n\ \cdot t_{op}}{\left(log\ log\ \frac{n}{p} + 2p\right) \cdot t_{op} + (t_s + \frac{n}{p} \cdot t_w) \cdot p}.$$

Тоді, загальні накладні витрати на паралелізм для цього алгоритму розраховуються як:

$$T_0 = p \cdot T_p - T_1 = p \cdot \left(\frac{n}{p} log\ log\ \frac{n}{p} + 2n\right) \cdot t_{op} + (t_s + \frac{n}{p} \cdot t_w) \cdot p - n\ log\ log\ n\ .$$

У таблиці 2.4 наведено приклад відображення кільця з 8 процесорів на гіперкуб $H(3)$. Усі сусідні процеси у топології кільце є сусідніми у гіперкубі.

Таблиця 2.4 – Логічне відображення кільця на гіперкуб

Код Грея $N = 1$	Код Грея $N = 2$	Код Грея $N = 3$	Номера процесорів	
			Гіперкуб	Кільце
0	00	000	0	0
1	01	001	1	1
	11	011	3	2
	10	010	2	3
		110	6	4
		111	7	5
		101	5	6
		100	4	7

Відображення на 2D-торі наведено нижче.

У таблиці 2.5 продемонстровано відображення кільця з 16 процесорами. Відображення виконано за допомогою кодів Грея на основі таких перетворень:

$$G(0,1) = 0, G(1,1) = 1,$$
$$G(i, s+1) = \{G(i,s), i < 2^s, 2^s + G(2^{s+1} - 1 - i, s), i \geq 2^s.$$

Таблиця 2.5 – Логічне відображення кільця на 2D-тор

Код	Код Грея	Код Грея	Код Грея	Номера процесорів

Грея $N = 1$	$N = 2$	$N = 3$	$N = 4$	2D-тор	Кільце
0	00	000	0000	0	0
1	01	001	0001	1	1
	11	011	0011	3	2
	10	010	0010	2	3
		110	0110	6	4
		111	0111	7	5
		101	0101	5	6
		100	0100	4	7
			1100	12	8
			1101	13	9
			1111	15	10
			1110	14	11
			1010	10	12
			1011	11	13
			1001	9	14
			1000	8	15

У таблицях наведено номери процесорів, які є відповідними для розглянутих топологій.

Таким чином, було виконано опис послідовного алгоритму сортування парно-непарної перестановки, наведено приклад виконання такого алгоритму. Експериментальним шляхом було доведено, що послідовний алгоритм має алгоритмічну і часову складність в найгіршому випадку $O(n^2)$. Також було розроблено програму, написану на мові програмування C++, що реалізує послідовний алгоритм сортування парно-непарною перестановкою. Розробка паралельного алгоритму парно-непарної перестановки містить: детальний та вичерпний опис виконання паралельного алгоритму. Було виконано зображення даного алгоритму на графах впливу, а також обґрунтовано і відображено схему паралельного алгоритму на топологію з’єднання 1D-тор Наведено аналітичні вирази динамічних характеристик паралельного алгоритму за моделлю Хокні.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Foster I. Designing and Building Parallel Programs [Електронний ресурс]. - Режим доступу : https://www.mcs.anl.gov/~itf/dbpp. - (Дата перегляду 10.04.2025).

2. Introduction to Parallel Computing / A. Grama, A. Gupta, G. Karypis, V. Kumar. - Addison Wesley, 2003. - 856 p.

3. Назарова І. А. Паралельні обчислення / І.А. Назарова, О.А. Дмитрієва. - Покровськ: ДВНЗ «ДонНТУ», 2020. - 246 с.

4. Луцків А. М. Паралельні та розподілені обчислення: навч. посіб. / А.М. Луцків, С.А. Лупенко, В.В. Пасічник. - Львів: Магнолія, 2017. 566 с.

5. Pacheco Peter S. An Introduction to Parallel Programming / Peter S. Pacheco. - 2nd Edition. - Morgan Kaufmann publications, 2020. - 450 p.

6. Методичні вказівки до практичних занять з дисципліни «Паралельні інформаційні системи» для студентів ОС «магістр» спеціальності 121 Інженерія програмного забезпечення денної та заочної форм навчання [Електронний ресурс] / уклад. І.А. Назарова, О.В. Александрова, М.О. Александров. - Луцьк: ДонНТУ, 2024. - 26 с. - Режим доступу : http://lc.donntu.edu.ua/elcat/download3/22608. - (Дата перегляду 10.04.2025).

7. Introduction to Parallel Computing [Електронний ресурс]. - Режим доступу : https://hpc.llnl.gov/documentation/tutorials/introduction-parallel-computing-tutorial/ - (Дата перегляду 10.04.2025).

8. MPI FORUM [Електронний ресурс]. - Режим доступу : https://www.mpi-forum.org. - (Дата перегляду 10.04.2025).

Додаток А

Теми індивідуальних варіантів практичних робіт

Чисельні методи:

1. Добуток матриці на вектор (блокові, стрічкові, для щільно-заповнених матриць)
2. Добуток матриці на вектор (блокові, стрічкові, для розріджених матриць різних типів)
3. Матричний добуток (блокові, стрічкові, для щільно-заповнених матриць)
4. Матричний добуток (блокові, стрічкові, для розріджених матриць різних типів)
5. Швидке матричне множення (метод Штрассена-Винограда і його модифікації)
6. Розв’язання системи лінійних алгебраїчних рівнянь (СЛАР) методом Гауса, Жордана, Халецького, тощо
7. Розв’язання СЛАР методом сполучених градієнтів
8. Розв’язання системи нелінійних алгебраїчних рівнянь (проста ітерація, Ньютона, тощо)
9. Розв’язання систем лінійних однорідних диференційних рівнянь (СЛОДУ) з постійними коефіцієнтами експоненціальним методом
10. Вкладені експоненціальні методи розв’язання СЛОДУ з постійними коефіцієнтами
11. Експоненціальні екстраполяційні методи розв’язання СЛОДУ з постійними коефіцієнтами
12. Розв’язання систем лінійних неоднорідних рівнянь (СЛОДУ) з постійними коефіцієнтами експоненціальним методом

13. Розв’язання систем звичайних диференційних рівнянь СЗДР однокроковими явними багатостадійними методами

14. Розв’язання СЗДР однокроковими повністю неявними багатостадійними методами

15. Розв’язання СЗДР однокроковими діагонально-неявними багатостадійними методами

16. Розв’язання СЗДР однокроковими вкладеними явними багатостадійними методами

17. Розв’язання СЗДР однокроковими вкладеними явними багатостадійними методами

18. Розв’язання СЗДР однокроковими явними багатостадійними методами на базі локальної екстраполяції Річардсона

19. Розв’язання СЗДР однокроковими блоковими або багатоточковими методами

20. Розв’язання СЗДР однокроковими вкладеними блоковими методами

21. Розв’язання СЗДР однокроковими екстраполяційними блоковими методами

22. Розв’язання СЗДР однокроковими блоковими методами з правилом Рунге

Методи сортування:

1. Метод «бульбашкового» сортування
2. Метод «вставками»
3. Парна-непарна перестановка
4. Метод сортування злиттям
5. Метод сортування Шелла
6. Метод сортування Бетчера
7. Базове швидке сортування

8. Покращене швидке сортування

Алгоритми для розв’язання задач на графах:

1. Пошук незалежних множин вершин у графі
2. Алгоритм Брона-Кербоша
3. Пошук клік графа
4. Пошук домінуючих множин у графі
5. Пошуки «у ширину» та «глибину»
6. Пошук найкоротших маршрутів (Форда, Флойда, хвильовий тощо)
7. Побудова кістяка мінімальної ваги (Краскала, Прима)
8. Розв’язання завдання комівояжера або «китайського листоноші»
9. Вершинне або реберне розфарбування
10. Алгоритм плоского укладання графа
11. Алгоритм пошуку сильних компонент орієнтованого графа (орграфа)
12. Алгоритм побудови конденсації орграфа
13. Алгоритм знаходження бази або антибази орграфа
14. Визначення ізоморфізму графів
15. Визначення ізоморфної вкладеності графів
16. Алгоритм пошуку максимального потоку або алгоритм Форда-Фалкерсона
17. Алгоритми пошуку ейлерева циклу в графі
18. Алгоритми пошуку гамільтонова циклу в графі
19. Алгоритми побудови абстрактних дерев заданого порядку та кількості центральних вершин.

Додаток Б
Рекомендації до обрання варіанту

Індивідуальний варіант реального обчислювального завдання для виконання всіх практичних робіт є єдиним. Номер варіанту може бути заданий викладачем з представлених в додатку А.

Студент також має можливість самостійно запропонувати тему для практичних робіт, особливо якщо вона пов’язана з його навчально-науковою, практичною чи дослідницькою діяльністю (це є рекомендованим підходом до вибору теми).

Обрана тема може передбачати ускладнення або розширення завдань, наведених у навчальних матеріалах дисциплін «Паралельні інформаційні системи», «Дискретна математика», «Дискретні структури та алгоритми», «Чисельні методи» та, власне, ТОРО. Наприклад, можна реалізувати всі відомі і розглянуті алгоритми паралельного множення матриць, методи пошуку найкоротших маршрутів в графах тощо.

Крім того, тема може передбачати модифікацію постановок завдань, розглянутих в представлених вище дисциплінах. Наприклад, в задачах матричних обчислень замість щільних матриць можуть використовуватися розріджені або матриці спеціального типу, для диференційних рівнянь добавити методи керування кроком інтегрування і так далі. Також тема може охоплювати розширення спектра задач і методів їх розв’язання: обчислення з матрицями та графами, розв’язання диференціальних рівнянь у часткових похідних, цифрова обробка сигналів, обробка зображень тощо.

В любому випадку, вибір індивідуального варіанту теми для виконання практичних робіт за дисципліною ТОРО повинен бути узгоджений та затверджений викладачем.