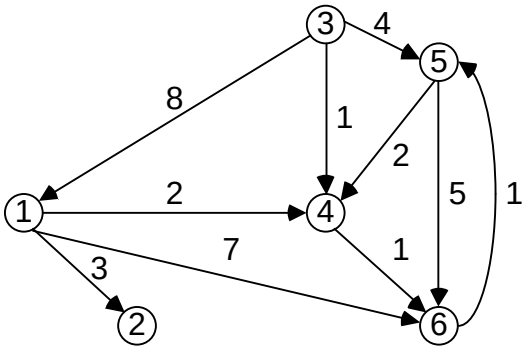
Мета роботи: розробити паралельні реалізації алгоритмів оброблення графів з допомогою технології MPI.

Теоретичні відомості

Математичні моделі у вигляді графів широко використовуються у випадку моделювання різноманітних явищ, процесів і систем. Як результат, багато які теоретичні й реальні прикладні задачі можуть бути розв'язані за допомогою тих або інших процедур аналізу графових моделей. Серед безлічі цих процедур може бути виділений деякий певний набір типових алгоритмів оброблення графів. Розгляду питань теорії графів, алгоритмів моделювання, аналізу й розв'язанню задач на графах присвячено досить багато різних видань, як можливий посібник з цієї тематики може бути рекомендована робота Кормен і ін. (1999). Почнемо ж ми з того, що запровадимо деякі основні графові позначення.

Нехай граф $G=(V,E)$ містить множину $V=\{v_i\}$ із n вершин (вузлів) і множину $E\subseteq V\times V=\{(v_i,v_j)\}$ із m ребер (дуг), що з'єднують вершини в графі G. В орієнтованому графі, крім того, кожна дуга має напрямок, тому дуги (v_i,v_j) й (v_j,v_i) різні, якщо $i\neq j$. Орієнтований граф може бути представлений матрицею суміжності $A=[a_{ij}]$, у якій кожний елемент a_{ij} представляє дугу між елементами i та j. $a_{ij}=1$, якщо існує дуга (v_i,v_j); у противному випадку $a_{ij}=0$. У зваженому графі, крім того, кожній дузі додатково зіставляють вагу w_{ij} – дійсне число. Приклад зваженого орієнтованого графа наведений на рис. 6.1 (а).

Відомі різні способи задання графів. За наявності малої кількості дуг у графі (тобто $m\ll n^2$) доцільно використовувати для визначення графів списки, що перераховують наявні в графах дуги. Представлення достатньо щільних графів, для яких майже всі вершини з'єднані між собою дугами (тобто $m\sim n^2$), може бути ефективно забезпечене за допомогою матриці ваг $W=[w_{ij}]$, де w_{ij} – вага дуги (v_i,v_j). Ваги неіснуючих дуг будемо вважати рівними 0 у випадку $i=j$ або ∞ у противному випадку. Зауважимо, що матриця ваг є простим узагальненням матриці суміжності. Так, наприклад, матриця ваг, що відповідає графу на рис. 6.1 (а), наведена на рис. 6.1 (б).



а)

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

б)

Рис. 6.1. Приклад зваженого орієнтованого графа та його матриці ваг

Як позитивний момент такого способу представлення графів можна відзначити, що використання матриці ваг дозволяє застосовувати під час реалізації обчислювальних процедур аналізу графів матричні алгоритми оброблення даних. Далі ми розглянемо способи паралельної реалізації алгоритмів на графах на прикладі задачі пошуку мінімального кістякового дерева графа й задачі пошуку найкоротших шляхів між усіма парами пунктів призначення. Для представлення графів під час розгляду всіх перерахованих задач будуть використовуватися матриці ваг.

Задача пошуку мінімального кістякового дерева

Кістяковим деревом неорієнтованого G графа називають підграф T графа G, який є деревом і містить усі вершини з G. Визначивши вагу підграфа для зваженого графа як суму ваг дуг, що входять до складу підграфа, під мінімальним кістяковим деревом (МКД) T будемо розуміти кістякове дерево мінімальної ваги. Змістовна інтерпретація задачі пошуку МКД може полягати, наприклад, у практичному прикладі побудови локальної мережі персональних комп'ютерів із прокладанням сполучних ліній зв'язку мінімальної довжини. Приклад зваженого неорієнтованого графа та відповідного мінімального кістякового дерева наведений на рис. 6.2.

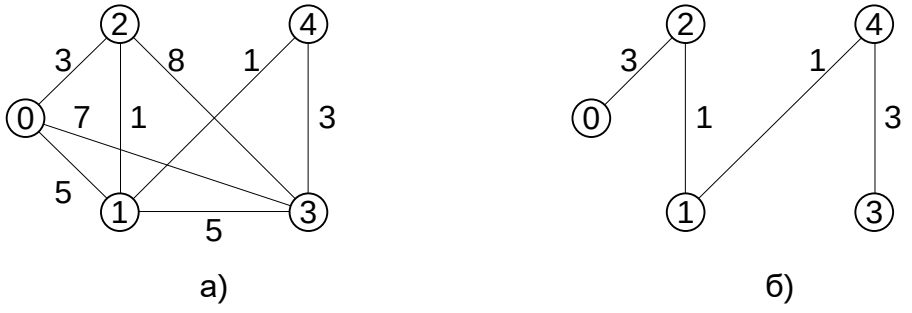


Рис. 6.2. Приклад зваженого неорієнтованого графа (а) та відповідного мінімального кістякового дерева (б)

Наведемо загальніший опис алгоритму розв'язання поставленої задачі, відомої під назвою методу Прима (Prim); докладніша інформація може бути отримана, наприклад, у книзі Кормен і ін. (1999).

Послідовний алгоритм Прима. Алгоритм починає роботу з довільної вершини графа G, обираної як корінь майбутнього МКД, і в ході послідовно виконуваних ітерацій розширює дерево, що конструюється, до МКД. Нехай V_T і E_T суть множина вершин і множина дуг відповідно, уже включених алгоритмом до МКД, а величини $d_i, 1 \le i \le n$ характеризують дуги мінімальної довжини від вершин, ще не включених до дерева, до множини V_T, тобто

$$\forall i \notin V_T \to d_i = \min\left\{w_{ij} : j \in V_T, \left(v_i, v_j\right) \in E\right\}$$

(якщо для якої-небудь вершини $i \notin V_T$ не існує жодної дуги в V_T, значення d_i встановлюють в ∞). На початку роботи алгоритму вибирають довільну вершину первинного дерева s як кореневу вершину МКД і вважають $V_T = \{s\}, E_T = \varnothing, d_s = 0$.

На кожній ітерації алгоритму Прима виконують такі дії:

- визначають значення величин d_i для всіх вершин, ще не включених до складу МКД;
- вибирають вершину t графа G, що має дугу мінімальної ваги до множини V_T, тобто таку, що $d_t = \min(d_i), i \notin V_T$;
- вершина t включається до V_T, а відповідна дуга – до E_T.

Після виконання $n-1$ ітерації методу буде сформовано МКД. Вага цього дерева є сумою ваг дуг, включених до складу множини E_T. Трудомісткість пошуку МКД характеризується квадратичною залежністю від кількості вершин графа, тобто $O(n^2)$.

Поділ обчислень на незалежні частини. Оцінимо можливості паралельного виконання розглянутого алгоритму пошуку мінімального кістякового дерева.

Ітерації методу повинні виконуватися послідовно й тому не можуть бути розпаралелені. З іншого боку, дії, виконувані на кожній ітерації алгоритму, є незалежними й можуть реалізовуватися одночасно. Так, наприклад, визначення величин d_i може здійснюватися для кожної вершини графа окремо, пошук дуги мінімальної ваги може бути реалізований за каскадною схемою тощо.

Розподіл даних між процесорами обчислювальної системи повинен забезпечувати незалежність перерахованих операцій алгоритму Прима. Зокрема, це може бути реалізовано, якщо кожна вершина графа розташовується на процесорі разом з усією пов'язаною з вершиною інформацією. Дотримання цього принципу приводить до того, що за умови рівномірного завантаження кожний процесор $p_j, 1 \le j \le p$, повинен містити:

- набір k вершин $V_j = \{v_{k\cdot j+1}, v_{k\cdot j+2}, ..., v_{k\cdot(j+1)}\}$, $k = \lceil n/p \rceil$;
- блок з k величин $\Delta_j = \{d_{k\cdot j+1}, d_{k\cdot j+2}, ..., d_{k\cdot(j+1)}\}$, що відповідає цьому набору;
- вертикальну смугу матриці ваг W графа G із k сусідніх стовпців $W_j = \{w_{k\cdot j+1}, w_{k\cdot j+2}, ..., w_{k\cdot(j+1)}\}$ (w_i є i-им стовпцем матриці W);
- формовані в процесі обчислень множини вершин V_T і дуг E_T.

Як підсумок можна дійти висновку, що базовою підзадачею у паралельному алгоритмі Прима може слугувати процедура обчислення блоку значень Δ_j для вершин V_j матриці ваг W графа G.

Виділення інформаційних залежностей. З урахуванням вибору базових підзадач загальна схема паралельного виконання алгоритму Прима буде полягати в наступному:

- визначення вершини t графа G, що має дугу мінімальної ваги до множини V_T; для вибору такої вершини необхідно здійснити пошук мінімуму в наборах величин d_i, наявних на кожному із процесорів, і виконати збирання отриманих значень на одному із процесорів;
- передача всім процесорам номера вибраної вершини для включення до складу кістякового дерева;
- оновлення наборів величин d_i з урахуванням додавання нової вершини.

Таким чином, у ході паралельних обчислень між процесорами здійснюються два типи інформаційних взаємодій – збирання даних від усіх процесорів на одному з

процесорів і передача повідомлень від одного процесора всім процесорам обчислювальної системи.

Масштабування та поділ підзадач між процесорами. Відповідно до визначення кількість базових підзадач завжди відповідає кількості наявних процесорів, і тому проблема масштабування для паралельного алгоритму не виникає.

Поділ підзадач між процесорами повинен враховувати характер виконуваних в алгоритмі Прима комунікаційних операцій. Для ефективної реалізації необхідних інформаційних взаємодій між базовими підзадачами топологія мережі передачі даних повинна мати структуру гіперкуба або повного графа.

Задача пошуку всіх найкоротших шляхів

Вихідною інформацією для задачі є зважений граф $G=(V,E)$, що містить n вершин $(|V|=n)$, у якому кожній дузі графа приписана невід'ємна вага. Граф будемо вважати орієнтованим, тобто, якщо з вершини i існує дуга до вершини j, то з цього не випливає наявності зворотної дуги з вершини j до вершини i. У тому випадку, якщо вершини все-таки з'єднані взаємооберненими дугами, то ваги, приписані їм, можуть не збігатися. Розглянемо задачу, в якій для наявного графа G потрібно знайти шляхи мінімальної довжини між кожною парою вершин графа. Як практичний приклад можна навести задачу складання маршруту руху транспорту між різними містами за заданими відстанями між населеними пунктами й інші подібні задачі.

Для розв'язання задачі пошуку найкоротших шляхів між усіма парами пунктів призначення використаємо алгоритм Флойда-Уоршелла (Floyd-Warshall) (див., наприклад, Кормен і ін. (1999)).

Послідовний алгоритм Флойда-Уоршелла. Для пошуку мінімальних відстаней між усіма парами пунктів призначення Флойд та Уоршелл запропонували алгоритм, складність якого має порядок $O(n^3)$. У загальному вигляді цей алгоритм може бути представлений у такий спосіб:

```
// Лістинг 6.1. Послідовний алгоритм Флойда-Уоршелла
 for (k = 0; k < n; k++)
  for (i = 0; i < n; i++)
   for (j = 0; j < n; j++)
    W[i,j] = min(W[i,j], W[i,k]+W[k,j]);
```

(реалізація операції вибору мінімального значення $\min$ повинна враховувати спосіб збереження в матриці ваг неіснуючих дуг графа). Як можна помітити, у ході виконання алгоритму матриця ваг W змінюється, після завершення обчислень у матриці W буде зберігатися необхідний результат, тобто довжини мінімальних шляхів для кожної пари вершин первинного графа.

Поділ обчислень на незалежні частини. Як випливає з загальної схеми алгоритму Флойда-Уоршелла, основне обчислювальне навантаження під час розв'язання задачі пошуку найкоротших шляхів полягає у виконанні операції вибору мінімального значення (див. лістинг 6.1). Зазначена операція є досить простою, отже, її розпаралелювання не приведе до помітного прискорення обчислень. Ефективніший спосіб організації паралельних обчислень може полягати в одночасному виконанні декількох операцій оновлення значень матриці W.

Покажемо коректність такого способу організації паралелізму. Для цього потрібно довести, що операції оновлення значень матриці W на тій самій ітерації

зовнішнього циклу k можуть виконуватися незалежно. Іншими словами, слід показати, що на ітерації k не відбувається зміни елементів W_{ik} і W_{kj} для жодної пари індексів (i, j). Розглянемо вираз, згідно з яким відбувається зміна елементів матриці W:

$$W_{ij} = \min\left(W_{ij}, W_{ik} + W_{kj}\right).$$

Для $i = k$ отримаємо

$$W_{kj} = \min\left(W_{kj}, W_{kk} + W_{kj}\right),$$

але тоді значення W_{kj} не зміниться, оскільки $W_{kk} = 0$.

Для $j = k$ вираз матиме такий вигляд:

$$W_{ik} = \min\left(W_{ik}, W_{ik} + W_{kk}\right),$$

що також показує незмінність значень W_{ik}. Як результат, необхідні умови для організації паралельних обчислень забезпечені й, тим самим, як базова підзадача може бути використана операція оновлення елементів матриці W (для вказування підзадач будемо використовувати індекси елементів, значення яких оновлюються у відповідних підзадачах).

Виділення інформаційних залежностей. Виконання обчислень у підзадачах стає можливим тільки тоді, коли кожна підзадача (i, j) містить необхідні для розрахунків елементи W_{ij}, W_{ik}, W_{kj} матриці W. Для виключення дублювання даних розмістимо в підзадачі (i, j) єдиний елемент W_{ij}, тоді отримання всіх інших необхідних значень може бути забезпечене тільки за допомогою передачі даних. Таким чином, кожний елемент W_{kj} k-го рядка матриці W повинен бути переданий усім підзадачам $(k, j), 1 \le j \le n$, а кожний елемент W_{ik} k-го стовпця матриці W повинен бути переданий усім підзадачам $(i, k), 1 \le i \le n$, як показано на рис. 6.3.

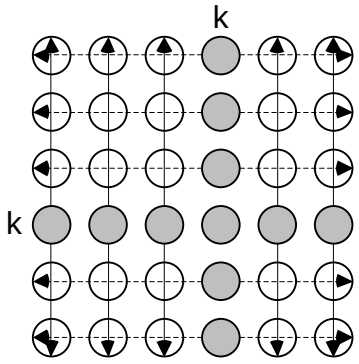


Рис. 6.3. Інформаційна залежність базових підзадач (стрілками показані напрямки обміну значеннями на ітерації *k*)

Масштабування та поділ підзадач між процесорами. Як правило, кількість доступних процесорів p є суттєво меншою, ніж кількість базових завдань n^2, тобто $p \ll n^2$. Можливий спосіб укрупнення обчислень полягає у використанні стрічкової схеми розбиття матриці W – такий підхід відповідає об'єднанню в рамках однієї базової підзадачі обчислень, пов'язаних з оновленням елементів однієї або декількох

рядків (горизонтальне розбиття) або стовпців (вертикальне розбиття) матриці W . Ці два типи розбиття практично рівноправні, а якщо враховувати додатковий момент, що для алгоритмічної мови С масиви в пам'яті розташовуються по рядках, то далі будемо розглядати тільки розбиття матриці W на горизонтальні смуги.

Слід зазначити, що за такого способу розбиття даних на кожній ітерації алгоритму Флойда-Уоршелла буде потрібно передавати між підзадачами тільки елементи одного з рядків матриці W . Для ефективного виконання подібної комунікаційної операції топологія мережі повинна мати вигляд гіперкуба або повного графа.

Контрольні запитання

1. Наведіть визначення графа. Які основні способи використовуються для задання графів?
2. Який сенс має задача пошуку мінімального кістякового дерева? Наведіть приклад використання задачі на практиці.
3. Наведіть загальну схему алгоритму Прима. Якою є трудомісткість цього алгоритму?
4. В який спосіб можна розпаралелити алгоритм Прима?
5. У чому полягає задача пошуку всіх найкоротших шляхів?
6. Наведіть загальну схему алгоритму Флойда-Уоршелла. Якою є трудомісткість цього алгоритму?
7. В який спосіб можна розпаралелити алгоритм Флойда-Уоршелла?

Індивідуальні завдання

1. Напишіть програму, що реалізує паралельний алгоритм Прима. Проведіть обчислювальні експерименти. Побудуйте теоретичні оцінки з урахуванням параметрів використовуваної обчислювальної системи. Порівняйте отримані оцінки з експериментальними даними.
2. Напишіть програму, що реалізує паралельний алгоритм Флойда-Уоршелла. Проведіть обчислювальні експерименти. Побудуйте теоретичні оцінки з урахуванням параметрів використовуваної обчислювальної системи. Порівняйте отримані оцінки з експериментальними даними.