Лабораторна робота №18. Алгоритми розпаралелення роботи з графами CUDA C/C++

МЕТА РОБОТИ: Розпаралелити пошук найкоротшої відстані за допомогою алгоритму Дейкстри.

3.1. Програма роботи

3.1.1. Отримати завдання.

3.1.2. Написати програми відповідних класів, основну та відповідні допоміжні функції, згідно з вказівками до виконання роботи.

3.1.3. Підготувати власні коректні вхідні дані (вказати їх формат і значення) і проаналізувати їх.

3.1.4. Оформити електронний звіт про роботу та захистити її.

3.2. Вказівки до виконання роботи

3.2.1. Студент, згідно з індивідуальним номером, вибирає своє завдання з розділу 3.4 і записує його до звіту.

3.2.2. Оголошення класу (структури), основну та відповідні допоміжні функції необхідно запрограмувати так, як це показано у розд. 2.4.

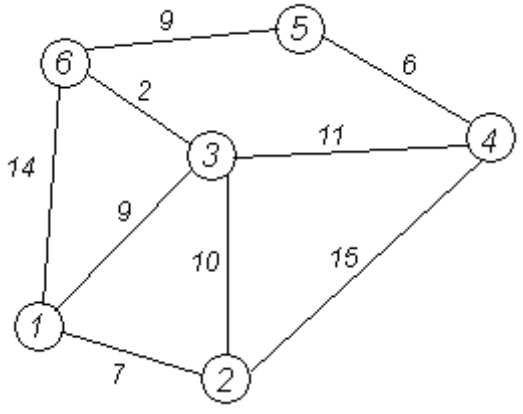
3.2.3. Власні вхідні дані мають бути коректними, знаходитися в розумних межах і відповідати тим умовам, які стосуються індивідуального завдання.

3.2.4. Звіт має містити такі розділи:

- мету роботи та завдання з записаною умовою задачі;
- коди всіх використовуваних .cu файлів, а також пояснення до них;
- результати реалізації програми, які виведені на консоль;
- висновки, в яких наводиться призначення програми, обмеження на її застосування і можливі варіанти удосконалення, якщо такі є.

3.3.Теоретичні відомості

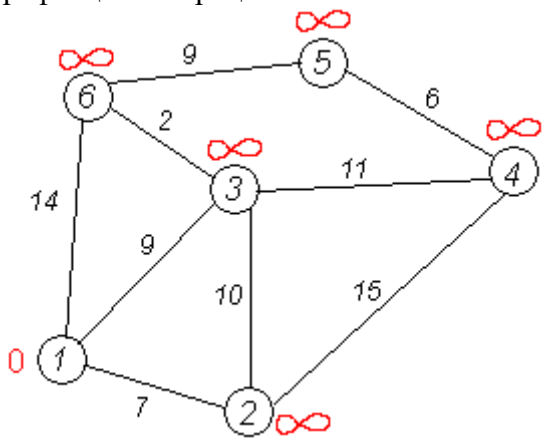
Зберігатимемо поточну мінімальну відстань до всіх вершин **V** (від даної вершини **a**) і на кожному кроці алгоритму намагатимемося зменшити цю відстань. Спочатку встановимо відстані до всіх вершин рівними нескінченості, а до вершини **a** — нулю.



Розглянемо виконання алгоритму на прикладі. Хай потрібно знайти відстані від 1-ї вершини до всіх інших. Кружечками позначені вершини, лініями — шляхи між ними («дуги»). Над дугами позначена їх «ціна» — довжина шляху. Надписом над кружечком позначена поточна найкоротша відстань до вершини.

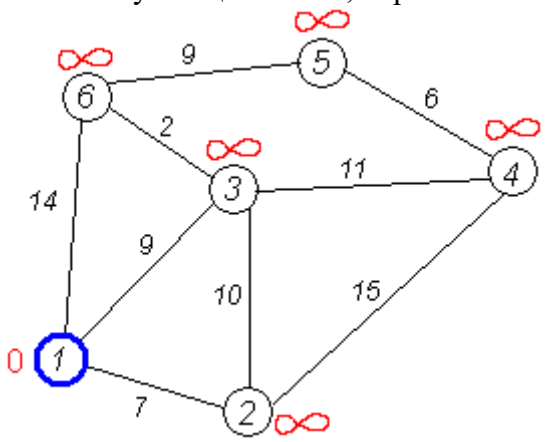
Крок 1

Ініціалізація. Відстань до всіх вершин графа **V** = . Відстань до **a** = 0. Жодна вершина графа ще не опрацьована.



Крок 2

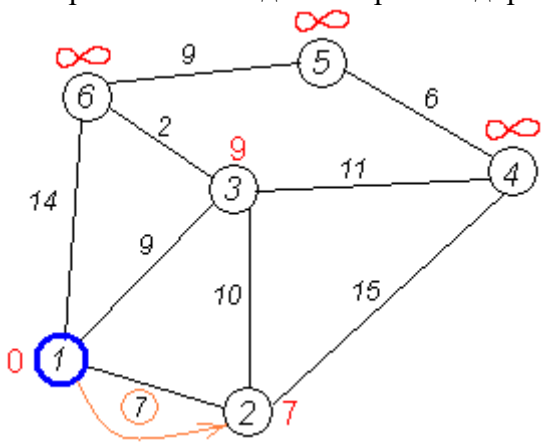
Знаходимо таку вершину (із ще не оброблених), поточна найкоротша відстань до якої мінімальна. В нашому випадку це вершина 1. Обходимо всіх її сусідів і, якщо шлях в сусідню вершину через 1 менший за поточний мінімальний шлях в цю сусідню вершину, то запам'ятовуємо цей новий, коротший шлях як поточний найкоротший шлях до сусіда.



Крок 3

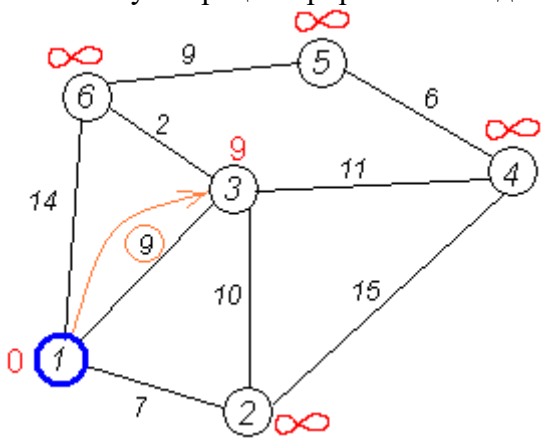
Перший по порядку сусід 1-ї вершини — 2-а вершина. Шлях до неї через 1-у вершину дорівнює найкоротшій відстані до 1-ї вершини + довжина дуги між 1-ю та 2-ю вершиною, тобто 0 + 7 = 7. Це менше поточного найкоротшого шляху до 2-ї вершини, тому

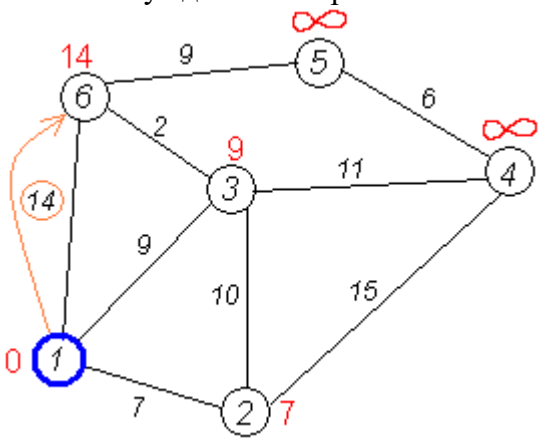
найкоротший шлях до 2-ї вершини дорівнює 7.



Кроки 4, 5

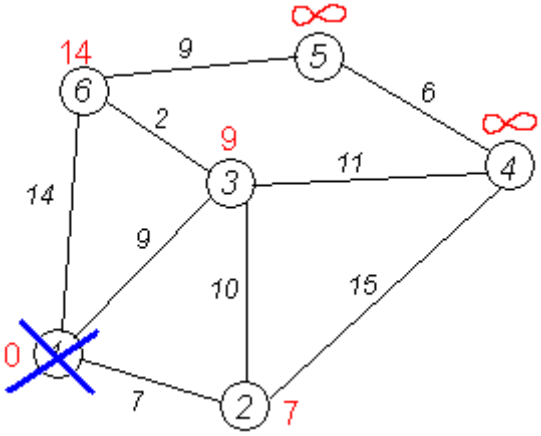
Аналогічну операцію проробляємо з двома іншими сусідами 1-ї вершини — 3-ю та 6-ю.





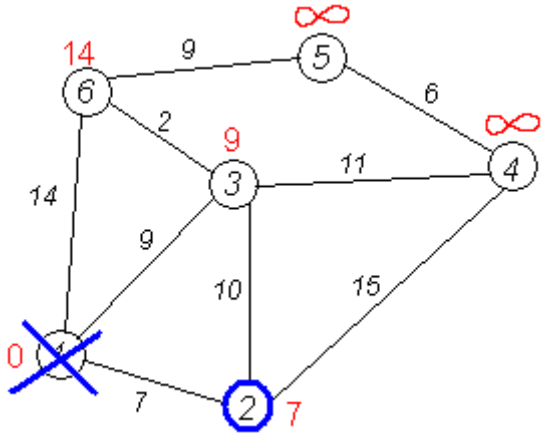
Крок 6

Всі сусіди вершини 1 перевірені. Поточна мінімальна відстань до вершини 1 вважається остаточною і обговоренню не підлягає (те, що це дійсно так, вперше довів Дейкстра). Тому викреслимо її з графа, щоб відмітити цей факт.



Крок 7

Практично відбувається повернення до кроку 2. Знову знаходимо «найближчу» необроблену (невикреслену) вершину. Це вершина 2 з поточною найкоротшою відстанню до неї = 7.



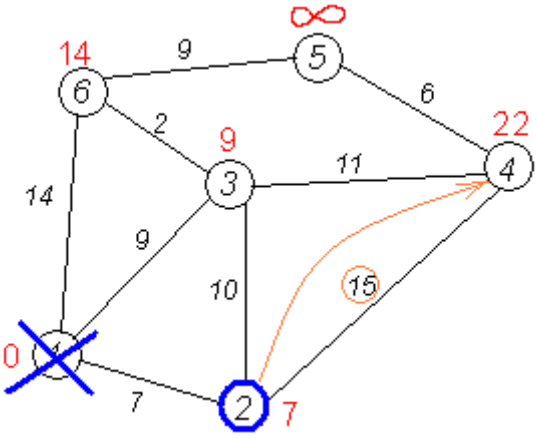
І знову намагаємося зменшити відстань до всіх сусідів 2-ї вершини, намагаючись пройти в них через 2-у. Сусідами 2-ї вершини є 1, 3, 4.

Крок 8

Перший (по порядку) сусід вершини № 2 — 1-ша вершина. Але вона вже оброблена (або викреслена — див. крок 6). Тому з 1-ю вершиною нічого не робимо.

Крок 8 (з іншими вхідними даними)

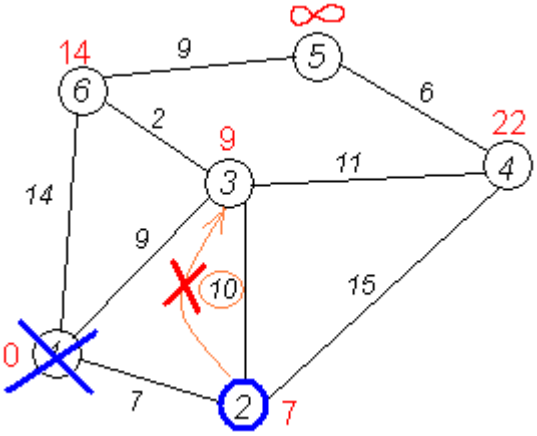
Інший сусід вершини 2 — вершина 4. Якщо йти в неї через 2-у, то шлях буде = найкоротша відстань до 2-ї + відстань між 2-ю і 4-ю вершинами = 7 + 15 = 22. Оскільки 22 < ∞,



встановлюємо відстань до вершини № 4 рівним 22.

Крок 9

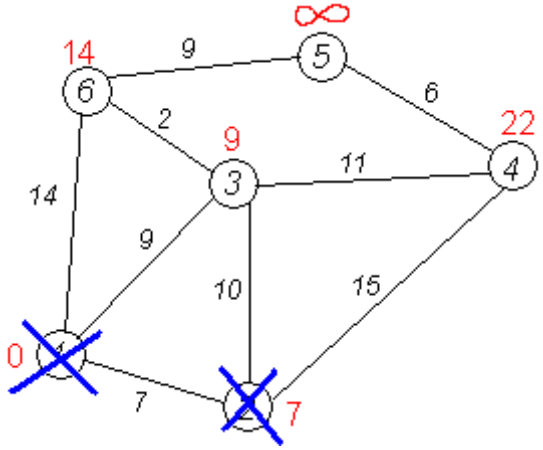
Ще один сусід вершини 2 — вершина 3. Якщо йти в неї через 2-у, то шлях буде = 7 + 10 = 17. Але 17 більше за відстань, що вже запам'ятали раніше до вершини № 3 і дорівнює 9, тому



поточну відстань до 3-ї вершини не міняємо.

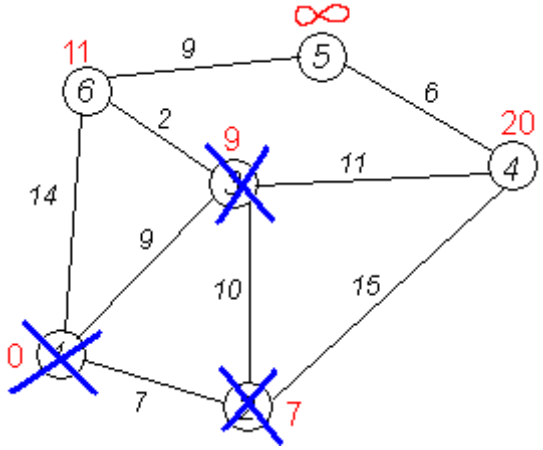
Крок 10

Всі сусіди вершини 2 переглянуті, заморожуємо відстань до неї і викреслюємо її з графа.



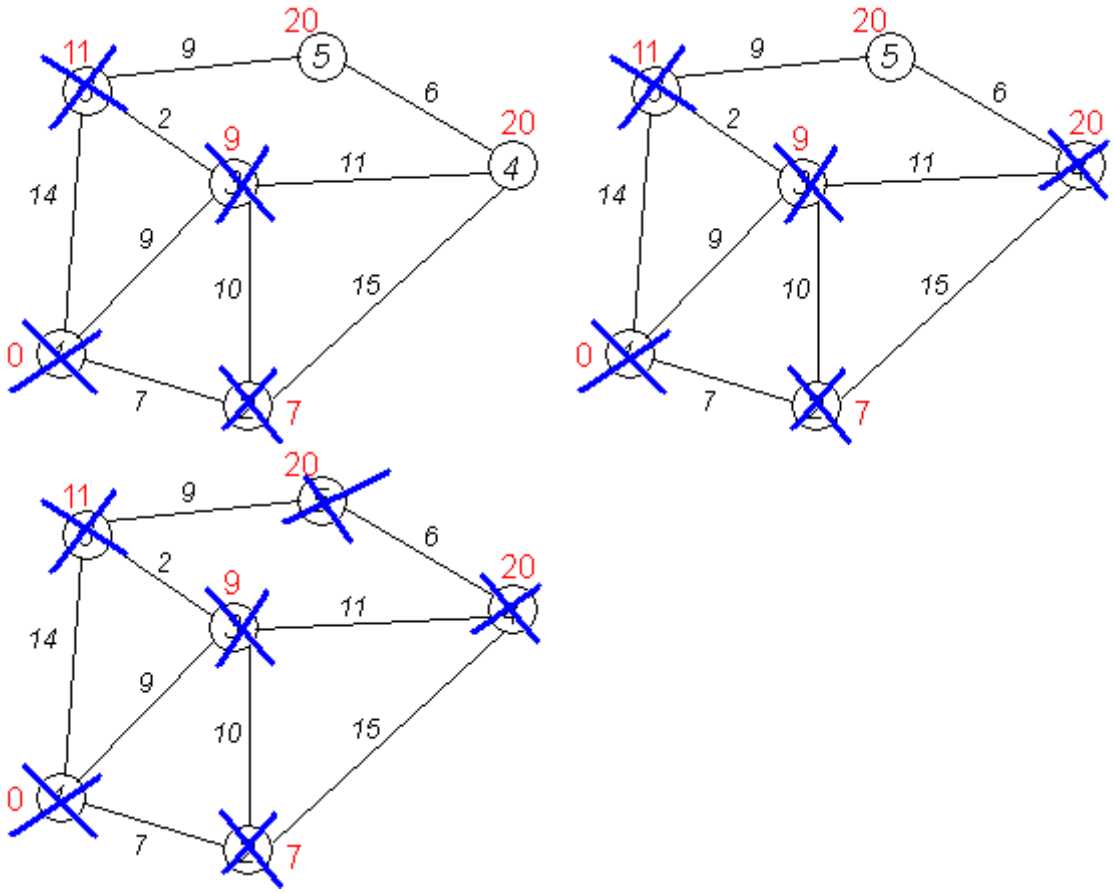
Кроки 11 — 15

По вже «відпрацьованій» схемі повторюємо кроки 2 — 6. Тепер «найближчою» виявляється вершина № 3. Після її «обробки» отримаємо такі результати:



Наступні кроки

Проробляємо те саме з вершинами, що залишилися (№ по порядку: 6, 4 і 5).



Завершення виконання алгоритму

Алгоритм закінчує роботу, коли викреслені всі вершини. Результат його роботи видно на останньому малюнку: найкоротший шлях від 1-ї вершини до 2-ї становить 7, до 3-ї — 9, до 4-ї — 20, до 5-ї — 20, до 6-ї — 11 умовних одиниць.

3.4Індивідуальні завдання

1). Дана мережа автомобільних доріг, що з'єднують міста Львівської області. Знайти найкоротшу відстань від Львова до кожного міста області, якщо рухатись можна тільки по дорогах.

2). Дана карта велосипедних доріжок Польщі та України. Знайти мінімальну відстань, яку треба проїхати, щоб дістатися від Кракова до Києва.

3). Є план міста з нанесеними на нього місцями розміщення пожежних частин. Знайти найближчу до кожного дому пожежну станцію.

4). Дана карта гірських вершин. Знайти відстані до кожної з них враховуючи розташування туристичної групи.

5). Дане розташування футболістів у штрафній площадці суперника. Знайти найкоротший шлях для взяття воріт суперника.

6). Дане розташування шарів на більярдному столі. Знайти найоптимальніший шлях для забиття червоного шару в лузу.

7). Дане розташування патрулів поліції в місті. Знайти відстані до скоєного ДТП.

8). Дана шахматна дошка з розташованими фігурами на ній. Знайти найкоротший шлях для поставлення мату супернику.

9). Дане розташування мезеїв міста Київ. Знайти відстань до кожного з них враховуючи локацію відвідувача.

10). Дано карту України з локацією туриста. Знайти найближчі “дива України”.