Тема 5. Графи

Лекція 3. Знаходження оптимальних маршрутів

1. ***Постановка завдання мінімізації мережі.***
2. ***Алгоритм Краскала.***
3. ***Алгоритм Прима.***
4. ***Алгоритм Прима з підмножинами вершин.***
5. ***Постановка задачі знаходження мінімальної відстані в мережі.***
6. ***Алгоритм Дейкстри для упорядкованого графа.***
7. ***Алгоритм Дейкстри для будь-якого графа.***

1. Постановка завдання мінімізації мережі

Між N віддаленими підмережами (S_1, ..., S_N) МСС необхідно прокласти оптоволоконний кабель. Витрати на прокладку кабелю між підмережами S_i і S_j складають C_{ij} (умов. од.). Всі підмережі рівноправні. Мінімізувати витрати на прокладку кабелю.

Основний алгоритм – знаходження **мінімального кістякового дерева.**

2. Алгоритм Краскала

Візьмемо зважений зв'язний граф G=(V, E), де V — множина вершин, E — множина ребер, для кожного з яких задано вагу.

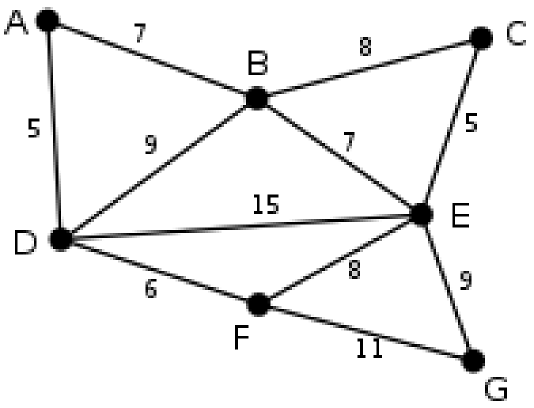
Тоді ациклічна множина ребер, що поєднують усі вершини графу і чия загальна вага мінімальна, називається **мінімальним кістяковим деревом**.

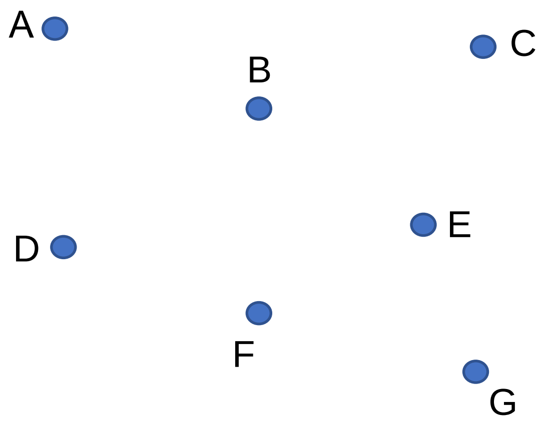
Алгоритм Краскала починається з побудови виродженого лісу, що містить V дерев, кожне з яких складається з однієї вершини.

Далі виконуються операції об'єднання двох дерев (після кроку 0 одне дерево – декілька зв’язаних вершин, друге – ізольована вершина), для чого використовуються найкоротші можливі ребра, поки не утвориться єдине дерево. Ознака закінчення процесу – кількість доданих ребер дорівнює кількості вершин, зменшеної на одиницю.

Це дерево і буде **мінімальним кістяковим деревом**.

Приклад. Розглянемо зважений неорієнтований (7,11)-граф, котрий задає всі можливі варіанти прокладення кабелю між 7 підмережами: A, B, C, D, E, F, G. Яка мінімальна довжина кабелю необхідна?





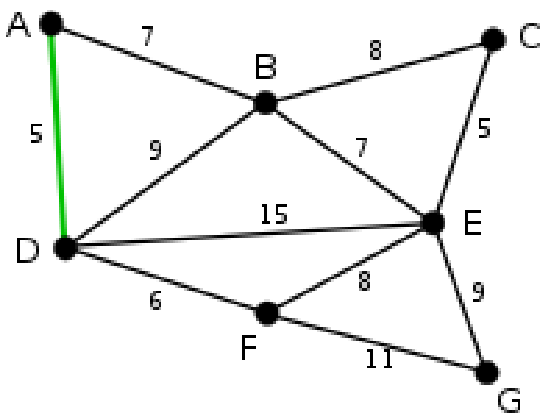
Крок 0. Це початковий граф. Цифри над ребрами позначають їх вагу. Жодне з ребер не додане до кістякового дерева (чорний колір), тому початок для формування кістякового дерева – це вироджений ліс, що складається з 7 незв’язаних вершин.

Поточна довжина кабелю L = 0.

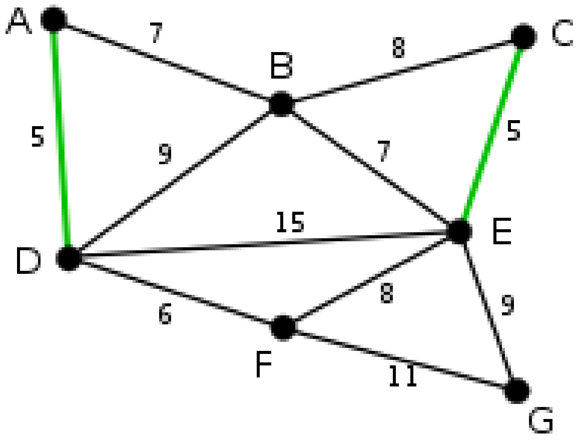
Складемо впорядкований за вагою список ребер:

АД (5)
CE (5)
DF (6)
AB (7)
BE (7)
BC (8)
EF (8)
BD (9)
EG (9)
FG (11)
DE (15)

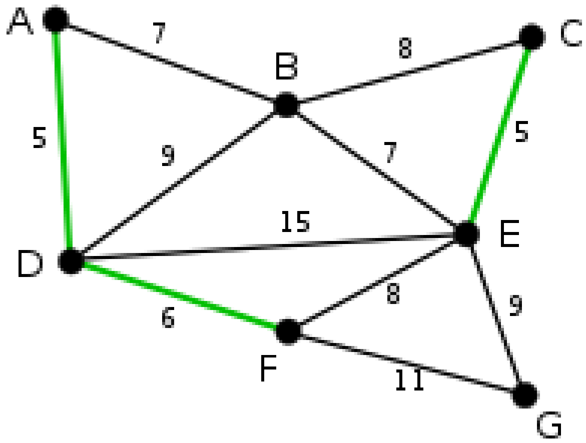
Крок 1. AD і CE мають найменшу вагу 5, і AD вибирається з них довільно та додається до кістякового дерева (зелений колір). L = 0+5=5



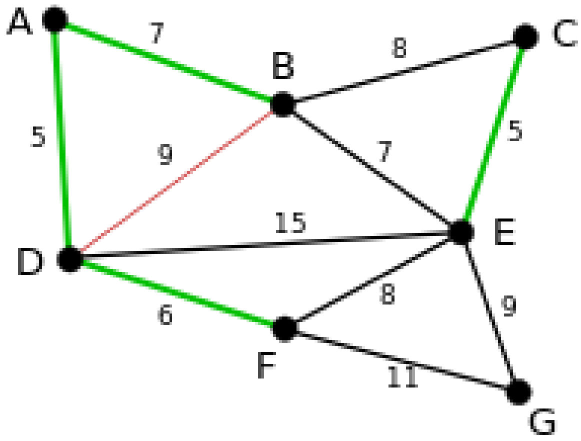
Крок 2. Далі CE є найлегшим ребром з вагою 5, тому воно також додається до дерева. L = 5 + 5 = 10.



Крок 3. Аналогічним чином обирається найлегше з недоданих ребер графу DF з вагою 6 і додається до кістякового дерева, L = 10 + 6 = 16.

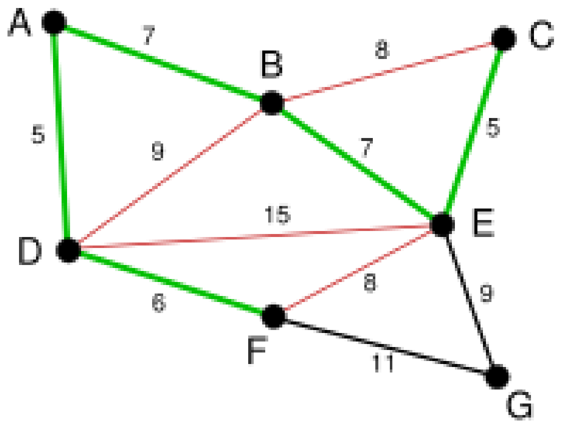


Крок 4. Наступними найлегшими ребрами є AB і BE, вагою 7. AB обирається довільно і додається до кістякового дерева. BD фарбується у червоний колір, оскільки воно є частиною циклу ABD, L = 16 + 7 = 23.

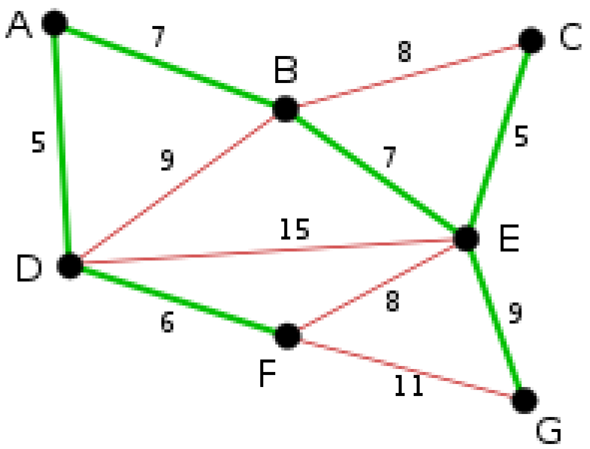


Крок 5. Наступним додається ребро BE з вагою 7. L = 23 + 7 = 30.

Червоним забарвлюємо ребра BC (цикл BCE), DE (цикл DEBA) і FE (цикл FEBAD).



Крок 6. Додаємо ребро EG вагою 9 і отримуємо мінімальне кістякове дерево (це шосте зелене ребро, тому процес закінчено), L = 30 + 9 = 39.



Отже, відповідь: мінімальна довжина кабелю складає **39** одиниць (наприклад, км), а на графі наведений один із можливих варіантів (у даному випадку єдиний).

3. Алгоритм Прима

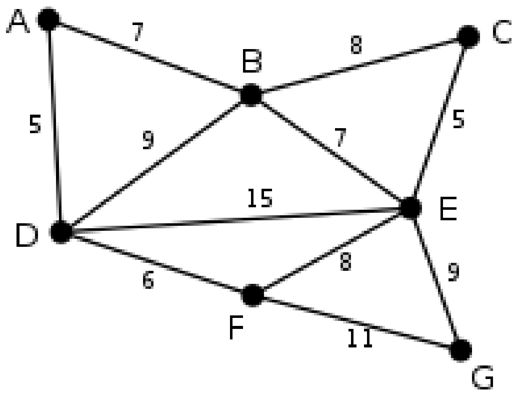
Побудова починається з дерева, що містить в собі одну (довільну) вершину. Протягом роботи алгоритму дерево розростається, поки не охопить усі вершини початкового графу. На кожному кроці алгоритму до поточного дерева приєднується найлегше з ребер, що з'єднують вершину з побудованого дерева і вершину, що не належить дереву.

Виконаємо те ж завдання.

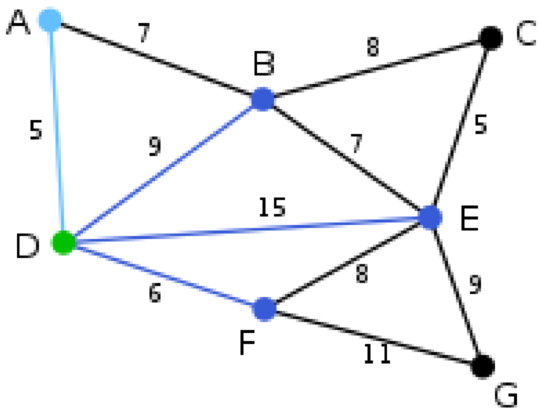
Приклад. Розглянемо зважений неорієнтований (7,11)-граф, котрий задає всі можливі варіанти прокладення кабелю між 7 підмережами: A, B, C, D, E, F, G. Яка мінімальна довжина кабелю необхідна?

Умова. Початковий зважений граф. Числа біля ребер показують їх ваги, які можна розглядати як відстані між вершинами.

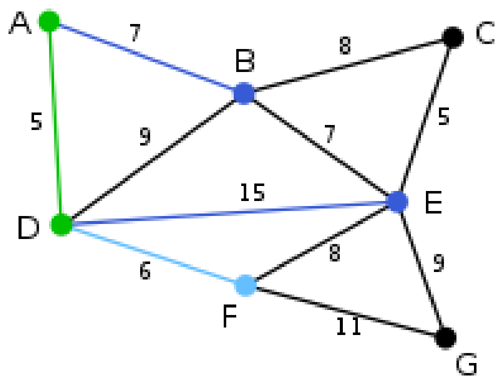
Поточна довжина кабелю L = 0.



Крок 1. За початкову довільно обирають вершину D. Кожна з вершин A, B, E і F з'єднана з D єдиним ребром. Вершина A — найближча до D, і вибирається як друга вершина разом з ребром AD. (блакитний колір – нове ребро, синій колір – ті ребра, що розглядалися). Поточна довжина кабелю L = 5.

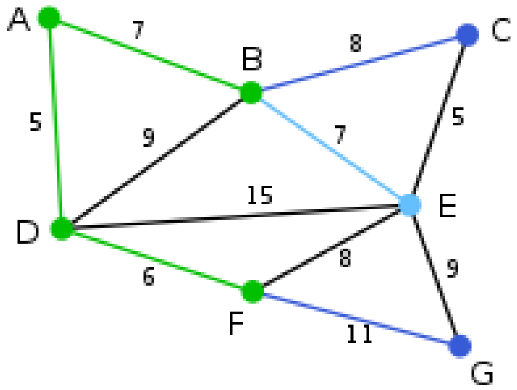


Крок 2. Наступна вершина — найближча до будь-якої з обраних вершин D або A. B віддалена від D на 9 і від A — на 7. Відстань до E дорівнює 15, а до F — 6. F є найближчою вершиною, тому вона включається в дерево разом з ребром DF (зелений колір - елемент кісткового дерева). Поточна довжина кабелю L = 5 + 6 = 11.

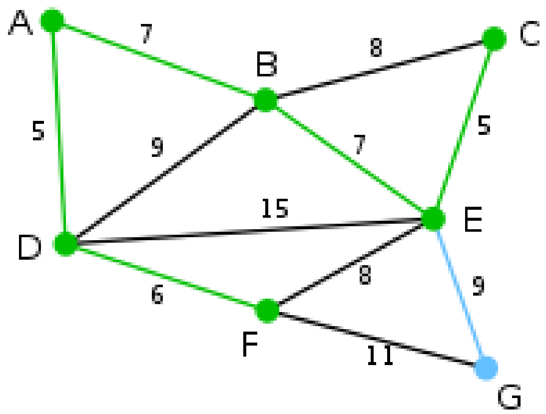


Кроки 3-5. Аналогічними кроками приходимо до такого дерева. У цьому разі є можливість вибрати або C, або E, або G. C віддалена від B на 8, E віддалена від B на

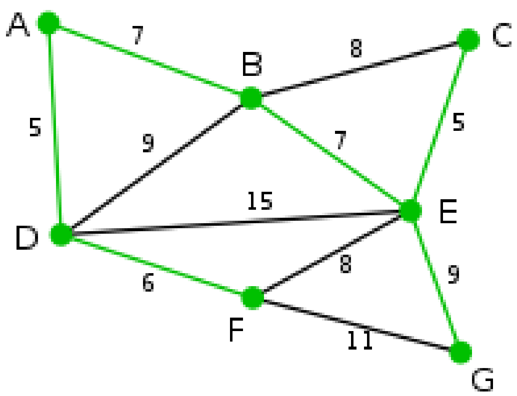
7, а G віддалена від F на 11. E — найближча вершина, тому обирають E і ребро BE. Поточна довжина кабелю L = 11 + 7 + 7 = 25. Потім додаємо вершину C, поточна довжина кабелю L = 25 + 5 = 30.



Крок 6. Єдина вершина, що залишилася — G. Відстань від F до неї одно 11, від E — 9. E ближче, тому обирають вершину G і ребро EG, поточна довжина кабелю L = 30 + 9 = 39.



Крок 7. .Вибрано всі вершини, мінімальне кістякове дерево побудовано



Відповідь: мінімальна довжина кабелю складає **39** одиниць

4. Алгоритм Прима з підмножинами вершин

<u>Крок 1</u>. Визначаємо множину номерів підмереж:
M = Ø – множина під’єднаних підмереж,
B = {1, 2, ..., N} - не під’єднанні вузли (підмережі),
C = Ø - множина вибраних маршрутів прокладки кабелю.

<u>Крок 2</u>. Шукаємо мінімальне ребро:

$$C_{i_0 j_0} = \min_{i,j}\{C_{i,j}\},\ B = B - \{i_0, j_0\};\ k = 2;\ C = \{C_{i_0 j_0}\};\ M = \{i_0, j_0\}$$

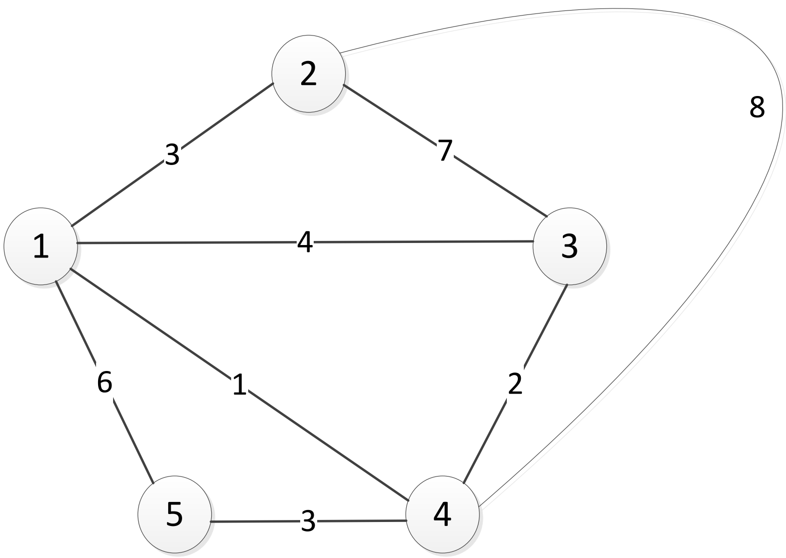
Додаємо його до множини C, а його вершини – до множини M.

<u>Крок 3</u>. (k-я ітерація) Пошук найближчої нерозподіленої вершини графа:

K=k+1, якщо k>N – вихід,

$$C_{i_k j_k} = \min_{\substack{i\in M\\ j\in B}}\{C_{i,j}\}.$$

Результатом є множина мінімальних витрат C: $C_{\min} = \sum\limits_{C_{i,j}\in C} \left|C_{i,j}\right|$.



<u>Приклад.</u>

1) M={1, 4}, B={2, 3, 5}, C={C_{14}}.
2) Між M і B наймінімальніша відстань – 2 (4-3):

M={1,4,3}, B={2,5}, C={C_{14}; C_{43}}.

3) Між M і B наймінімальніша відстань – 3 (4-5; 2-1), обираємо будь-яке:

M={1,4,3,2}, B={5}, C={C_{14}; C_{43}; C_{12}}.

4) Вершина 5 ближче за все до 4, тобто:

M={1,4,3,2,5}, B=Ø, C={C_{14}, C_{43}, C_{12}, C_{45}},

C_{min}=1+2+3+3=9 (од.)

Рішення з таблицею маршрутів: стовпці М, рядки В, шукаємо мінімальні елементи на перетині М і В, далі за алгоритмом:

М / В	1	2	3	4	5
1	x	3	4	1	6
2	3	x	7	8	∞
3	4	7	x	2	∞
4	1	8	2	x	3
5	6	∞	∞	3	x

5. Постановка задачі знаходження мінімальної відстані в мережі

Є N мережних вузлів, пронумерованих від 1 до N. Необхідно передати повідомлення з вузла 1 до вузла N з мінімальними витратами мережевого ресурсу, якщо витрати при передачі з вузла i у вузол j дорівнюють d_{ij} (i <j). Передбачається, що вузли впорядковані і повідомлення може бути передано далі тільки на вузол з великим номером (це можливо, оскільки передбачається мережу без циклів).

Основний алгоритм – **знаходження найкоротшого маршруту.**

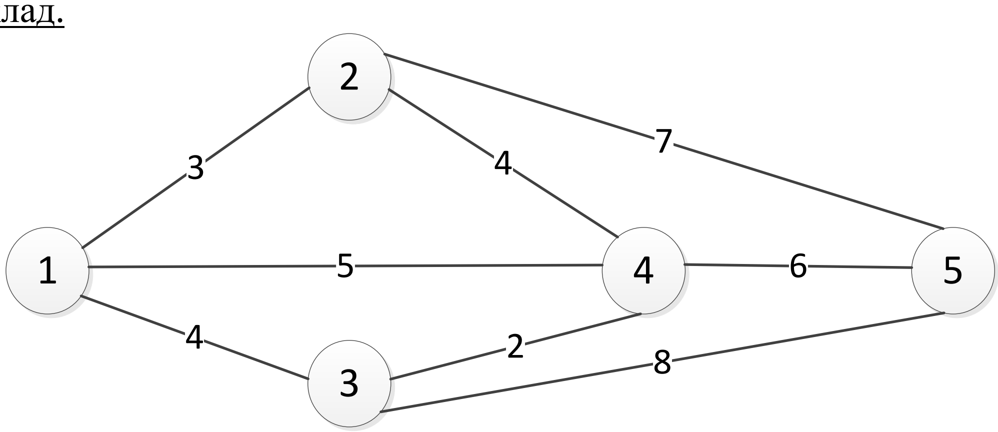
6. Алгоритм Дейкстри для упорядкованого графа

Крок 1. Нехай u_j – довжина мінімального маршруту з вузла 1 у вузол j; j=0; $u_1=0$.

Крок 2. j=j+1. Якщо j>N, то u_N – довжина маршруту, котрий шукаємо.

Крок 3. $u_j = \min\limits_{i<j}\{u_i + d_{ij}\}$

Приклад.



1) $u_1=0$

2) $u_2=\min\{u_1+d_{12}\}=3$

3) $u_3=\min\{u_1+d_{13}\}=4$

4) $u_4=\min\{u_1+d_{14};\ u_2+d_{24};\ u_3+d_{34};\}=\min\{0+5;\ 3+4;\ 4+2\}=5$

5) $u_5=\min\{u_2+d_{25};\ u_3+d_{35};\ u_4+d_{45};\}=\min\{3+7;\ 4+8;\ 5+6\}=10$

Шлях 5→2→1=10 (од.)

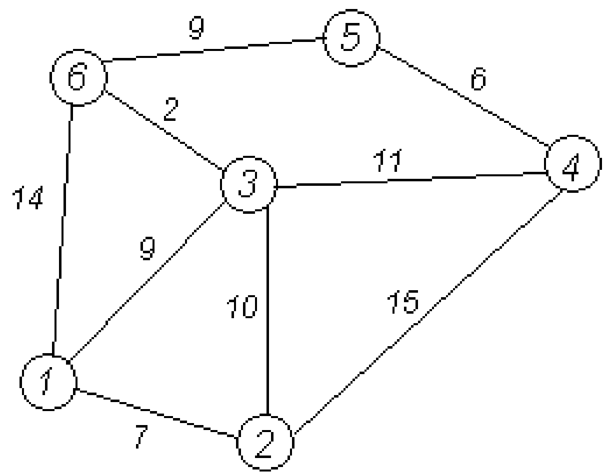
7. Алгоритм Дейкстри для будь-якого графа

Дано неорієнтований зв'язний граф G(V, U). Знайти відстань від вершини а до всіх інших вершин V.

Зберігатимемо поточну мінімальну відстань до всіх вершин V (від даної вершини а) і на кожному кроці алгоритму намагатимемося зменшити цю відстань.

Спочатку встановимо відстані до всіх вершин рівними нескінченості, а до вершини а — нулю.

Розглянемо виконання алгоритму на прикладі.



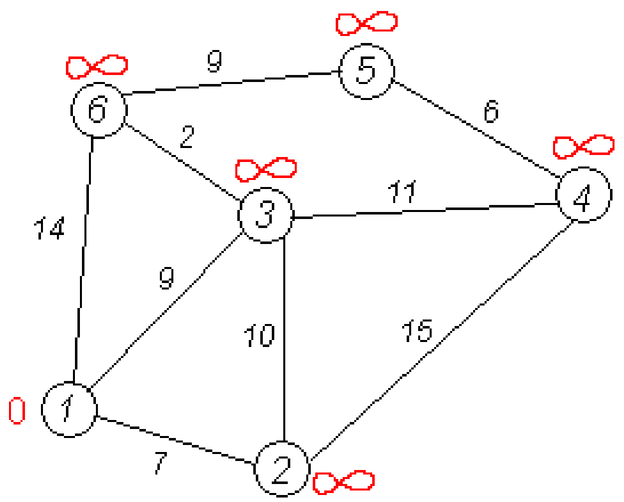
Нехай потрібно знайти відстані від 1-ї вершини до всіх інших.

Кружечками позначені вершини, лініями — шляхи між ними («дуги»).

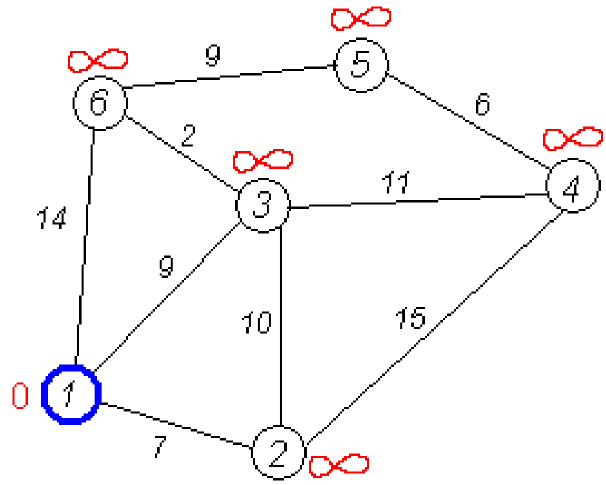
Над дугами позначена їх «ціна» — довжина шляху. Надписом над кружечком позначена поточна найкоротша відстань до вершини.

Крок 1. Ініціалізація.

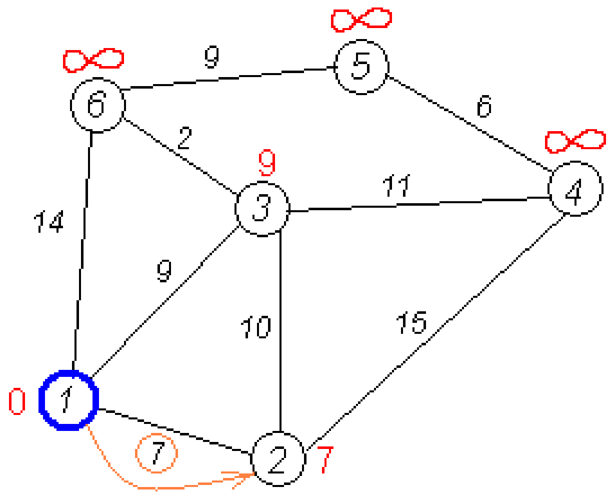
Встановлюємо відстані до всіх вершин рівними нескінченості, а до вершини а — нулю. Жодної вершини графу ще не опрацьовано.



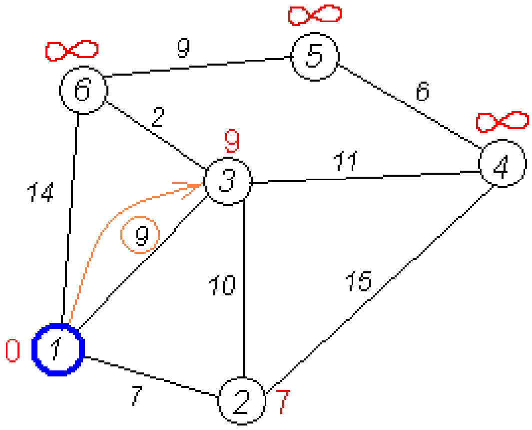
Крок 2.
Знаходимо таку вершину (із ще не опрацьованих), поточна найкоротша відстань до якої мінімальна. В нашому випадку це вершина **1**.
Починаємо її опрацьовувати (відмічаємо синім кружечком).
Для цього будемо обходити всіх її сусідів і, якщо шлях в сусідню вершину через **1** менший за поточний мінімальний шлях в цю сусідню вершину, то запам'ятовуємо цей новий, коротший шлях як поточний найкоротший шлях до сусіда.

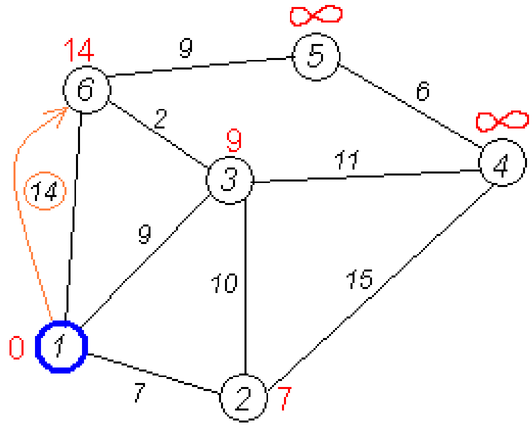


Крок 3.
Перший по порядку сусід 1-ї вершини — 2-а вершина. Шлях до неї через 1-у вершину дорівнює найкоротшій відстані до 1-ї вершини + довжина дуги між 1-ю та 2-ю вершиною, тобто 0 + 7 = 7 (червона стрілочка). Це менше поточного найкоротшого шляху до 2-ї вершини, тому найкоротший шлях до 2-ї вершини дорівнює 7.

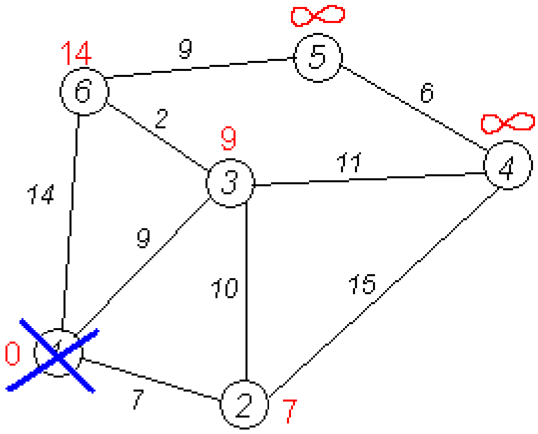


Кроки 4, 5.
Аналогічну операцію проробляємо з 2 іншими сусідами 1 вершини — 3-ю та 6-ю.

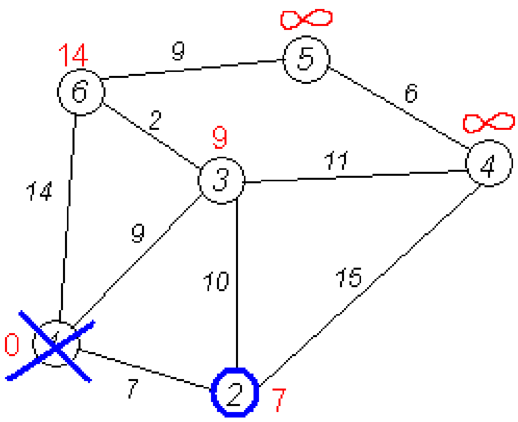




Крок 6
Всі сусіди вершини 1 перевірені. Поточна мінімальна відстань до вершини 1 вважається остаточною і обговоренню не підлягає (те, що це дійсно так, вперше довів Дейкстра). Тому викреслимо її з графу, щоб відмітити цей факт.



Крок 7.
Практично відбувається повернення до кроку 2. Знову знаходимо «найближчу» необроблену (невикреслену) вершину. Це вершина 2 з поточною найкоротшою відстанню до неї = 7. Обводимо її синім кружечком і починаємо опрацьовувати її сусідів.

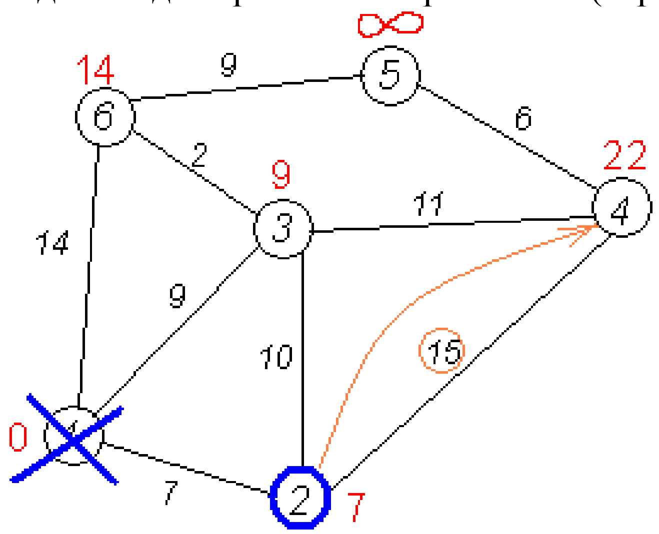


І знову намагаємося зменшити відстань до всіх сусідів 2-ї вершини, намагаючись пройти в них через 2-у. Сусідами 2-ї вершини є 1, 3, 4.

Крок 8

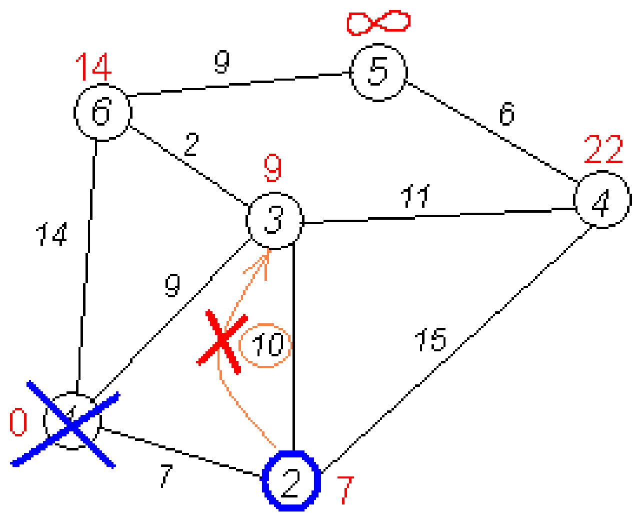
Перший (по порядку) сусід вершини № 2 — 1-ша вершина. Але вона вже оброблена (або викреслена на кроці 6). Тому з 1-ю вершиною нічого не робимо.

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 (червона стрілка).



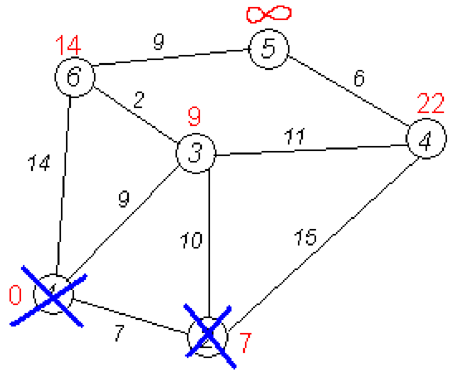
Крок 9

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 (закреслена червона стрілка до неї).



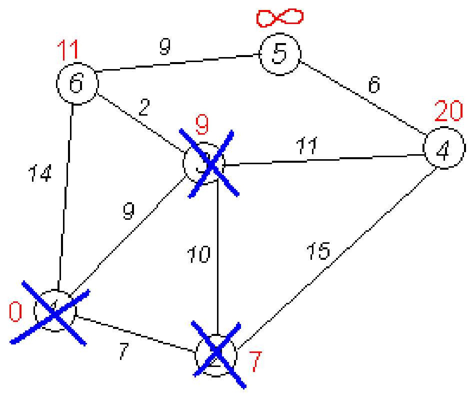
Крок 10

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



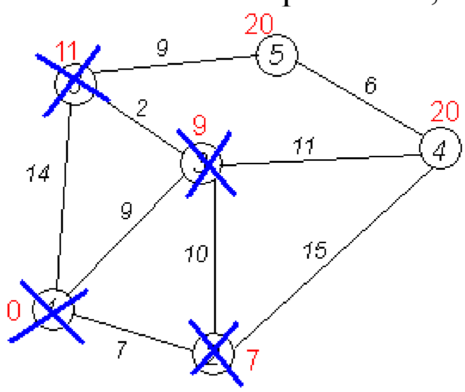
Кроки 11 — 15

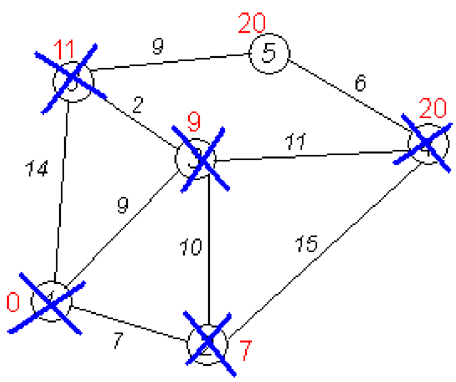
По вже «відпрацьованій» схемі повторюємо кроки 2 — 6. Тепер «найближчою» виявляється вершина № 3. Після її «обробки» отримаємо такі результати:

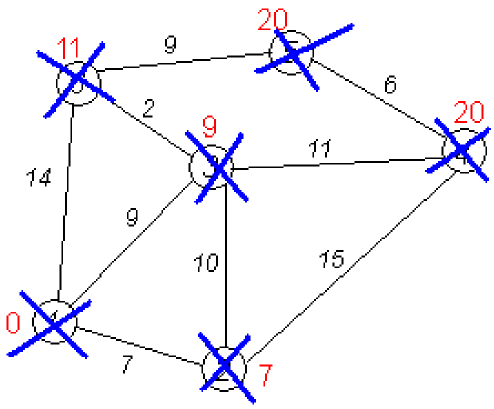


Наступні кроки

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 рисунку: 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.