Лекція 29. Алгоритм Белмана-Форда.

Довжиною маршруту називають суму довжин (ваг) ребер, які входять у маршрут.

Задача: знайти найкоротший маршрут між двома заданими (фіксованими) вершинами графа.

Щоб знайти найкоротший маршрут від вершини *x* до вершини *y*, треба знайти найкоротші маршрути від вершини *x* до вершин, які сусідні (суміжні) з вершиною *y*. Тоді слід розглядати сусідів сусідів і т.д. У результаті спочатку будуємо найкоротші маршрути від вершини *x* до всіх вершин графа, а тоді вибираємо той маршрут, який нас цікавить.

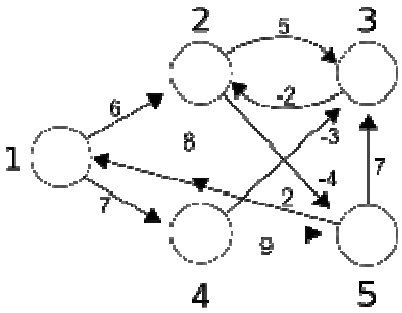
Алгоритм Белмана-Форда дозволяє знайти найкоротші маршрути від заданої (фіксованої) вершини графа до всіх решта вершин графа. Обчислювальна складність вказаного алгоритму дорівнює $O(|V|\cdot|E|)$. Алгоритм Белмана-Форда ґрунтується на наведеному далі принципі оптимальності.

Принцип оптимальності Белмана (динамічного програмування): будь-який підмаршрут оптимального маршруту є оптимальним.

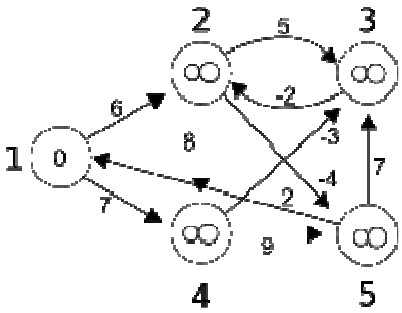
Дозволені ребра з від'ємною вагою, але заборонені цикли з від'ємною довжиною. Всі маршрути ведемо паралельно; частину з них по ходу обриваємо. Якщо граф має *n* вершин, то результат отримуємо за *n*-1 кроки.

Далі наведено приклад роботи описаного алгоритму на скінченому орієнтованому графі з 5 вершинами. Потрібно знайти найкоротші маршрути з вершини з номером 1 до вершин з номерами 2, 3, 4, 5. Зрозуміло, що для цього слід виконати 4 кроки. Після 4-го кроку у вершинах 2, 3, 4, 5 маємо числа, які дорівнюють довжинам найкоротших маршрутів від фіксованої вершини 1 до відповідно вершин 2, 3, 4, 5.

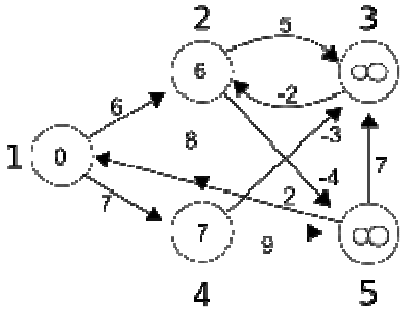
Для того, щоб отримати не тільки довжини найкоротших маршрутів, а й самі найкоротші маршрути, треба зберігати матрицю з інформацією на кожному кроці, яка вершина є попередньою до кожної вершини. Так, довжина найкоротшого маршруту від вершини 1 до вершини 2 дорівнює 2. Він містить ребра (1, 4), (4, 3), (3, 2).



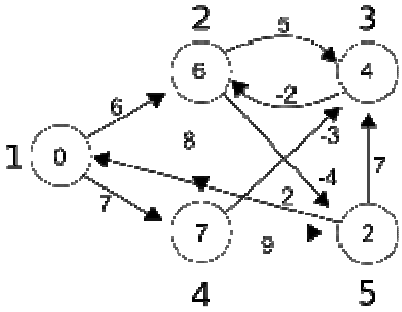
Початковий граф



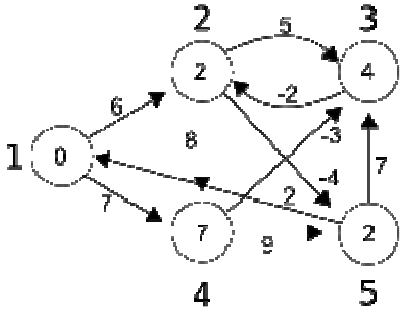
Початковий граф з мітками



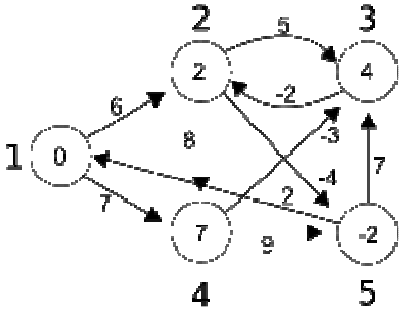
1-й крок роботи



2-й крок роботи



3-й крок роботи



4-й крок роботи

Алгоритм можна застосовувати і для неорієнтованих графів. Для цього кожне неорієнтоване ребро слід замінити на два орієнтованих ребра протилежних напрямків. Ваги обидвох орієнтованих ребер дорівнюють вазі початкового неорієнтованого ребра.

Можливі прикладні застосування: розпізнавання мови, завадостійке кодування (код Вітербі).