Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Алгоритми та складність

Завдання № 9

“ Алгоритм Джонсона ”

Виконав студент 2-го курсу

Групи К-29

Демиденко Андрій Геннадійович

2021

**Предметна область**

Предметна область: Відділ кадрів

Об’єкти: Підрозділи, Співробітники

Примітка: Існує множина підрозділів. Кожен підрозділ включає в себе множину співробітників.

**Завдання**

Алгоритм Джонсона для розріджених графів (включає алгоритми Беллмана-Форда і Дейкстри). В алгоритмі Дейкстри використайте піраміду Фібоначчі

**Теорія**

Алгоритм Джонсона дозволяє знайти найкоротші шляхи між усіма парами вершин зваженого орієнтованого графа. Цей алгоритм працює, якщо у графі містяться ребра здодатною чи від'ємною вагою, але відсутні цикли з від'ємною вагою.

В алгоритмі Джонсона використовують алгоритм Дейкстри  та алгоритм Дейкстри втілені у вигляді підпрограм. Ребра зберігають у вигляді переліків суміжних вершин. Алгоритм повертає звичайну матрицю   {\displaystyle D=d\_{ij}} розміром {\displaystyle |V|\times |V|}  {\displaystyle d\_{ij}=\delta (i,\;j)}або видає повідомлення про те, що вхідний граф містить цикл із від'ємною вагою.

Алгоритм Дейкстри— алгоритм на графах, відкритий Дейкстрою. Знаходить найкоротшийшлях від однієї вершини графа до всіх інших вершин. Класичний алгоритм Дейкстри працюєтільки для графів без циклів від'ємної довжин.

1. Кожній вершині з V зіставимо мітку - мінімальне відоме відстань від цієї вершини до a.

Алгоритм працює покроково - на кожному кроці він «відвідує» одну вершину і намагається зменшувати мітки.

Робота алгоритму завершується, коли всі вершини відвідані.

1. Ініціалізація.

Мітка самої вершини a покладається рівною 0, мітки інших вершин - нескінченності.

Це відображає те, що відстані від a до інших вершин поки невідомі.

Всі вершини графа позначаються як невідвіданих.

1. Крок алгоритму Декстри :

Якщо все вершини відвідані, алгоритм завершується.

В іншому випадку, з ще не відвіданих вершин вибирається вершина u, що має мінімальну позначку.

Ми розглядаємо різні маршрути, в яких u є передостаннім пунктом. Вершини, в які ведуть ребра з u, назвемо сусідами цієї вершини. Для кожного сусіда вершини u, крім позначених відвідані, розглянемо нову довжину шляху, що дорівнює сумі значень поточної мітки u і довжини ребра, що з'єднує u з цим сусідом.

Якщо отримане значення довжини менше значення мітки сусіда, замінимо значення мітки отриманим значенням довжини. Розглянувши всіх сусідів, позначимо вершину u як відвіданих і повторимо крок алгоритму.

Усі невідвідані вершини графу зберігаються у Піраміді Фібоначчі, яка з кажним кроком алгоритму зменшуються тобто в кожному кроці алгоритму дейкстри з неї видаляються відвідані вершини, і так як видалленя з піраміди Фібоначчі має середню складність то час роботи алгоритму Дейкстри зменшується з що і зменшує складність самого алгоритму Джонсона

Алгоритм Беллмана—Форда — алгоритм пошуку найкоротшого шляху взваженому графі.Знаходить найкоротші шляхи від однієї вершини графа до всіх інших. На відміну від алгоритмуДейкстри, алгоритм Беллмана—Форда допускає ребра з негативною вагою. Запропонованонезалежно Річардом Беллманом і Лестером Фордом

Для знаходження найкоротших шляхів від однієї вершини до всіх інших, скористаємося методом динамічного програмування.

**Крок 1**

На цьому кроці не започатковано відстані від вихідної вершини до всіх інших вершин, як нескінченні, а відстань до самого src приймається рівним 0. Створюється масив dist [] розміру | V | з усіма значеннями рівними нескінченності, за винятком елемента dist [src], де src - вихідна вершина.

**Крок 2**

Другим кроком обчислюються найкоротші відстані. Наступні кроки потрібно виконувати | V | -1 раз, де | V | - число вершин в даному графі.

Проведіть наступна дія для кожного ребра u-v:

Якщо dist [v]> dist [u] + вага ребра uv, то поновіть dist [v]

dist [v] = dist [u] + вага ребра uv

**Крок 3**

На цьому кроці повідомляється, чи присутній в графі цикл негативного ваги. Для кожного ребра u-v необхідно виконати наступне:

Якщо dist [v]> dist [u] + вага ребра uv, то в графі присутній цикл негативного ваги.

**Алгоритм**

* Перевірка графа на від’ємні цикли ,у разі виявлення алгоритм

Повертає інформацію про те що алгоритм неможливий завершує свою робототу

* Спочатку до графу додається новий вузол q, пов'язаний ребрами з нульовим вагою з кожним з інших вузлів
* По-друге, алгоритм Беллмана - Форда використовується, починаючи з нової вершини q, для знаходження для кожної вершини v мінімальної ваги h (v) шляху з q в v.
* Потім ребра вихідного графа повторно зважуються з використанням значень, обчислених алгоритмом Беллмана - Форда: ребру від u до v, має довжину, дається нова довжина

w (u, v) + h (u) - h (v).

* Нарешті, q видаляється, і алгоритм Дейкстри використовується для пошуку найкоротших шляхів від кожного вузла s до кожної іншої вершини в переглянутому графі. Відстань у вихідному графі потім обчислюється для кожної відстані D (u, v) шляхом додавання h (v) - h (u) до відстані, що повертається алгоритмом Дейкстри

**Складність**

Якщо в алгоритмі Дейкстри неспадну чергу з пріоритетами втілено у вигляді піраміди Фібоначі, то тривалість роботи алгоритму Джонсона дорівнює (  )

**Мова програмування**

С++

**Модулі програми**

distanceVector dijkstra(GraphFunctionality& graph, edgesContainer newEdgesWeight, size\_t fromVertex) {//функція викнує алгоритм Дейкстри

void relax(distanceVector& dist, GraphFunctionality& graph, size\_t u, size\_t v) {//робить релаксацію ребра тобто виключення з циклу

distanceVector belmanFord(GraphFunctionality& graph, size\_t fromVertex) throw(runtime\_error) {//функція виконує алгоритм Белмана-Форда

vector<distanceVector> johnsonAlgorithm(GraphFunctionality& g) {//функція виконує алгоритм Джонсона

pair<size\_t, int> findMin(vector<bool>& in, edgesContainer& edgesWeight) {//функція пошуку мінімального ребра

void initDistanceVector(distanceVector& distance, size\_t fromVertex) {//допоміжна функція знаходить довжину маршруту

class Graph //клас для опису графу

class car //клас для опису працівника

**Інтерфейс користувача**

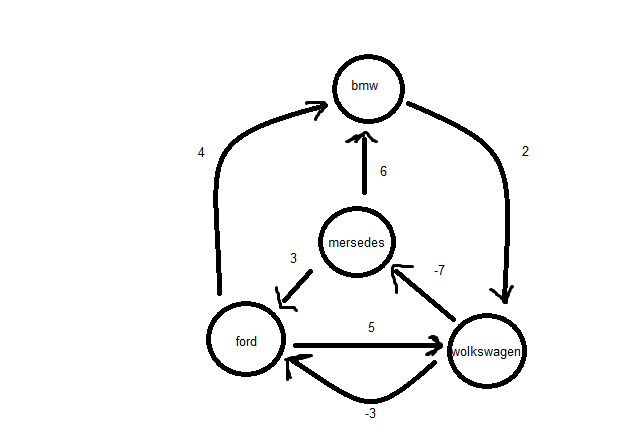
Вхідні дані зроблені у вигляді юніт-тесту а вихідні виводяться у консоль.

**Приклад виводу програми**

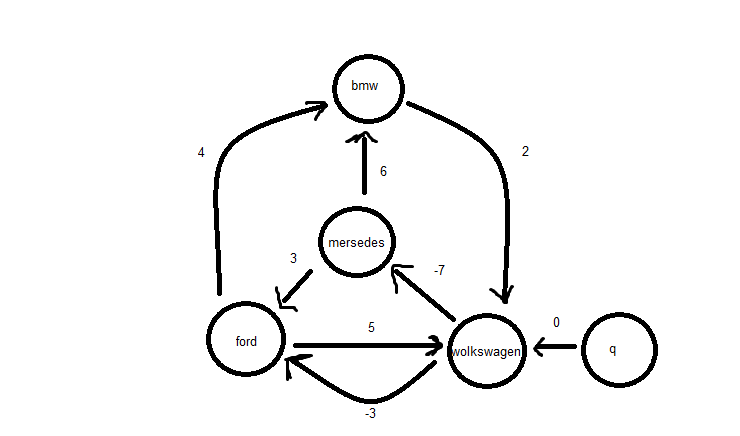
****

**Тестовий приклад**

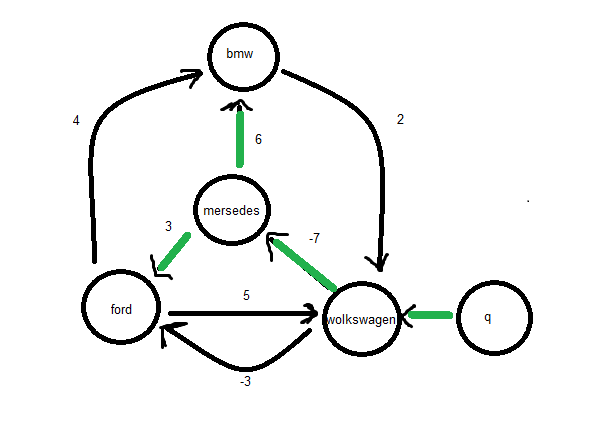
Нехай маємо множину відділу кадрів різних підрозділів взаємне положення яких можна подати у вигляді орієнтованого графу

****

Додаємо уявну вершину q



Шукаємо для всіх вершин найкоротші шляхи до q за Алгоритмом Белмана-Форда



h(ford) =3+(-7) = -4

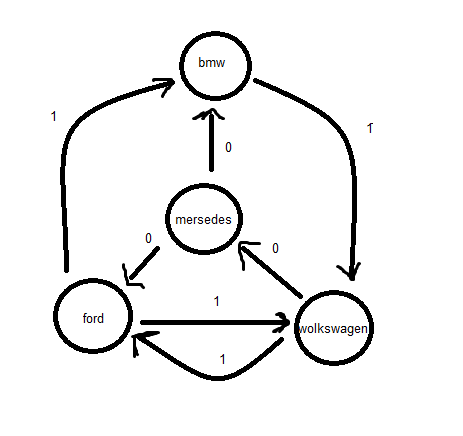
h(wolkswagen)=0

h(mersedes)=-7

h(bmw)=6+(-7)= -1

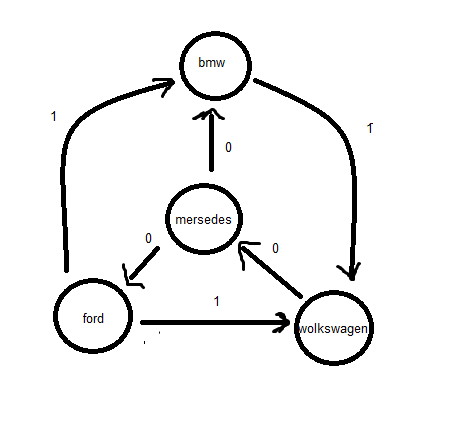
далі для кожного ребра w ( u , v ) даєм йому нове значення w ( u , v ) + h ( u ) - h ( v )

Далі граф має вигляд з новими вагами

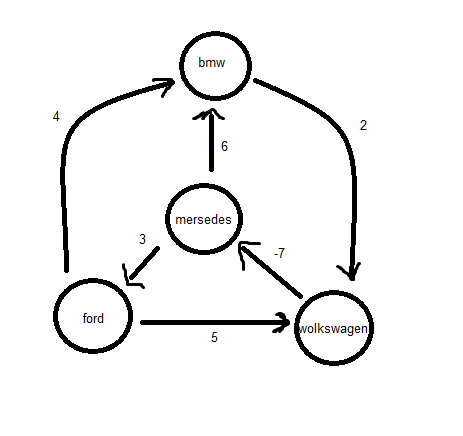


Застосовуємо алгоритм Дейкстри для цього графу

Вихідний граф



Повертаємо ребрам попередні значення



Це остаточний граф з мінмальним маршрутами і Алгоритм завершив свою роботу

**Висновок:**

Так як ми для зберігання невідвіданих вершин використовували Фібоначієву купу то ми значно зменшили складність алгоритму з тої яка б була при наївній реалізації.При розріджених графах складність алгоритму стає меншою чи складність алгоритму Флойда-Уоршала який виконує цю задачу за

**Література:**

* <https://habr.com/ru/company/otus/blog/484382/>
* <https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D1%8B>
* Кормен,Лайзерсон Алгоритми, построєние и анализ ст. 739

{\displaystyle O(V^{2}\log V+VE)}

