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

Жутовський Глiб Євгенович

2021

**Предметна область**

Предметна область: Навчальний відділ

Об’єкти:  Групи, Студенти

Примітка: Існує множина навчальних груп. Кожна група включає в себе множину студентів.

**Завдання**

Реалізувати 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 з додатною чи від'ємною вагою, але відсутні цикли з від'ємною вагою. В алгоритмі Джонсона використовують алгоритм Беллмана-Форда та алгоритм Дейкстри втілені у вигляді підпрограм. Ребра зберігають у вигляді переліків суміжних вершин. Алгоритм повертає звичайну матрицю  розміром або видає повідомлення про те, що вхідний граф містить цикл із від'ємною вагою.

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

Кроки алгоритму Дейкстри:

1. 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.
2. Якщо все вершини відвідані, алгоритм завершується.
3. В іншому випадку, з ще не відвіданих вершин вибирається вершина u, що має мінімальну позначку.
4. 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.
5. 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 кожним кроком алгоритму зменшуються, тобто в кожному кроці алгоритму Дейкстри з неї видаляються відвідані вершини, і так як видалення 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 в зваженому графі. Знаходить найкоротші шляхи від однієї вершини графа до всіх інших. На відміну від алгоритму Дейкстри, алгоритм Беллмана-Форда допускає ребра з негативною вагою. Запропоновано незалежно Річардом Беллманом і Лестером Фордом

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

1. На цьому кроці не започатковано відстані від вихідної вершини до всіх інших вершин, як нескінченні, а відстань до самого current приймається рівним 0. Створюється масив dist [] розміру | V | з усіма значеннями рівними нескінченності, за винятком елемента dist [current], де current - вихідна вершина.
2. Другим кроком обчислюються найкоротші відстані. Наступні кроки потрібно виконувати | V | -1 раз, де | V | - число вершин в даному графі. Проведіть наступну дію для кожного ребра u-v: Якщо dist [v] > dist [u] + вага ребра u-v, то поновіть dist [v]: dist [v] = dist [u] + вага ребра u-v.
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 u-v, 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 і завершує свою роботу

1. Додаємо новий вузол q, пов'язаний ребрами з нульовим вагою з кожним з інших вузлів.
2. Алгоритм Беллмана - Форда використовується, починаючи з нової вершини q, для знаходження для кожної вершини v мінімальної ваги h (v) шляху з q в v.
3. Ребра вихідного графа повторно зважуються з використанням значень, обчислених алгоритмом Беллмана - Форда: ребру від u до v, має довжину, дається нова довжина .
4. 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 шляхом додавання 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**

class Student - для опису студентів.

class Group – для опису груп.

class Graph – для опису графу(загального).

class GraphMatrix – опис графу через матрицю суміжностей.

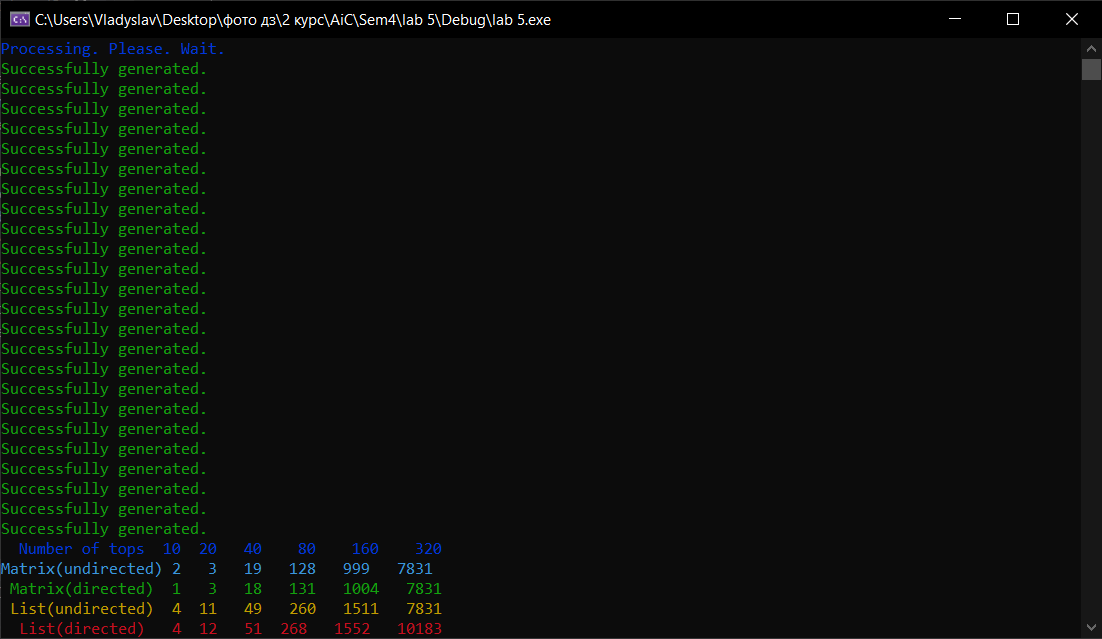
class GraphList – опис графу через список суміжностей.

void Menu() – головне меню програми.

**Інтерфейс користувача**

Програма працює наступним чином: користувачу дається на вибір кілька варіантів роботи програми. Це перевірка ефективності програми, демонстративний режим і інтерактивний режим.

**Приклад виводу програми**



**Тестовий приклад**

Нехай маємо множину кабінетів, де займаються групи, взаємне положення яких можна подати у вигляді орієнтованого графу.

6

4

A

2

C

D

B

-7

3

5

-3

6

4

A

2

C

D

B

-7

3

5

-3

q

0

Додаємо уявну вершину q

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

6

4

A

2

C

D

B

-7

3

5

-3

q

0

далі для кожного ребра даємо йому нове значення

Застосовуємо алгоритм Дейкстри для цього графу

0

1

A

1

C

D

B

0

0

1

0

1

A

1

C

D

B

0

0

1

1

Вихідний граф

Повертаємо ребрам попередні значення

6

4

A

2

C

D

B

-7

3

5

Це остаточний граф з мінімальними маршрутами і алгоритм завершив свою роботу.

**Висновок**

Так як ми для зберігання невідвіданих вершин використовували Фібоначеву купу, то ми значно зменшили складність алгоритму з тої, яка б була при наївній реалізації. При розріджених графах складність алгоритму стає меншою.

**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)}