ДОДАТОК

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота №5

з дисципліни
«Дискретна математика»

Виконала:
студент групи КН-114
Ярка Ірина
Викладач:
Мельникова Н.І.

Львів – 2019 р.

Знаходження найкоротшого маршруту за алгоритмом Дейкстри. Плоскі планарні графи

Мета роботи: набуття практичних вмінь та навичок з використання алгоритму Дейкстри.

Теоретичні відомості

Задача знаходження найкоротшого шляху з одним джерелом полягає у знаходженні найкоротших(мається на увазі найоптимальніших за вагою) шляхів від деякої вершини(джерела) до всіх вершин графа G. Для розв'язку цієї задачі використовується «жадібний» алгоритм, який називається алгоритмом Дейкстри.

«Жадібними» називаються алгоритми, які на кожному кроці вибирають оптимальний із можливих варіантів.

1-й крок індукції. Нехай зафіксовано вершину x_0, E_1 – множина усіх ребер $e \in E$, інцидентних v_0. Серед ребер $e \in E_1$ вибираємо ребро $e(1) = (v_0, v_1)$, що має мінімальну вагу, тобто $w(e(1)) = \min_{e \in E_1} w(e)$. Тоді v_1 називаємо першою найближчою вершиною (НВ), число $w(e(1))$ позначаємо $l(1) = l(v_1)$ і називаємо відстанню до цієї НВ. Позначимо $V_1 = \{v_0, v_1\}$ – множину найближчих вершин.

2-й крок індукції. Позначимо E_2 – множину усіх ребер $e=(v', v'')$, $e \in E$, таких що $v' \in V_1$, $v'' \in (V \setminus V_1)$. Найближчим вершинам $v \in V_1$ приписано відстані $l(v)$ до кореня v_0, причому $l(v_0)=0$. Введемо позначення: $\overline{V_1}$ – множина таких вершин $v'' \in (V \setminus V_1)$, що $\exists$ ребра виду $e = (v, v'')$, де $v \in V_1$. Для всіх ребер $e \in E_2$ знаходимо таке ребро $e_2=(v', v_2)$, що величина $l(v')+w(e_2)$ найменша. Тоді v_2 називається другою найближчою вершиною, а ребра e_1, e_2 утворюють зростаюче дерево для виділених найближчих вершин $D_2 = \{e_1, e_2\}$.

(s+1)-й крок індукції. Нехай у результаті s кроків виділено множину найближчих вершин $Vs = \{v_0, v_1, \ldots, v_s\}$ і відповідне їй зростаюче дерево $D_s = \{e_1, e_2, \ldots, e_s\}$... Для кожної вершини $v \in V_s$ обчислена відстань $l(v)$ від кореня v_0 до v; $\overline{V_s}$ – множина вершин $v \in (V \setminus V_s)$, для яких існують ребра вигляду $e = (v_r, v)$, де $v_r \in V_s$,

v∈(V\$V_s$). На кроці s+1 для кожної вершини $v_r \in V_s$ обчислюємо відстань до вершини v_r : $L(s+1)(v_r) = l(v_r) + \min_{v^* \in V_s} w(v_r, v^*)$, де min береться по всіх ребрах e=(v_r, v∗), $v^* \in \overline{V}_s$, після чого знаходимо min серед величин L(s+1)(v_r). Нехай цей min досягнуто для вершин v_{r0} і відповідної їй $v^* \in \overline{V}_s$, що назвемо v_{s+1}. Тоді вершину v_{s+1} називаємо (s+1)-ю НВ, одержуємо множину $V_{s+1} = V_s \cup v_{s+1}$ і зростаюче дерево $D_{s+1} = D_s \cup (v_{r0}, v_{s+1})$. (s+1)-й крок завершується перевіркою: чи є чергова НВ v_{s+1} відзначеною вершиною, що повинна бути за умовою задачі зв'язано найкоротшим ланцюгом з вершиною v_0 . Якщо так, то довжина шуканого ланцюга дорівнює $l(v_{s+1}) = l(v_{r0}) + w(v_{r0}, v_{s+1})$; при цьому шуканий ланцюг однозначно відновлюється з ребер зростаючого дерева D_{s+1}. У противному випадку випливає перехід до кроку s+2.

Плоскі і планарні графи

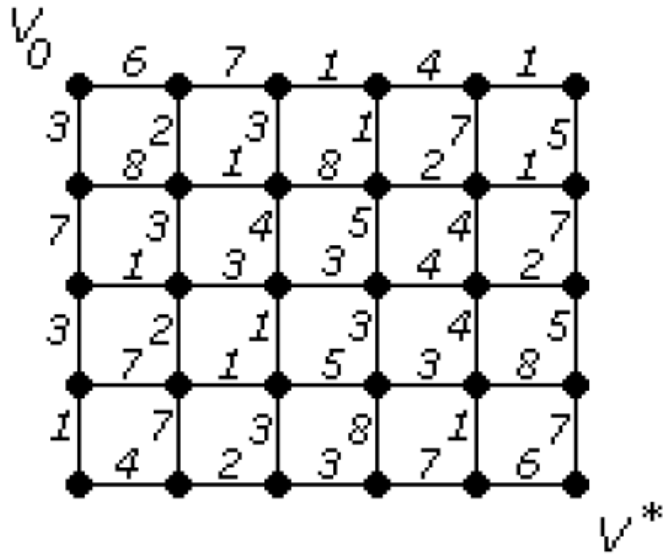
Плоским графом називається граф, вершини якого є точками площини, а ребра – безперервними лініями без самоперетинань, що з'єднують відповідні вершини так, що ніякі два ребра не мають спільних точок крім інцидентної їм обом вершини. Граф називається *планарним*, якщо він є ізоморфним плоскому графу.

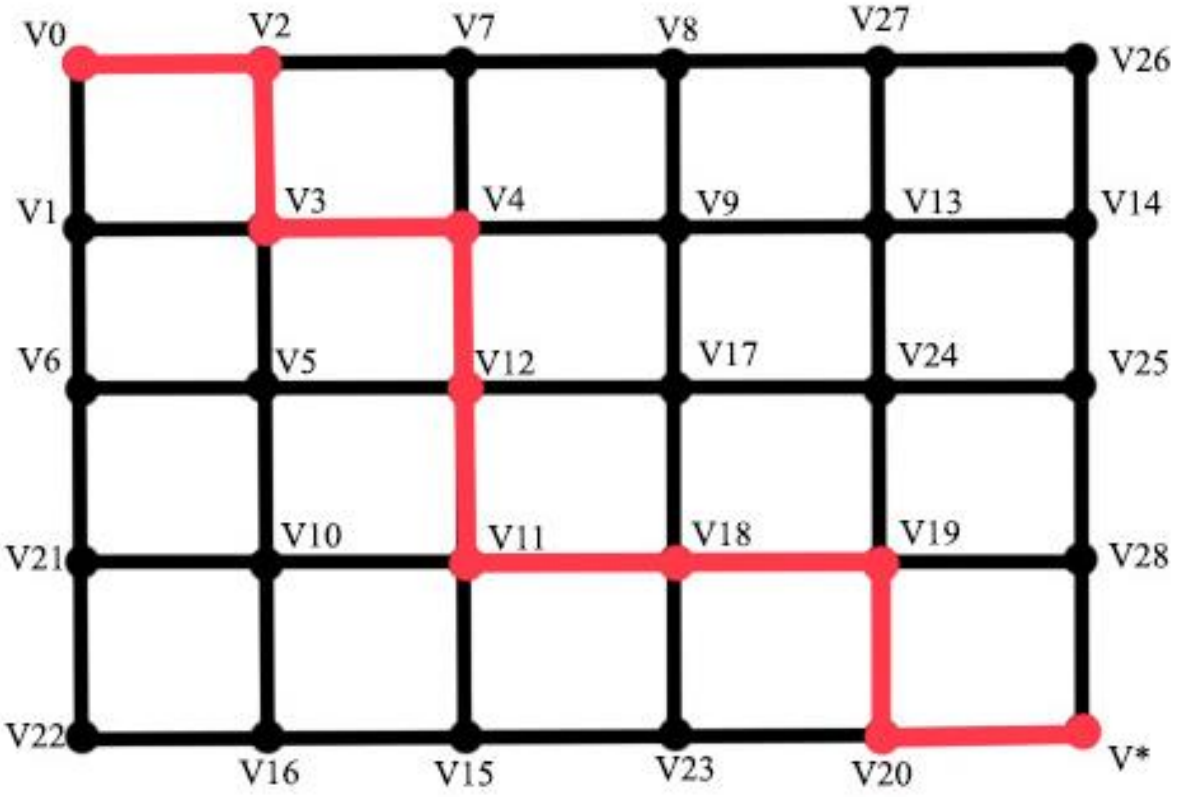
Гранню плоского графа називається максимальна по включенню множина точок площини, кожна пара яких може бути з'єднана жордановою кривою, що не перетинає ребра графа. Границею грані будемо вважати множину вершин і ребер, що належать цій грані.

ІНДИВІДУАЛЬНІ ЗАВДАННЯ

Завдання № 1. Розв'язати на графах наступні 2 задачі:

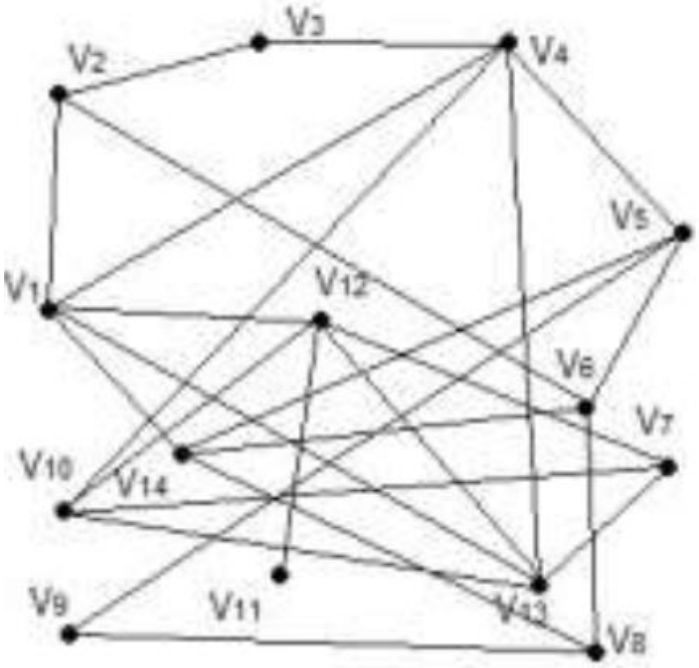
1. За допомогою алгоритму Дейкстра знайти найкоротший шлях у графі поміж парою вершин V_0 і V_*.



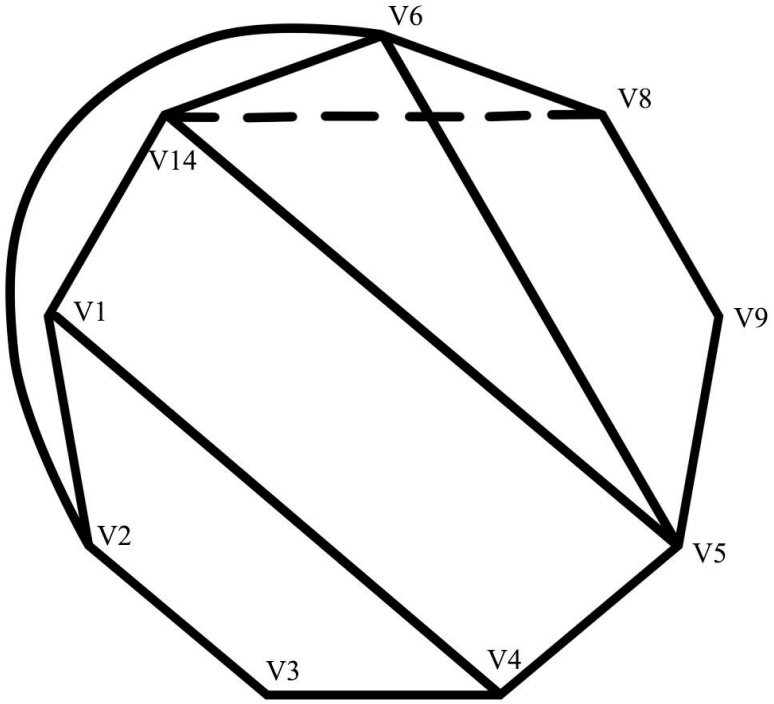


Вага найкоротшого шляху = 25.

2. За допомогою γ -алгоритма зробити укладку графа у площині, або довести що вона неможлива.



Виділимо підграф з ребер V1, V2, V3, V4, V5, V9, V8, V6, V14
Далі з’єднуємо V2 і V6, V1 і V4, V5 і V6, V5 і V14.
Далі бачимо, що з’єднати V8 і V14 неможливо, отже і гама-укладка також неможлива.



Завдання №2. Написати програму, яка реалізує алгоритм Дейкстри знаходження найкоротшого шляху між парою вершин у графі. Протестувати розроблену програму на графі згідно свого варіанту.

Скріншоти коду програми:

```
#include <iostream>
using namespace std;
int n, i, j, q;
int dist[40];
bool visited[40];
int pred[40];

void creating(int c[40][40]){
    int width, height, temp;
    cout << "Enter the number of vertices: ";
    cin >> n;
    for(i = 0; i < n; i++){
        for(j = 0; j < n; j++){
            c[i][j] = 0;
        }
    }
    cout << "Enter the width and height of graph: ";
    cin >> width >> height;
    cout << "length between vertices : ";
    for(i = 0; i < n; i++){
        for(j = i+1; j < n; j++){
            if(j==i+1 || j == i+width){
                cin >> temp;
                c[i][j] = temp;
            }
            else{
                c[i][j] = 0;
            }
        }
    }
}

int minimumDist(){
    int minimum = 10000, minDist;
    for(int v = 0; v < n; v++){
        if(visited[v] == false && dist[v]<=minimum){
            minimum = dist[v];
                minDist = v;
        }
    }
    return minDist;
}

void printPath(int j){
    if(pred[j] == -1){
        return;
    }
    printPath(pred[j]);
    cout << "V" << j+1 << " -> ";
}

void deykstra(int c[40][40]){
```

```
    int start;
    cout << "Enter the first node: ";
    cin >> start;
    for(i = 0; i < n; i++){
        pred[0] = -1;
        dist[i] = 10000;
        visited[i] = false;
    }
    dist[start-1] = 0;
    for(int count = 0; count < n-1; count++){
        int u = minimumDist();
        visited[u] = true;
        for(int v = 0; v < n; v++){
            if(!visited[v] && c[u][v] && dist[u] + c[u][v] < dist[v]){
                pred[v] = u;
                dist[v] = dist[u] + c[u][v];
            }
        }
    }
    cout << "The least way is: " << dist[29] << endl;
    cout << "The way is: V1 -> ";
    printPath(29);
}

int main() {
    int c[40][40];
    creating(c);
    deykstra(c);
    return 0;
}
```

Скріншоти виконання програми:

```
Enter the number of vertices:30
 Enter the width and height of graph:6 5
 length between vertices :6 3 7 2 1 3 4 1 1 7 0 5 8 7 1 3 8 4 2 5 1 4 0 7 1 3 3 2 3 1 4 3 2 4 0 5 7 1 1 7 5 3 3 8 8 1 0 7 4 2 3 7 6
 Enter the first node:1
 The l
east way is: 29
The way is: V1 -> V2 -> V8 -> V14 -> V20 -> V21 -> V22 -> V23 -> V29 -> V30 ->
```

Висновок

Ми набули практичних вмінь та навичок з використання алгоритму Дейкстри.