

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

Кафедра систем штучного інтелекту



Лабораторна робота № 5

З дисципліни: “Дискретна математика”

Виконав
студент групи КН 113
Карабін Я. В.

Викладач:
Мельникова Н. І.

Тема роботи

Знаходження найкоротшого маршруту за алгоритмом Дейкстри.

Плоскі планарні графи

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

Короткі теоретичні відомості

Задача про найкоротший ланцюг. Алгоритм Дейкстри.

Дано n -вершинний граф $G = (V, E)$, у якому виділено пару вершин $v_0, v^* \in V$, і кожне ребро зважене числом $w(e) \geq 0$. Нехай $X = \{x\}$ – множина усіх простих ланцюгів, що з'єднують v_0 з v^* , $x = (V_x, E_x)$. Цільова функція $F(x) = \sum_{e \in E_x} w(e) \rightarrow \min$. Потрібно знайти найкоротший ланцюг, тобто $x_0 \in X: F(x_0) = \min_{x \in X} F(x)$.

Перед описом **алгоритму Дейкстри** подамо визначення термінів “ k -а найближча вершина” і “дерево найближчих вершин”. Перше з цих понять визначається індуктивно так.

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 $V_s = \{v_0, v_1, \dots, v_s\}$ і відповідне їй зростаюче дерево $D_s = \{e_1, e_2, \dots, e_s\}$. 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 $v_r \in V_s$, $v \in (V \setminus V_s)$. На кроці $s+1$ для кожної вершини $v_r \in V_s$ обчислюємо відстань до вершини v_r : $L(s + 1)(v_r) = l(v_r) + \min_{v^* \in \overline{V_s}} w(v_r, v^*)$, 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 протилежному випадку впливає 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 обох 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.

Алгоритм γ укладання графа G являє собою процес послідовного приєднання до деякого укладеного підграфа \tilde{G} графа G нового ланцюга, обидва кінці якого належать \tilde{G} . При цьому в якості початкового плоского графа \tilde{G} вибирається будь-який простий

цикл графа G . Процес продовжується доти, поки не буде побудовано плоский граф, ізоморфний графові G , або приєднання деякого ланцюга виявиться неможливим. В останньому випадку граф G не є планарним.

Нехай побудоване деяке укладання підграфа \tilde{G} графа G .

Сегментом S відносно \tilde{G} будемо називати підграф графа G одного з наступних виглядів:

- ребро $e \in E$, $e = (u, v)$, таке, що $e \notin \tilde{E}$, $u, v \in \tilde{V}$, $\tilde{G} = (\tilde{V}, \tilde{E})$;
- зв'язний компонент графа $G - \tilde{G}$, доповнений всіма ребрами графа G , інцидентними вершинам узятим з компонента, і кінцями цих ребер.

Вершину v сегмента S відносно G будемо називати *контактною*, якщо $v \in \tilde{V}$.

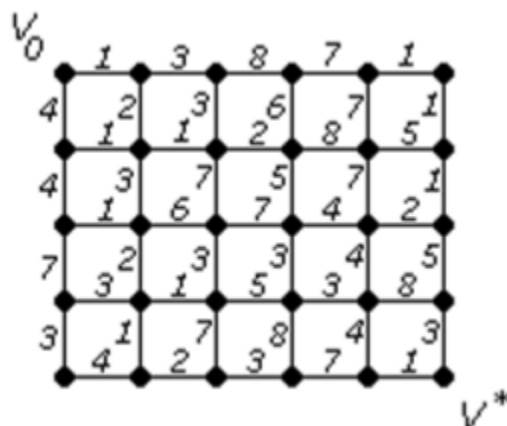
Припустимою гранню для сегмента S відносно \tilde{G} називається грань Γ графа \tilde{G} , що містить усі контактні вершини сегмента S . Через $\Gamma(S)$ будемо позначати множину припустимих граней для S .

Назвемо α – ланцюгом простий ланцюг L сегмента S , що містить дві різні контактні вершини і не містить інших контактних вершин.

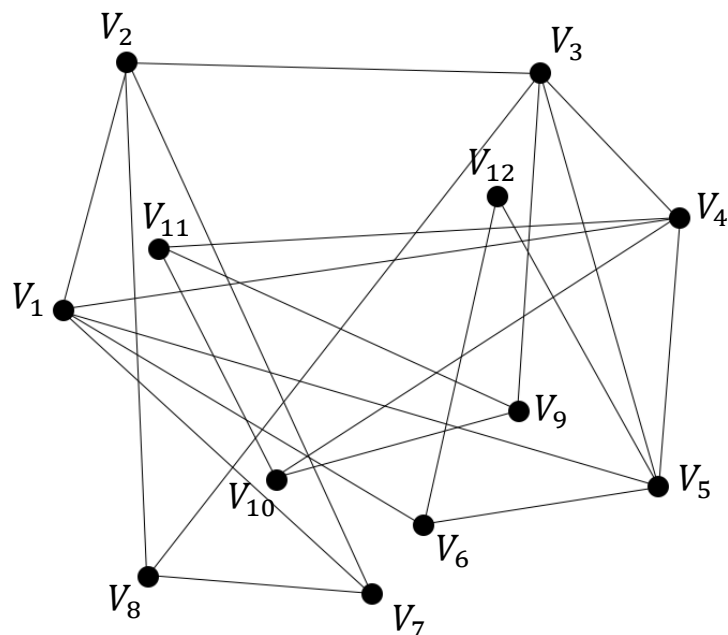
ІНДИВІДУАЛЬНІ ЗАВДАННЯ. ДОДАТОК 1

Варіант №12

1. За допомогою алгоритму Дейксти знайти найкоротший шлях у графі поміж парою вершин V_0 і V^* .



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



Розв'язки

1.

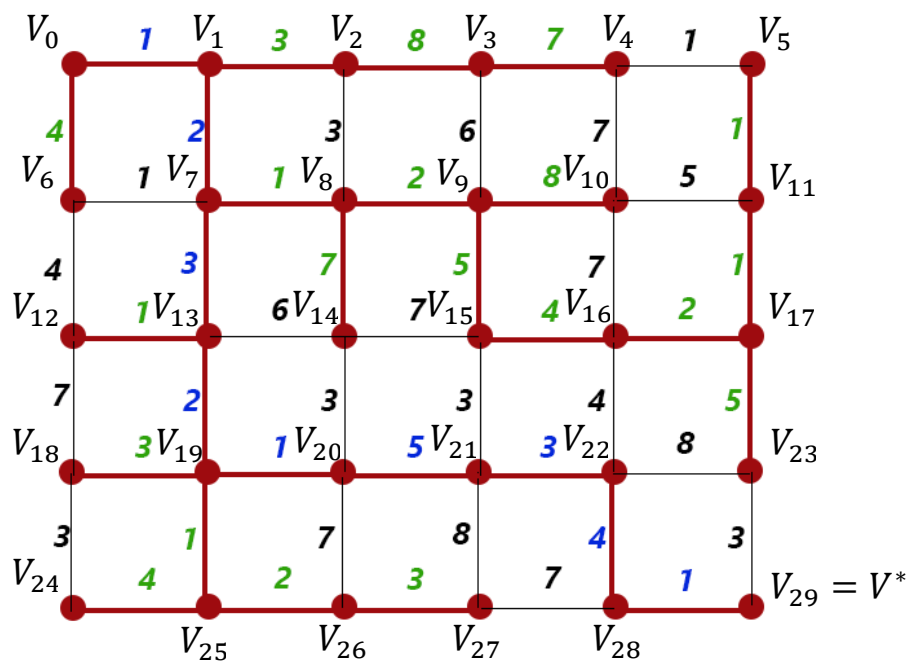


Рис. 1. Кістякове дерево для знаходження найкоротшого шляху від джерела, до кожної вершини графа

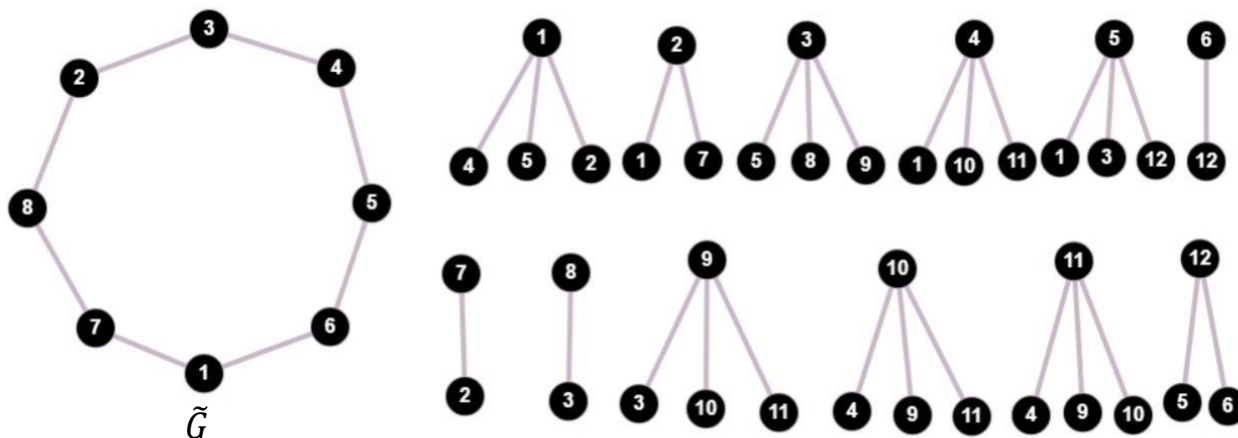
Далі наведена таблиця відстаней від джерела до решти вершин:

V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9	V_{10}	V_{11}	V_{12}	V_{13}	V_{14}
1	4	12	19	19	4	3	4	6	14	18	7	6	11

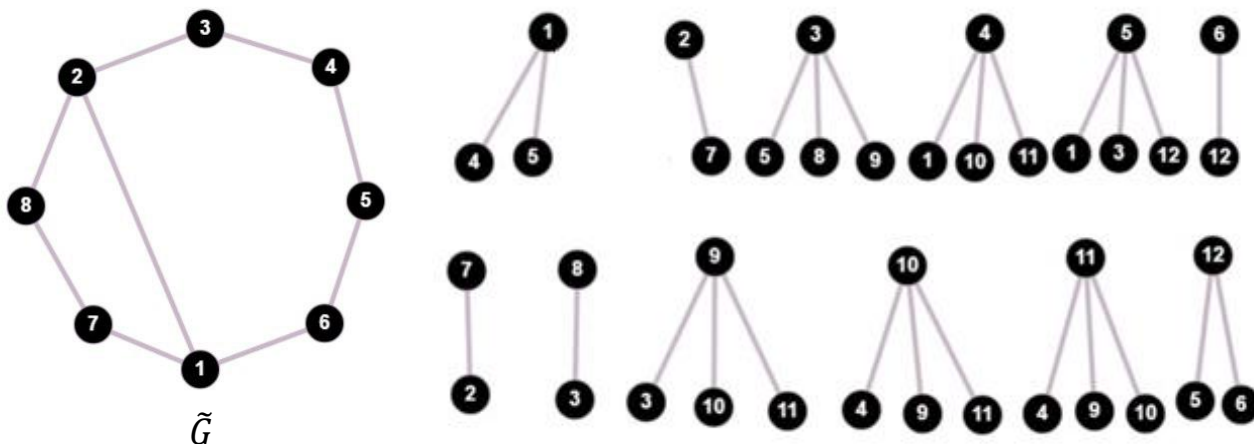
V_{15}	V_{16}	V_{17}	V_{18}	V_{19}	V_{20}	V_{21}	V_{22}	V_{23}	V_{24}	V_{25}	V_{26}	V_{27}	V_{28}
11	15	17	11	8	9	14	17	22	13	9	11	14	21

Найкоротша відстань від джерела до останньої вершини: $l(V_{29}) = l(V^*) = 22$.

2. Вибираємо найдовший цикл, і поступово приєднуємо утворені ланцюги.



Крок 1: Приєднання першого ланцюга



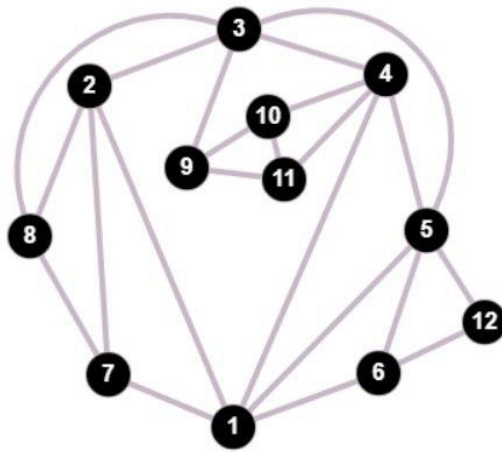
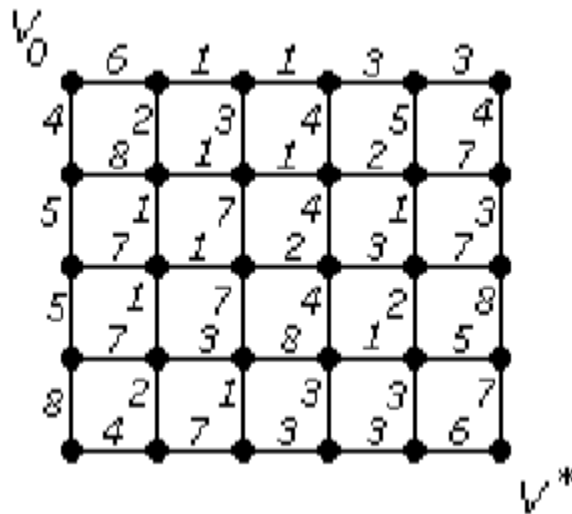


Рис. 2. Успішна укладка графа у площині

Додаток 2

Написати програму, яка реалізує алгоритм Дейкстри знаходження найкоротшого шляху між парою вершин у графі. Протестувати розроблену програму на графі згідно свого варіанту.



Представимо даний граф у текстовому файлі “Graph”, вказавши кількість вершин, номери з’єднаних ребер та їх ваги відповідно.

30	(7,8) 1
(0,1) 6	(8,9) 1
(1,2) 1	(9,10) 2
(2,3) 1	(10,11) 7
(3,4) 3	(12,13) 7
(4,5) 3	(13,14) 1
(6,7) 8	(14,15) 2

(15,16)	3	(6,12)	5
(16,17)	7	(7,13)	1
(18,19)	7	(8,14)	7
(19,20)	3	(9,15)	4
(20,21)	8	(10,16)	1
(21,22)	1	(11,17)	3
(22,23)	5	(12,18)	5
(24,25)	4	(13,19)	1
(25,26)	7	(14,20)	7
(26,27)	3	(15,21)	4
(27,28)	3	(16,22)	2
(28,29)	6	(17,23)	8
(0,6)	4	(18,24)	8
(1,7)	2	(19,25)	2
(2,8)	3	(20,26)	1
(3,9)	4	(21,27)	3
(4,10)	5	(22,28)	3
(5,11)	4	(23,29)	7

Програмна реалізація:

```
#include <bits/stdc++.h>
using namespace std;
ifstream fin("Graph.txt");
int V;
int minDistance(int dist[], bool sptSet[])
{
    // Ініціалізація мінімального значення
    int min = INT_MAX, min_index;

    for (int v = 0; v < V; v++)
        if (sptSet[v] == false && dist[v] <= min)
            min = dist[v], min_index = v;

    return min_index;
}

// Виведення масиву найкоротших відстаней від джерела до решти вершин
int printSolution(int dist[])
{
    printf("Vertex \t\t Distance from Source\n");
    for (int i = 0; i < V; i++)
        printf("%d \t\t %d\n", i, dist[i]);
}
```



```

// Функція, що реалізує алгоритм найкоротшого шляху від джерела до вершини
// за допомогою алгоритму Дейкстри з матриці суміжності
void Dejkstra(int** Matrix_cost, int source)
{
    int dist[V];
    bool sptSet[V];
    for (int i = 0; i < V; i++)
        dist[i] = INT_MAX, sptSet[i] = false;
    dist[source] = 0;

    // Знайти найкоротший шлях до кожної вершини
    for (int count = 0; count < V - 1; count++) {
        int u;
        u = minDistance(dist, sptSet);
        sptSet[u] = true;
        for (int v = 0; v < V; v++)
            if (!sptSet[v] && Matrix_cost[u][v] && dist[u] != INT_MAX
                && dist[u] + Matrix_cost[u][v] < dist[v]) {

                dist[v] = dist[u] + Matrix_cost[u][v];
            }
    }
    printSolution(dist);
}

int main()
{
    char c;
    int Vertexs_number, vertex1, vertex2, cost;
    if (!fin.is_open()) {
        cout << "File is not found!";
        exit(EXIT_SUCCESS);
    }
    fin >> Vertexs_number;
    V = Vertexs_number;
    int** Matrix_cost = new int* [V];
    for (int i = 0; i < V; i++)
        *(Matrix_cost + i) = new int[V];

    for (int i = 0; i < Vertexs_number; i++) {
        for (int j = 0; j < Vertexs_number; j++) {
            Matrix_cost[i][j] = 0;
        }
    }
    while (!fin.eof()) {
        fin >> c >> vertex1 >> c >> vertex2 >> c >> cost;
        vertex1; vertex2;
        if (vertex1 > vertex2) swap(vertex1, vertex2);
        for (int i = 0; i < Vertexs_number; i++) {
            for (int j = i + 1; j < Vertexs_number; j++) {
                if (i == vertex1 && j == vertex2) {
                    Matrix_cost[i][j] = cost;
                    Matrix_cost[j][i] = cost;
                    break;
                }
            }
        }
    }
}

```


```

    }
}
Dejkstra(Matrix_cost, 0);

for (int i = 0; i < V; i++) {
    delete Matrix_cost[i];
}
delete Matrix_cost;
}

```

Результат виконання програми

 "C:\Users\Admin\Desktop\New folder\~шЁЪЁхЁър\diskretlab5\bin\Debug\diskretlab5.exe"

Vertex	Distance from Source
0	0
1	6
2	7
3	8
4	11
5	14
6	4
7	8
8	9
9	10
10	12
11	18
12	9
13	9
14	10
15	12
16	13
17	20
18	14
19	10
20	13
21	16
22	15
23	20
24	16
25	12
26	14
27	17
28	18
29	24

Process returned 0 (0x0) execution time : 0.069 s
Press any key to continue.

Програма виводить на екран номери вершин і відстань від джерела до кожної з них відповідно.

Висновки

Ми набули практичних вмінь та навичок з використання алгоритму Дейкстри. Також ми навчилися робити гамма-укладку графа та виявляти чи граф все-таки планарний, тобто чи можна його зобразити на площині без перетину ребер.