МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра систем штучного інтелекту

Звіт

Лабораторна робота № 5

з дисципліни «Дискретна математика»

Виконала:
Студентка групи КН-113
Білинська Віолетта
Викладач:
Мельникова Н.І

Львів-2019р.

ТЕМА РОБОТИ

Знаходження найкоротшого маршруту за алгоритмом Дейкстри. Плоскі планарні графи.

МЕТА РОБОТИ

Набуття практичних вмінь та навичок з використання алгоритму Дейкстри.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Планарний граф - граф, який може бути зображений на площині без перетину ребер. Для визначення планарності графа застосовується критерій планарності, складовими якого є достатня умова планарності і необхідна умова планарності.
Достатня умова — якщо граф містить дводольний підграф K3, 3 або повний підграф K5, то він є не планарним. Необхідна умова — якщо граф не планарний, то він повинен містити більше 4 вершин, степінь яких більше 3, або більше 5 вершин степеня 2.
Плоский граф - 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. У протилежному випадку граф називатиметься просторовим.
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.
«Жадібний» алгоритм - це алгоритм, який на кожному кроці вибирають оптимальний із можливих варіантів.
Алгоритм Дейкстри призначений для пошуку маршруту найкоротшої довжини від заданої вершини графа до інших. Класичний алгоритм Дейкстри працює тільки для графів без циклів від'ємної довжини.

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 ɣ укладання графа G являє собою процес послідовного приєднання до деякого укладеного підграфа G~ графа G нового ланцюга, обидва кінці якого належать G~. При цьому При цьому в якості початкового плоского графа 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.
Сегментом S відносно G~ називається підграф графа G одного з наступних виглядів:

- ребро e ∈ E , e = (u, v) , таке, що e ∉ E~, u, v ∈ V~, G~ = (V~, E~);
- зв'язний компонент графа G - G~, доповнений всіма ребрами графа G, інцидентними вершинам узятого компонента, і кінцями цих ребер.

Вершиною v сегмента S відносно G~ називається **контактною**, якщо v ∈ V~.
Припустимою гранню для сегмента S відносно G~ називається грань Г графа G~, що містить усі контактні вершини сегмента S.

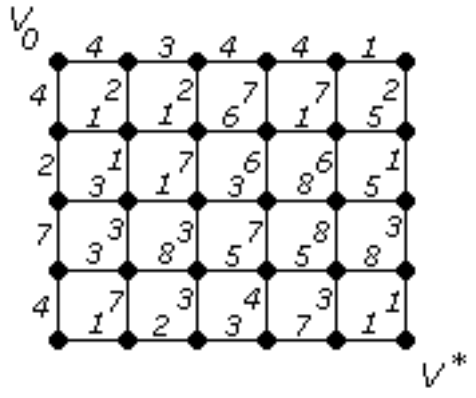
ЗАГАЛЬНЕ ЗАВДАННЯ

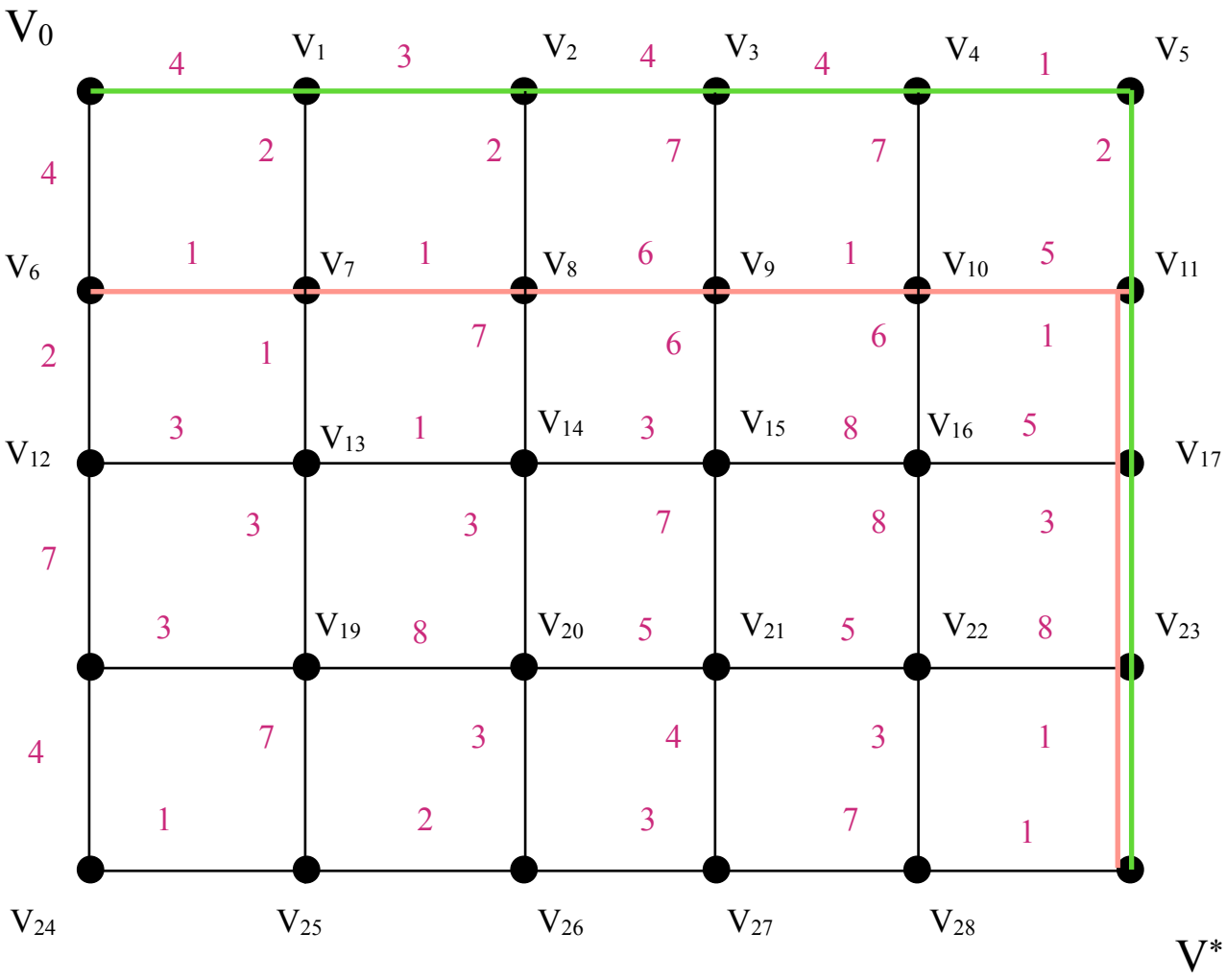
Варіант № 3.

Завдання № 1.

Розв'язати на графах наступні задачі:

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





Виконання:

1) Розпочинаємо з початкової вершини, яку будемо вважати поточною. В даний момент це вершина V_0. Переглядаємо її шлях до суміжних вершин - V_1 і V_6. До першої вершини відстань дорівнює 4, до шостої - 4.
2) Проходимо через $V_0 \rightarrow V_1 \rightarrow V_2$, відстань дорівнює 7.
3) Проходимо через $V_1 \rightarrow V_2 \rightarrow V_3$, відстань дорівнює 11.
4) Проходимо через $V_2 \rightarrow V_3 \rightarrow V_4$, відстань дорівнює 15.
5) Проходимо через $V_3 \rightarrow V_4 \rightarrow V_5$,відстань дорівнює 16.
6) Проходимо через $V_4 \rightarrow V_5 \rightarrow V_{11}$, відстань дорівнює 18.
7) Проходимо через $V_1 \rightarrow V_7$, відстань дорівнює 6.
8) Проходимо через $V_2 \rightarrow V_8$, відстань дорівнює 9.
9) Проходимо через $V_3 \rightarrow V_9$, відстань дорівнює 18.
10) Проходимо через $V_4 \rightarrow V_{10}$, відстань дорівнює 22.
11) Проходимо через $V_6 \rightarrow V_7$, відстань дорівнює 5, що є меншою за минулу відстань від вершини V_1, то викреслюєм минулу і робимо отриману відстань поточною.

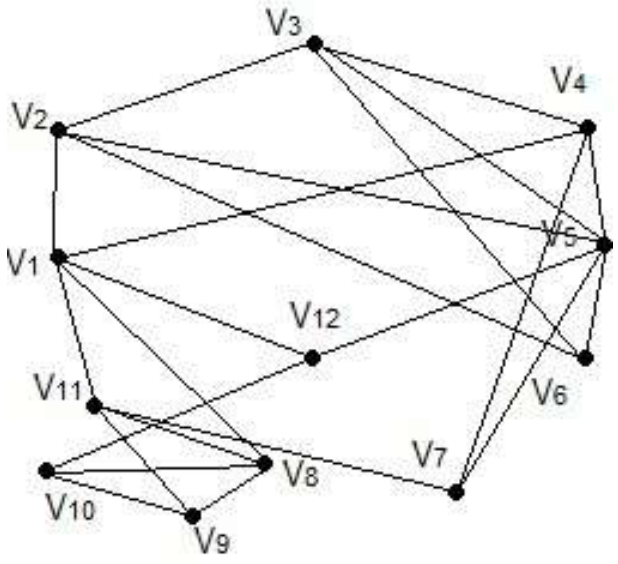
12) Проходимо через $V_7 \rightarrow V_8$, відстань дорівнює 6, що є меншою за минулу відстань від вершини V_2, то викреслюєм минулу і робимо отриману відстань поточною.
13) Проходимо через $V_8 \rightarrow V_9$, відстань дорівнює 2, що є меншою за минулу відстань від вершини V_3, то викреслюєм минулу і робимо отриману відстань поточною.
14) Проходимо через $V_9 \rightarrow V_{10}$, відстань дорівнює 13, що є меншою за минулу відстань від вершини V_4, то викреслюєм минулу і робимо отриману відстань поточною.
15) Проходимо через $V_{10} \rightarrow V_{11}$, відстань дорівнює 18, що дорівнює минулій відстань від вершини V_5, отже, залишаємо обидві відстані і саме тут утворюються два шляхи.
16) Проходимо через $V_{11} \rightarrow V_{17}$, відстань дорівнює 19.
17) Проходимо через $V_{12} \rightarrow V_{13}$, відстань дорівнює 9, тому залишаємо минулу відстань без змін.
18) Проходимо через $V_{13} \rightarrow V_{14}$, відстань дорівнює 7, що є меншою за минулу відстань від вершини V_8, то викреслюєм минулу і робимо отриману відстань поточною.
19) Проходимо через $V_{14} \rightarrow V_{15}$, відстань дорівнює 10, що є меншою за минулу відстань від вершини V_9, то викреслюєм минулу і робимо отриману відстань поточною.
20) Проходимо через $V_{15} \rightarrow V_{16}$, відстань дорівнює 18, що є меншою за минулу відстань від вершини V_{10}, то викреслюєм минулу і робимо отриману відстань поточною.
21) Проходимо через $V_{16} \rightarrow V_{17}$, відстань дорівнює 23, тому залишаємо минулу відстань без змін.
22) Проходимо через $V_{17} \rightarrow V_{13}$, відстань дорівнює 22.
23) Проходимо через $V_{18} \rightarrow V_{19}$, відстань дорівнює 16, тому залишаємо минулу відстань без змін.
24) Проходимо через $V_{19} \rightarrow V_{20}$, відстань дорівнює 17, тому залишаємо минулу відстань без змін.
25) Проходимо через $V_{20} \rightarrow V_{21}$, відстань дорівнює 15, що є меншою за минулу відстань від вершини V_{15}, то викреслюєм минулу і робимо отриману відстань поточною.
26) Проходимо через $V_{21} \rightarrow V_{22}$, відстань дорівнює 20, що є меншою за минулу відстань від вершини V_{16}, то викреслюєм минулу і робимо отриману відстань поточною.
27) Проходимо через $V_{22} \rightarrow V_{23}$, відстань дорівнює 28, тому залишаємо минулу відстань без змін.

28) Проходимо через $V_{23} \rightarrow V^*$, відстань дорівнює 23.
29) Проходимо через $V_{18} \rightarrow V_{24}$, відстань дорівнює 17.
30) Проходимо через $V_{24} \rightarrow V_{25}$, відстань дорівнює 25, тому залишаємо минулу відстань без змін.
31) Проходимо через $V_{25} \rightarrow V_{26}$, відстань дорівнює 18, тому залишаємо минулу відстань без змін.
32) Проходимо через $V_{26} \rightarrow V_{27}$ відстань дорівнює 16, що є меншою за минулу відстань від вершини V_{21}, то викреслюєм минулу і робимо отриману відстань поточною.
33) Проходимо через $V_{27} \rightarrow V_{28}$ відстань дорівнює 23, що дорівнює минулій відстань від вершини V_{22}, тому залишаємо минулу відстань без змін.
34) Проходимо через $V_{28} \rightarrow V^*$ відстань дорівнює 23, що дорівнює минулій відстань від вершини V_{23}, тому залишаємо минулу відстань без змін.

Отже, у даному графі за алгоритмом Дейкстри існують два мінімальних шляхи від вершини V_0 до V^* , які дорівнюють 23.

Завдання № 2.

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

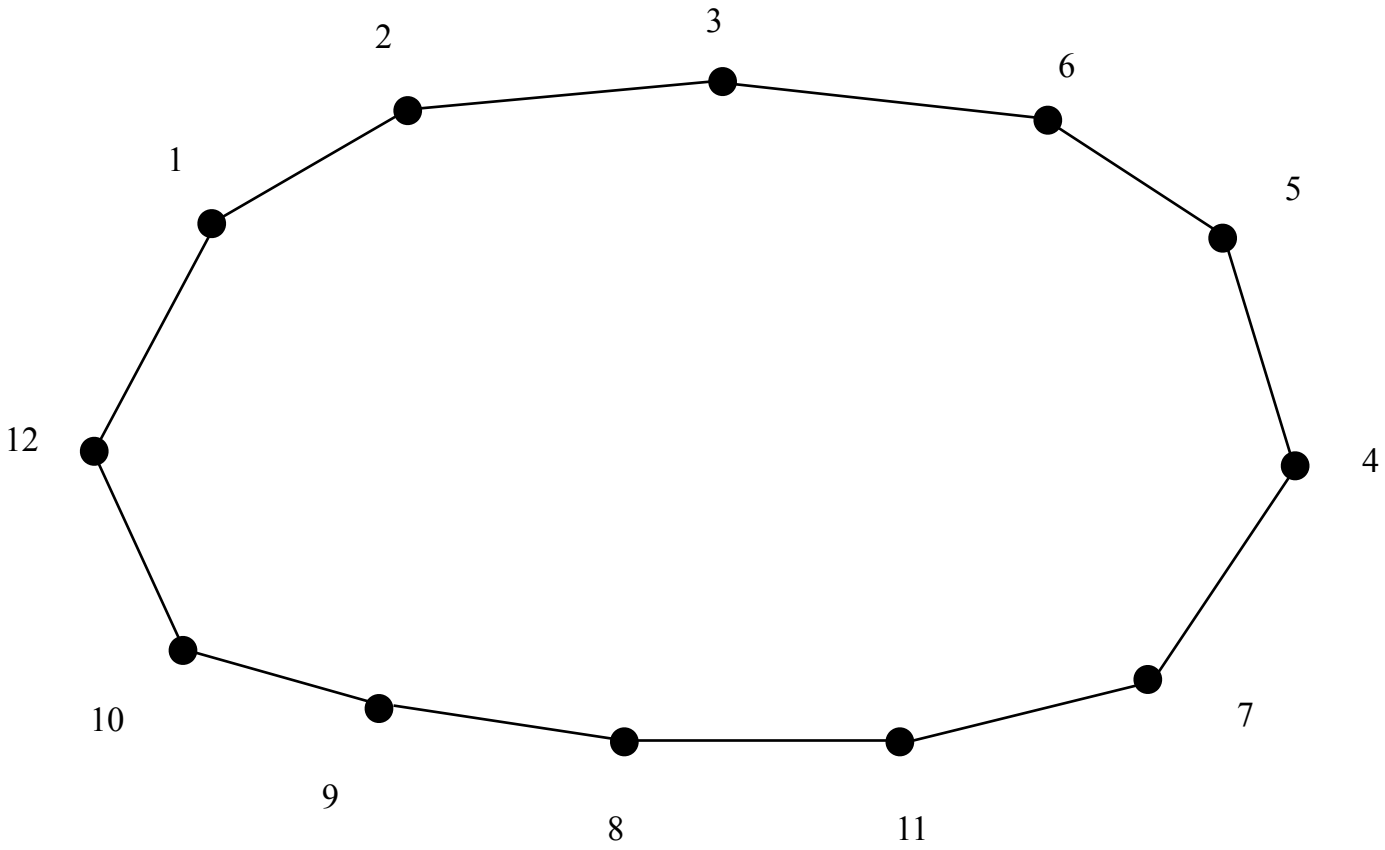


До умов коректної роботи алгоритму існують певні умови для графів:

1. Граф повинен бути зв'язним.
2. Граф повинен мати хоча б один цикл.
3. Граф не повинен мати мостів, тобто ребер, після видалення яких, граф розпадається на дві компоненти зв’язності.

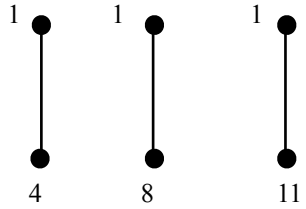
Якщо порушено властивість 1, то граф потрібно укладати окремо за компонентами зв'язності. Якщо порушено властивість 2, то граф - дерево і намалювати його плоску укладку тривіально. Випадок порушення властивості 2 потрібно розглянути більш докладно. Якщо в графі є мости, то їх потрібно розрізати, провести окремо плоску укладку кожного компонента зв'язності, а потім з'єднати їх мостами.

Етап 1: виділимо деякий простий цикл графа і укладемо його на площині. У даному випадку я виділяю цикл $1 \to 2 \to 3 \to 6 \to 5 \to 4 \to 7 \to 11 \to 8 \to 9 \to 10 \to 12 \to 1$.

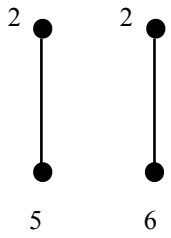


Етап 2: виділити грані, які не входять до новоутвореного простого циклу.

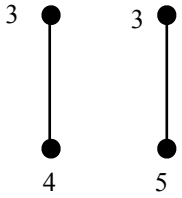
Вершина 1:



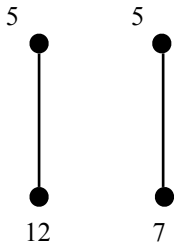
Вершина 2:



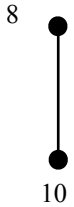
Вершина 3:



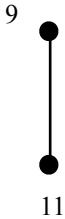
Вершина 5:



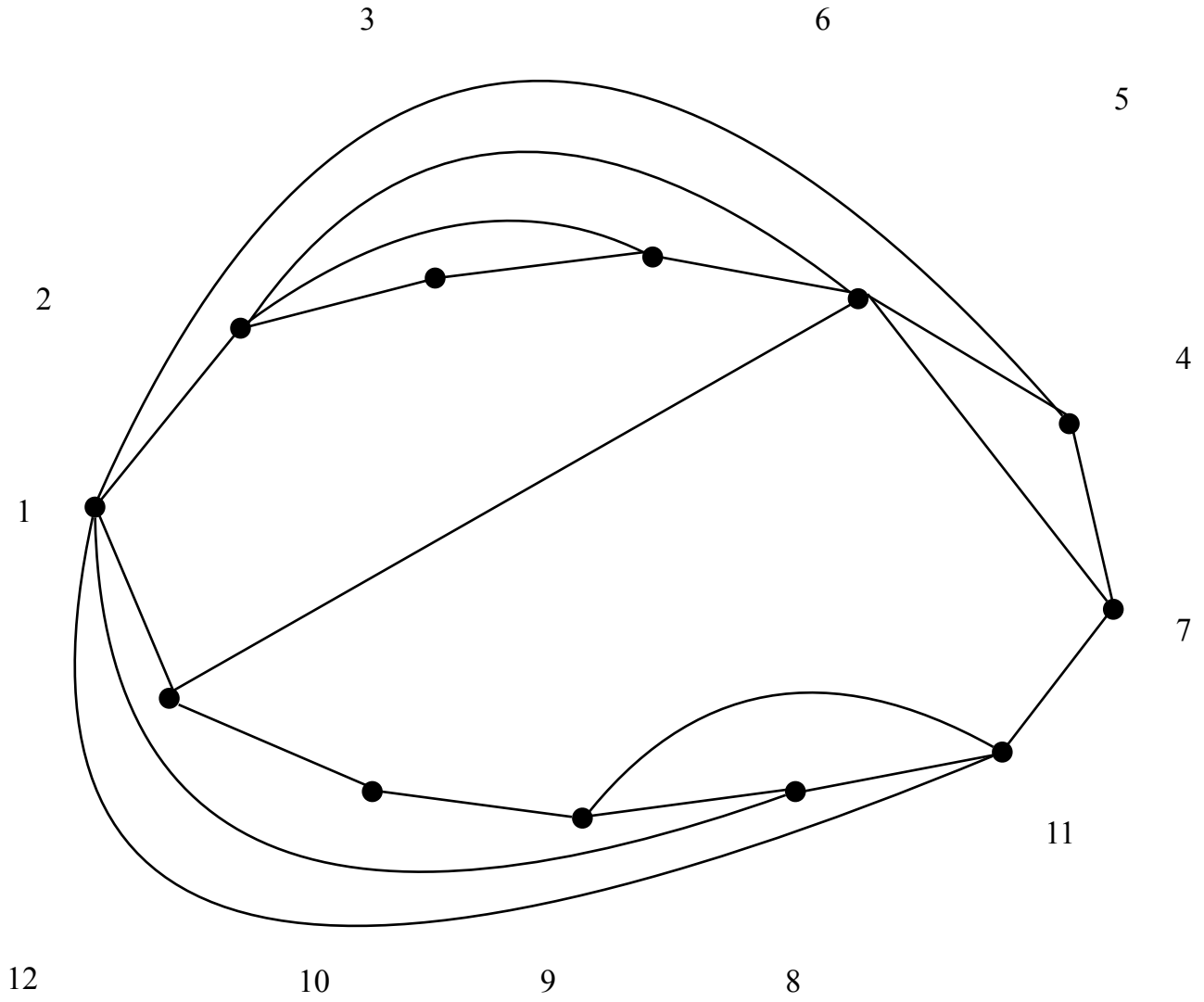
Вершина 8:



Вершина 9:

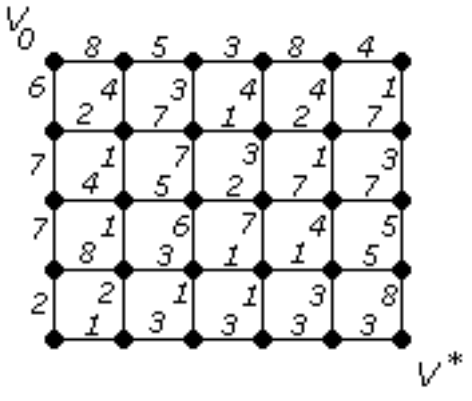


Етап 3: укладаємо граф на плоскій площині.



Завдання №2.

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



ПРОГРАМНА РЕАЛІЗАЦІЯ

```cpp
#include<iostream>

using namespace std;

int N;
int graph[40][40];
int path[40];
bool check[40];
int formation[40];

void makegrap()
{
    int i, j, max, u, v, w;
    int col, row;

    cout <<"Enter the number of nodes: ";
    cin>> N;
    cout << "Enter the number of columns: ";
    cin >> col;
    cout << "Enter the number of rows: ";
    cin >> row;


    for(i = 0; i < N; i++) {
        for (j = i + 1; j < N; j++) {
            if (j == i + 1 || j == i + col) {
                cout << "Verticle № " << i + 1 << " - - " ;
                cout << "verticle № " << j + 1 << ": ";
                cin >> graph[i][j];
            } else {
                graph[i][j] = 0;
            }
        }
    }

}

int pathmin()
{
    int min = 10000, minDist;
    for (int v = 0; v < N; v++)
        if (check[v] == false && path[v] <= min)
        {
```

```cpp
        }
    return minDist;
}
void printgraph(int j)
{
    if (formation[j] == -1)
        return;
    printgraph(formation[j]);
    cout << "Verticle №" << j + 1 << " -> ";
}
void pathformation() {
    int src;
    cout << "From which node do you want to start search? ";
    cin >> src;
    cout <<"Your choice is accepted..." << endl;

    for (int i = 0; i < N; i++) {
        formation[0] = -1;
        path[i] = 10000;
        check[i] = false;
    }
    path[src - 1] = 0;
    for (int count = 0; count < N - 1; count++) {
        int u = pathmin();
        check[u] = true;
        for (int v = 0; v < N; v++)
            if (!check[v] && graph[u][v] &&
                path[u] + graph[u][v] < path[v]) {
                formation[v] = u;
                path[v] = path[u] + graph[u][v];
            }
    }

    cout << "Finding the least way..." << endl;
    cout << "Counting the weight of the least way..." << endl;
    cout << "Eureka! " << endl;
    cout << "Here is the most optimal way: " << endl;
    cout << "Verticle №1 -> ";
    printgraph( j: 29);
    cout << endl;
    cout << "The weight of way: ";
    cout << path[29] << endl;
}
```

```
        int u = pathmin();
        check[u] = true;
        for (int v = 0; v < N; v++)
            if (!check[v] && graph[u][v] &&
                path[u] + graph[u][v] < path[v]) {
                formation[v] = u;
                path[v] = path[u] + graph[u][v];
            }
    }

    cout << "Finding the least way..." << endl;
    cout << "Counting the weight of the least way..." << endl;
    cout << "Eureka! " << endl;
    cout << "Here is the most optimal way: " << endl;
    cout << "Verticle №1 -> ";
    printgraph( j: 29);
    cout << endl;
    cout << "The weight of way: ";
    cout << path[29] << endl;
}
int main()
{
    makegrap();
    pathformation();
    return 0;
}
```

РЕЗУЛЬТАТИ ПРОГРАМИ

```
Enter the number of nodes: 30
Enter the number of columns: 6
Enter the number of rows: 5
Verticle № 1 - - verticle № 2: 8
Verticle № 1 - - verticle № 7: 6
Verticle № 2 - - verticle № 3: 5
Verticle № 2 - - verticle № 8: 4
Verticle № 3 - - verticle № 4: 3
Verticle № 3 - - verticle № 9: 3
Verticle № 4 - - verticle № 5: 8
Verticle № 4 - - verticle № 10: 4
Verticle № 5 - - verticle № 6: 4
Verticle № 5 - - verticle № 11: 4
Verticle № 6 - - verticle № 7: 0
Verticle № 6 - - verticle № 12: 1
Verticle № 7 - - verticle № 8: 2
Verticle № 7 - - verticle № 13: 7
Verticle № 8 - - verticle № 9: 7
Verticle № 8 - - verticle № 14: 1
Verticle № 9 - - verticle № 10: 1
Verticle № 9 - - verticle № 15: 7
Verticle № 10 - - verticle № 11: 2
Verticle № 10 - - verticle № 16: 3
Verticle № 11 - - verticle № 12: 7
Verticle № 11 - - verticle № 17: 1
Verticle № 12 - - verticle № 13: 0
Verticle № 12 - - verticle № 18: 3
Verticle № 13 - - verticle № 14: 4
Verticle № 13 - - verticle № 19: 7
Verticle № 14 - - verticle № 15: 5
Verticle № 14 - - verticle № 20: 1
Verticle № 15 - - verticle № 16: 2
Verticle № 15 - - verticle № 21: 6
Verticle № 16 - - verticle № 17: 7
Verticle № 16 - - verticle № 22: 7
Verticle № 17 - - verticle № 18: 7
Verticle № 17 - - verticle № 23: 4
Verticle № 18 - - verticle № 19: 0
Verticle № 18 - - verticle № 24: 5
Verticle № 19 - - verticle № 20: 8
Verticle № 19 - - verticle № 25: 2
Verticle № 20 - - verticle № 21: 3
```

```
Verticle № 20 - - verticle № 26: 2
Verticle № 21 - - verticle № 22: 1
Verticle № 21 - - verticle № 27: 1
Verticle № 22 - - verticle № 23: 1
Verticle № 22 - - verticle № 28: 1
Verticle № 23 - - verticle № 24: 5
Verticle № 23 - - verticle № 29: 3
Verticle № 24 - - verticle № 25: 0
Verticle № 24 - - verticle № 30: 8
Verticle № 25 - - verticle № 26: 1
Verticle № 26 - - verticle № 27: 3
Verticle № 27 - - verticle № 28: 3
Verticle № 28 - - verticle № 29: 3
Verticle № 29 - - verticle № 30: 3
From which node do you want to start search? 1
Your choice is accepted...
Finding the least way...
Counting the weight of the least way...
Eureka!
Here is the most optimal way:
Verticle №1 -> Verticle №7 -> Verticle №8 -> Verticle №14 -> Verticle №20 -> Verticle №21 -> Verticle №22 -> Verticle №28 -> Verticle №29 -> Verticle №30 ->
The weight of way: 21
```

Висновок

На цій лабораторній роботі я навчилася знаходити найкоротший маршрут за алгоритмом Дейкстри, набула практичних вмінь з укладки графа і програмно реалізувала алгоритм Дейкстри і протестила на заданому мені графі.