МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота № 5

з дисципліни
«Дискретна математика»

Виконав:
студент групи КН-114
Кмитюк Катерина
Викладач:
Мельникова Н.І.

Львів – 2019 р.

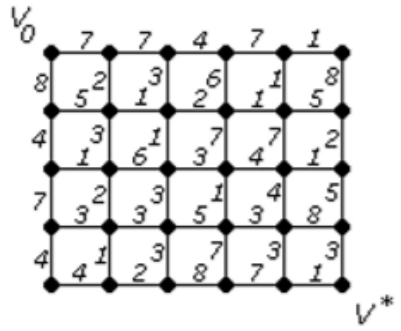
Тема: Знаходження найкоротшого маршруту за алгоритмом Дейкстри. Плоскі планарні графи.

Мета: Набуття практичних вмінь та навичок з використання алгоритму Дейкстри.

Завдання з додатку 1:

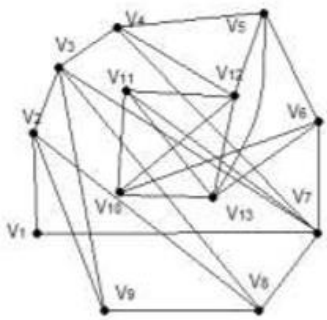
1. За допомогою алгоритму Дейкстра знайти найкоротший шлях у графі поміж парою вершин V_0 і V^*.

10



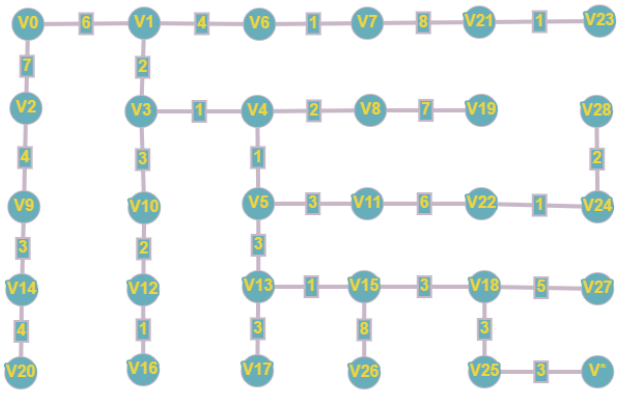
2. За допомогою γ-алгоритма зробити укладку графа у площині, або довести що вона неможлива.

10



Розв'язання:

1. Для побудови найкоротшого шляху від V0 дод V* додаємо нумеруючи вершини, що є найближчими до V0. Отримаємо наступне кістякове дерево:

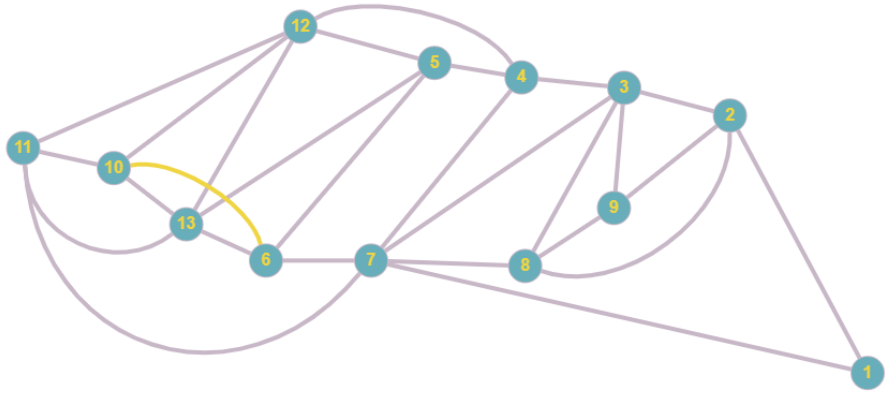


Множина найближчих вершин V={V0, V1, V2, V3, V4, V5, V6, V7, V8, V9, V10, V11, V12, V13, V14, V15, V16, V17, V18, V19, V20, V21, V22, V23, V24, V25, V26, V27, V28, V*}

Відповідне зростаюче дерево D={6, 7, 2, 1, 1, 4, 1, 2, 4, 3, 3, 2, 3, 3, 1, 1, 3, 3, 7, 4, 8, 6, 1, 1, 3, 8, 5, 2, 3}

Довжина шляху V0 – V* дорівнює 23.

2. Для розкладу побудуємо цикл і будемо поступово додавати ребра, що не увійшли в цикл, розміщуючи їх таким чином всередині утвореної фігури або зовні, щоб ребра не перетинались

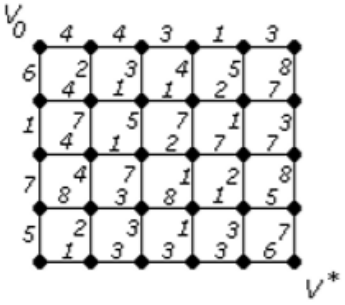


Початковий цикл утворюють вершини 11-10-13-6-7-8-9-2-3-4-5-12-11. Додаємо ребра 12-4, 12-10, 12-13,1 5-13, 5-6, 4-7, 3-7, 3-8, 3-9, 2-8, 2-1, 7-1, 7-11, 13-11. Але ребро 10-6 не можливо розмістити без перетину. Отже, для даного графа неможливо зробити γ-укладку.

Завдання з додатку 2:

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

10



Програмна реалізація:

```
#include <iostream>

using namespace std;

int main()
{
   int mind, min;

   int *v, *d;
   int m,n,k,p;
   int **w;
   int r, temp;
   bool *visit;

   cout << "Input count of vertices" << endl;
   cin >> m >> n;
   k=m*n;

   w=(int **)calloc(k, sizeof(int *));
   for (int i=0; i<k; i++)
      w[i]=(int *)calloc(k+1, sizeof(int));


   v = (int *)calloc(k, sizeof(int));
   visit = (bool *)calloc (k, sizeof(bool));
   d = (int *)calloc(k,sizeof (int));

   r=1;
```

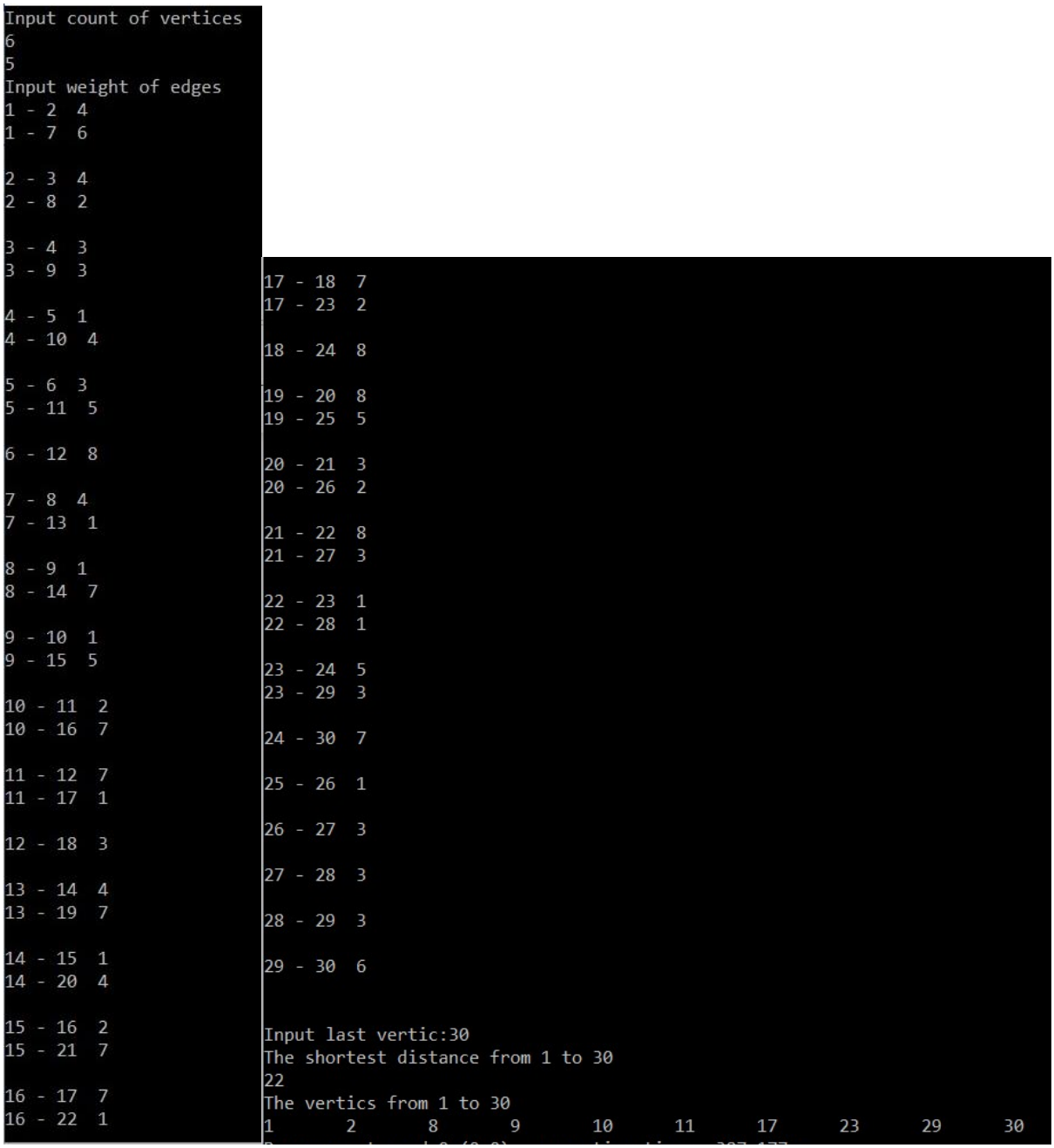
```
cout << "Input weight of edges" << endl;
for (int i=0; i<k; i++)
{
   for (int j=0; j<k; j++)
   {
      if (j==i+m)
      {
         cout << i+1 << " - " << j+1 << "  ";
         cin >> w[i][j];
      }
      else
      if (j==i+1)
      {
         if ((i+1)!=(m*r))
         {
            cout << i+1 << " - " << j+1 << "  ";
            cin >> w[i][j];
         }
         else
         {
            w[i][j]=w[j][i];
            r++;
         }
      }
      else w[i][j]=w[j][i];

   }
   cout << endl;
   visit [i]=0;
```

```
    }

    cout << "Input last vertic:"; cin >> p;

    for (int i = 0; i<k; i++)
    {
        d[i] = 10000;
        v[i] = 1;
    }
    d[0] = 0;

    do
    {
    mind = 10000;
    min = 10000;
    for (int i = 0; i<k; i++)
    { // Если вершину ещё не обошли и вес меньше min
      if ((v[i] == 1) && (d[i]<min))
      { // Переприсваиваем значения
        min = d[i];
        mind = i;
      }
    }
    // Добавляем найденный минимальный вес к текущему весу вершины и сравниваем с текущим минимальным весом вершины
    if (mind != 10000)
    {
      for (int i = 0; i<k; i++)
      {
```

```
    if (w[mind][i] > 0)
    {
      temp = min + w[mind][i];
      if (temp < d[i])
      {
        d[i] = temp;
      }
    }
  }
  v[mind] = 0;
}
} while (mind < 10000);

int q=p-1;
// Вывод кратчайших расстояний до вершин
cout << "The shortest distance from " <<1<<" to "<<p<< endl;
cout << d[q]<<endl;

int *ver;
ver = (int *)calloc(k, sizeof(int)); // массив посещенных вершин
ver[0] = p; // начальный элемент - конечная вершина
r = 1; // индекс предыдущей вершины
int weight = d[q];
while (q != 0) // пока не дошли до начальной вершины
{
    for (int i = 0; i<k; i++) // просматриваем все вершины
    if (w[q][i] != 0)   // если связь есть
    {
```

```
        temp = weight - w[q][i]; // определяем вес пути из предыдущей
вершины
        if (temp == d[i]) // если вес совпал с рассчитанным
        {               // значит из этой вершины и был переход
        weight = temp; // сохраняем новый вес
        q = i;       // сохраняем предыдущую вершину
        ver[r] = i + 1; // и записываем ее в массив
        r++;
        }
      }
  }
   cout << "The vertics from " <<1<<" to "<<p<< endl;
   for (int i = r - 1; i >= 0; i--)
      cout << ver[i]<<"\t";

   return 0;
}
```

Приклад виконання програми:



Висновок: у ході лабораторної роботи я навчилась використовувати алгоритм Дейкстри, робити γ-укладку графа на площині.