МІНІСТЕРСТВО ОСВІТИ І НАУКИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних наук та інформаційних технологій

Кафедра систем штучного інтелекту

Лабораторна робота №5

з курсу “Дискретна математика ”

Виконав:
ст. гр. КН-110
Петров Кирил

Викладач:
Мельникова Н.І.

Львів – 2018

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

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

ТЕОРЕТИЧНІ ВІДОМОСТІ ТА ПРИКЛАДИ РОЗВ'ЯЗАННЯ ЗАДАЧ

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.

Задача про найкоротший ланцюг. Алгоритм Дейкстри.

Дано n-вершинний граф $G = (V, E)$, у якому виділено пару вершин $v\ v \in V\ *$

0 , , і кожне ребро зважене числом $w(e) \geq 0$. Нехай

$X = \{x\}$ – множина усіх простих ланцюгів, що з'єднують 0 v з * v ,

( ) $x\ x\ E\ V\ x$, = . Цільова функція min) ( ) ($\rightarrow = \sum$

$\in e\ Ex$

$F\ x\ w\ e$. Потрібно

знайти найкоротший ланцюг, тобто : 0 $x \in X$ ( ) min ( ) 0 $F\ x\ F\ x$

$x \in X$

=

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 x0, E1 – множина усіх ребер e∈E, інцидентних v0. Серед ребер e∈E1 вибираємо ребро e(1) =(v0, v1), що має мінімальну вагу, тобто ((1)) min ( )

1

$w\ e\ w\ e$

$e \in E$

= . Тоді

v1 називаємо першою найближчою вершиною (НВ), число w(e(1)) позначаємо l(1) = l(v1) і називаємо відстанню до цієї НВ. Позначимо V1={v0, v1} – множину найближчих вершин.

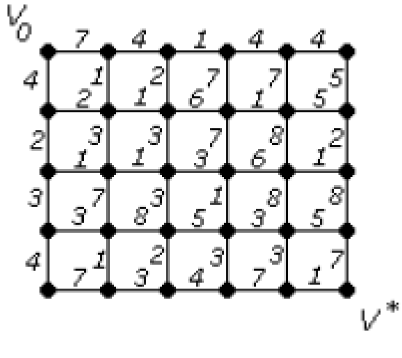
2-й крок індукції. Позначимо E2 – множину усіх ребер e=(v',v''), e∈E, таких що v'∈V1, v''∈(V \V1). Найближчим вершинам v∈V1 приписано відстані l(v) до кореня v0 , причому l(v0)=0. Введемо позначення: 1 V – множина таких вершин v''∈(V \V1), що ∃ ребра виду e =(v, v''), де v∈V1. Для всіх ребер e∈E2 знаходимо таке ребро e2=(v', v2), що величина l(v')+w(e2) найменша. Тоді v2 називається другою найближчою вершиною, а ребра e1, e2 утворюють зростаюче дерево

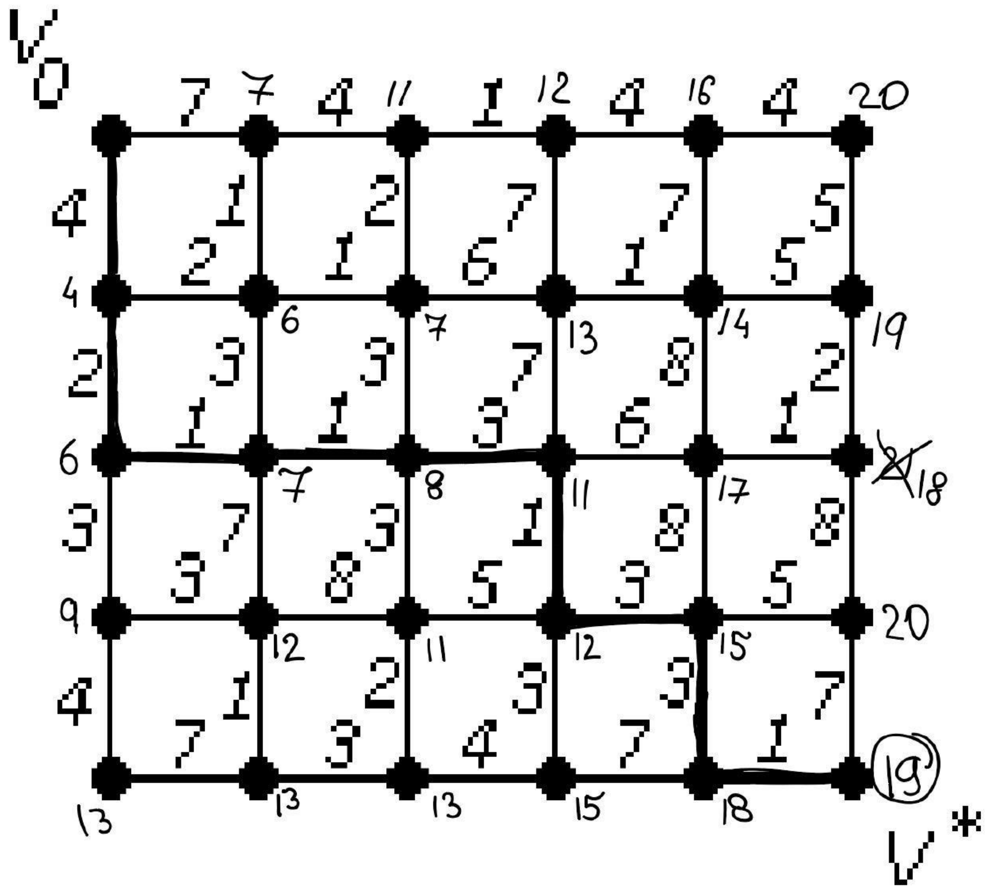
для виділених найближчих вершин D2 ={e1, e2}.
(s+1)-й крок індукції. Нехай у результаті s кроків виділено множину найближчих вершин Vs={v0, v1, …, vs} і відповідне їй зростаюче дерево Ds={e1, e2, …, es}... Для кожної вершини v∈Vs обчислена відстань l(v) від кореня v0 до v; *s V* – множина вершин v∈(V\Vs), для яких існують ребра вигляду e =(vr, v), де vr ∈Vs , v∈(V\Vs). На кроці s+1 для кожної вершини vr∈Vs обчислюємо відстань до вершини vr : (1)( ) ( ) min (,) *
*
L s v l v w v v r
v V
r r
∈ *s*
+ = + , де min
береться по всіх ребрах e=(vr, v∗), *v* ∈*V s* *
, після чого знаходимо min
серед величин L(s+1)(vr). Нехай цей min досягнуто для вершин *vr*0 і відповідної їй *v* ∈*V s* * , що назвемо vs+1. Тоді вершину vs+1 називаємо (s+1)-ю НВ, одержуємо множину Vs+1 =Vs Y *vs*+1 і зростаюче дерево Ds+1 =Ds Y (*vr*0 ,*vs*+1). (s+1)-й крок завершується перевіркою: чи є чергова НВ vs+1 відзначеною вершиною, що повинна бути за умовою задачі зв'язано найкоротшим ланцюгом з вершиною v0 . Якщо так, то довжина шуканого ланцюга дорівнює l(vs+1)=l(*vr*0)+w(*vr*0, vs+1); при цьому шуканий ланцюг однозначно відновлюється з ребер зростаючого дерева Ds+1. У противному випадку випливає перехід до кроку s+2.

Варіант 4

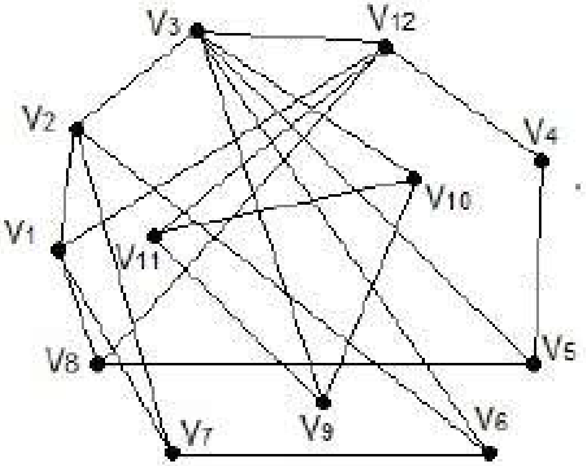
1. За допомогою алгоритму Дейкстра знайти найкоротший шлях у графі поміж парою вершин *V*0 і *V** .

4

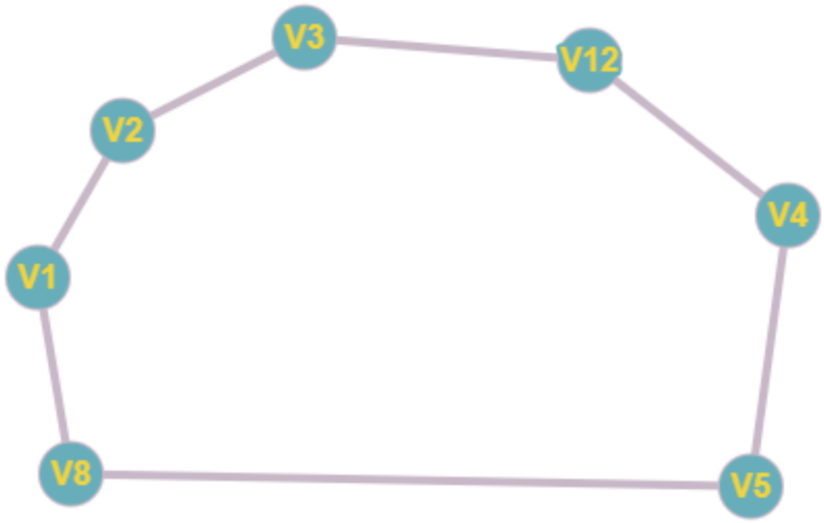




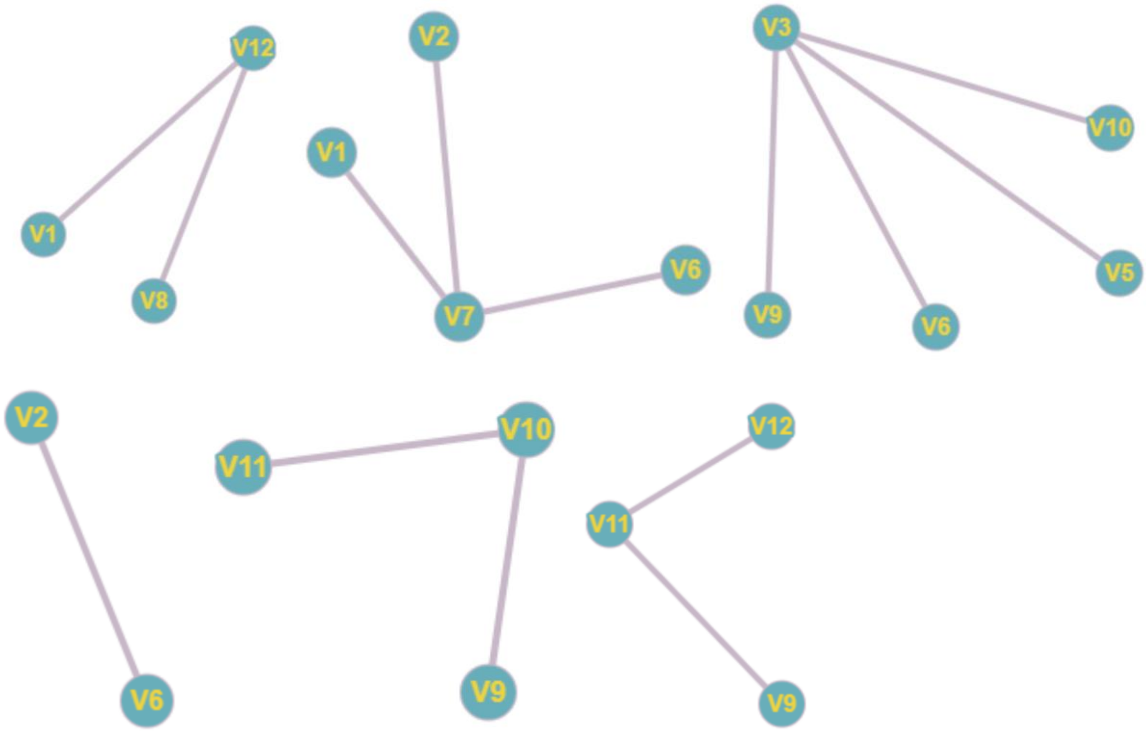
2. За допомогою γ-алгоритма зробити укладку графа у площині, або довести що вона неможлива.



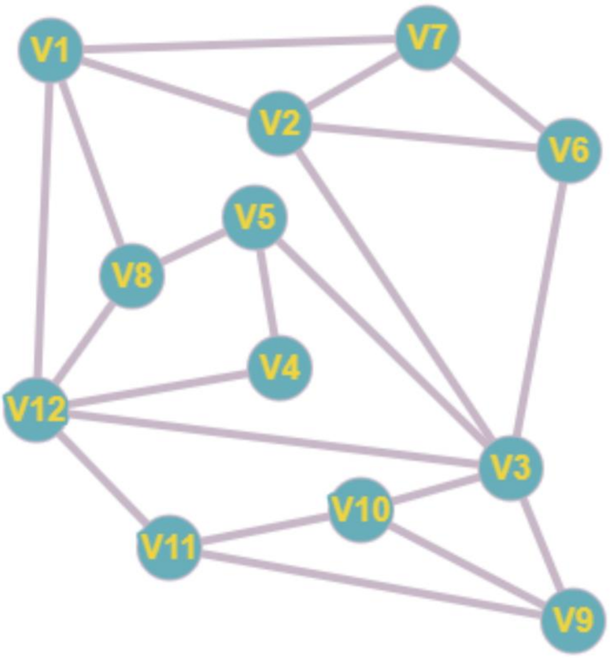
Вибираємо довільний цикл з графа:



Сегменти:



Плоский планарний граф:



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

```
#include<stdio.h>

#include<stdlib.h>

#define SIZE 30

#define INFINITY 100000

#define NOT_DEFINED -1


//Function Declaration

int readFile(char*);

int dijikstra();


//Global Graph matrix

int graph[SIZE][SIZE];



int main(int argc,char ** argv) {

	char * input;
```

```
	if(argc!=2) {
		printf("\nError: Invalid number of arguments!");
		return 0;
	}
	input = argv[1];
	readFile(input);
	dijikstra();
	printf("\n\n");
	return 0;
}


//Checks if Heap is Empty
//0-Heap is empty else 1-Heap not empty

int isHeapEmpty(int heap[SIZE]) {
	int i,res=0;
	for(i=0;i<SIZE;i++) {
		if(heap[i]==1) {
			res = 1;
			break;
		}
	}
	return res;
}


//Extract minimum element from the heap.
//return index f the minimum element
```

```
int extractMin(int heap[SIZE],int dist[SIZE]) {
	int i,min=INFINITY,res=-1;
	for(i=0;i<SIZE;i++) {
		if(min > dist[i] && heap[i] != 0) {
			min = dist[i];
			res = i;
		}
	}

	return res;

}

//Dijikstras algorithm Implementation

int dijikstra() {
	int i,start,near,current;
	int dist[SIZE],heap[SIZE],parent[SIZE];
	/*Initialization. Start vertex is vertex 0*/
	start=current=0;
	for(i=0;i<SIZE;i++) {
		dist[i]=INFINITY;
		heap[i]=1; //Denotes it is present in heap
		parent[i]=NOT_DEFINED;
	}
	dist[0]=0;

	while(isHeapEmpty(heap) == 1) {
```

```
            near = extractMin(heap,dist);
            if(near<0)
                break;
            heap[near]=0;
            for(i=0;i<SIZE;i++)
             {
        if(heap[i]==1 && (dist[i] > dist[near] + graph[near][i]))
                 {


                    dist[i] = dist[near] + graph[near][i];
                    parent[i] = near;
                }
            }
            current=near;
        }
        /*Prints the Output*/
        for(i=0;i<SIZE;i++) {
                if(i==start)
                    continue;
                current=i;
                if(dist[i] >=INFINITY) {
                    printf("\n\n (S%d,S%d): No Path Found!",start,i);
                    continue;
                }
                printf("\n\n (S%d,S%d): Distance=%d\tPath=
%d",start,i,dist[i],current);
                while(1) {
                    current=parent[current];
```

```
                    printf(" <-%d",current);
                    if(current==start)
                        break;
                }
        }
        return 0;
}


//Reads the input matrix

int readFile(char * filename) {
        FILE *fp;
        char num[255];
        int i=0,j=0,k=0,val;

        fp = fopen(filename,"r");
        if(fp == NULL) {
                printf("\nERROR in opening the file!\n\n");
                return 0;
        }
        char ch;
        ch=fgetc(fp);
        for(i=0;i<SIZE;i++) {
                for(j=0;j<SIZE;j++) {
                        k=0;
                        while(ch!='\n' && ch!=',' && ch!=EOF) {
                                num[k++]=ch;
                                ch=fgetc(fp);
```

```
                }
                num[k]='\n';
                val = atoi(num);
                if(val == NOT_DEFINED) {
                        graph[i][j]=INFINITY;
                }
                else
                    graph[i][j]=val;

                ch=fgetc(fp);
            }
        }
        fclose(fp);
        return 0;
}
```

```
(S0,S17): Distance=21  Path= 17 <-11 <-10 <-9 <-8 <-7 <-6 <-0
(S0,S18): Distance=9   Path= 18 <-12 <-6 <-0
(S0,S19): Distance=12  Path= 19 <-18 <-12 <-6 <-0
(S0,S20): Distance=11  Path= 20 <-14 <-13 <-12 <-6 <-0
(S0,S21): Distance=14  Path= 21 <-15 <-14 <-13 <-12 <-6 <-0
(S0,S22): Distance=17  Path= 22 <-21 <-15 <-14 <-13 <-12 <-6 <-0
(S0,S23): Distance=22  Path= 23 <-22 <-21 <-15 <-14 <-13 <-12 <-6 <-0
(S0,S24): Distance=13  Path= 24 <-18 <-12 <-6 <-0
(S0,S25): Distance=13  Path= 25 <-19 <-18 <-12 <-6 <-0
(S0,S26): Distance=13  Path= 26 <-20 <-14 <-13 <-12 <-6 <-0
(S0,S27): Distance=17  Path= 27 <-26 <-20 <-14 <-13 <-12 <-6 <-0
(S0,S28): Distance=18  Path= 28 <-22 <-21 <-15 <-14 <-13 <-12 <-6 <-0
(S0,S29): Distance=19  Path= 29 <-28 <-22 <-21 <-15 <-14 <-13 <-12 <-6 <-0
```

Висновок: в результаті проведеної роботи ми ознайомились із знаходженням найкоротшого маршруту за алгоритмом Дейкстри та побудовою плоских планарних графів.