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"

# Знаходження найкоротшого маршруту за алгоритмом Дейкстри. Плоскі планарні графи

Виконав:
ст. гр. КН-110
Помірко Олег

Викладач:
Мельникова Н.І.

Львів – 2018

# Лабораторна робота № 5.

**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 $v_0, v^* \in V$, 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 $F(x)=\sum_{e\in E_x} w(e) \to \min$. 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 $Vs=\{v_0, v_1, \ldots, v_s\}$ 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 $v_r \in V_s$, $v \in (V \setminus V_s)$. На кроці s+1 для кожної вершини $v_r \in V_s$ обчислюємо відстань до вершини $v_r$ : $L(s+1)(v_r)=l(v_r)+\min\limits_{v^* \in V_s} w(v_r,v^*)$, де min береться по всіх ребрах $e=(v_r, v^\star)$, $v^* \in \overline{V}_s$, 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 $V_{s+1}=V_s \text{ Y } v_{s+1}$ і зростаюче дерево $D_{s+1}=D_s \text{ Y } (v_{r0}, v_{s+1})$. (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.

Приклад. У графі на рис.5.1 знайти найкоротший ланцюг для виділеної пари вершин $v_0$ , $v_\star$.

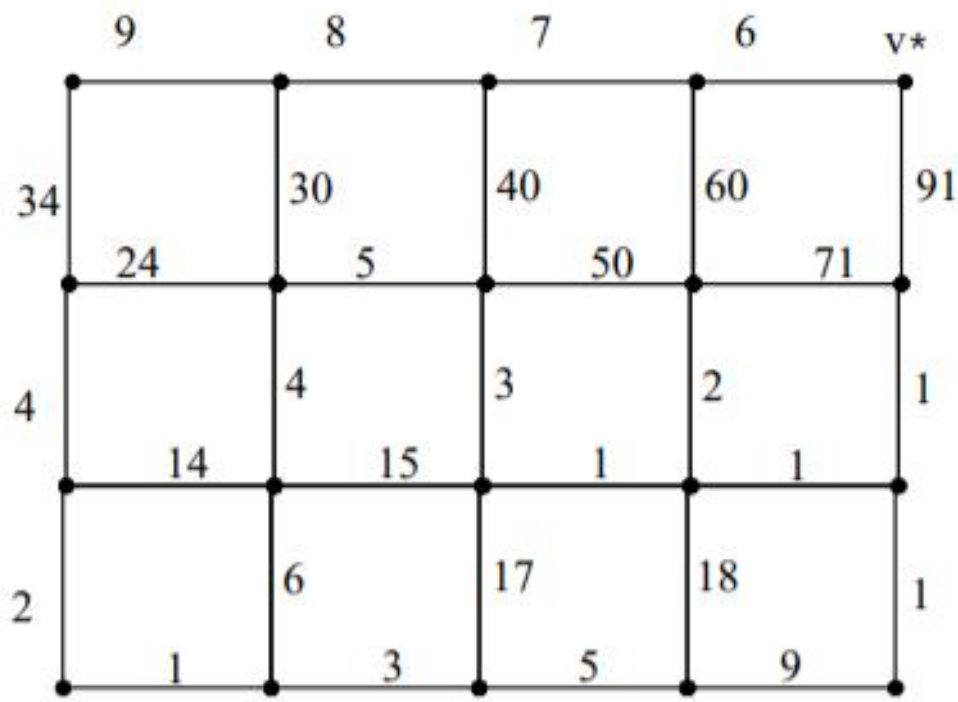



Рисунок 5.1

Розв'язання.

Будемо позначати найближчі вершини $v_1$, $v_2$, $v_3$, … у порядку їхньої появи (див. рис. 5.2): $l(v_1)=1$, $l(v_2)=2$, $l(v_3)=4$, $l(v_4)=6$, $l(v_5)=7$,

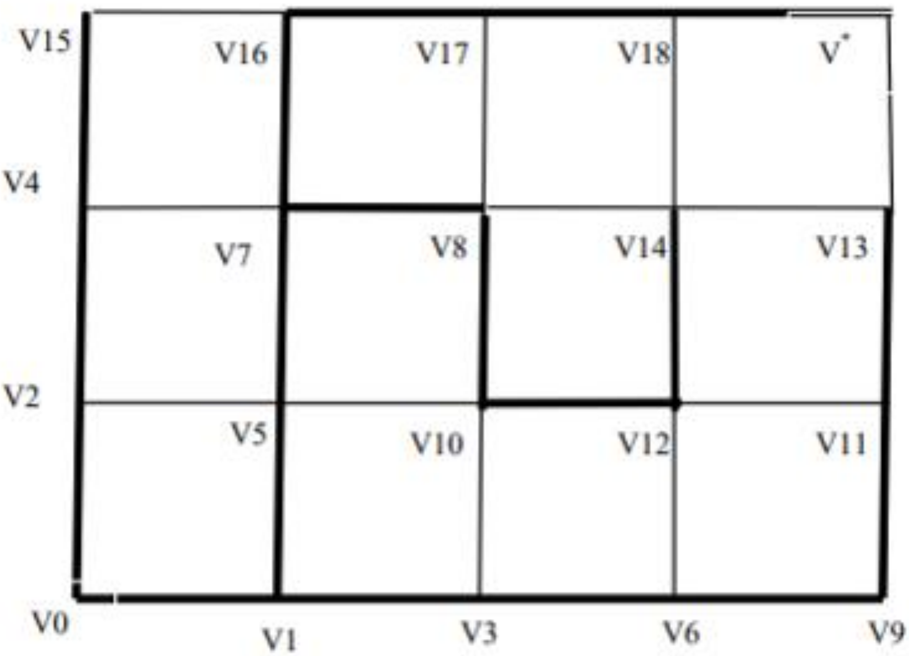



Рисунок 5.2

$l(v_6)=9$, $l(v_7)=11$, $l(v_8)=16$, $l(v_9)=18$, $l(v_{10})=19$, $l(v_{11})=19$, $l(v_{12})=20$, $l(v_{13})=20$, $l(v_{14})=22$, $l(v_{15})=40$, $l(v_{16})=41$, $l(v_{17})=49$, $l(v_{18})=56$, $l(v^*)=62$.

Дерево найближчих вершин виділено на рисунку 5.2 жирними лініями і є кістяковим деревом, тому що містить усі вершини графа.

Шуканий найкоротший ланцюг: $[v_0, v_1, v_5, v_7, v_{16}, v_{17}, v_{18}, v^*]$, довжина ланцюга $l = l(v\star)=62$.

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.

Алгоритм $\gamma$ укладання графа $G$ являє собою процес послідовного приєднання до деякого укладеного підграфа $\widetilde{G}$ графа $G$ нового ланцюга, обидва кінці якого належать $\widetilde{G}$. При цьому в якості початкового плоского графа $\widetilde{G}$ 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 планарным.

Нехай побудоване деяке укладання підграфа $\widetilde{G}$ графа $G$.

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

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

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

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

Назвемо $\alpha$-*ланцюгом* простий ланцюг L сегмента S, що містить дві різні контактні вершини і не містить інших контактних вершин.

Тепер формально опишемо алгоритм $\gamma$.

0. Виберемо деякий простий цикл С графа $G$ і укладемо його на площині; покладемо $\widetilde{G} = G$.

1. Знайдемо грані графа $\widetilde{G}$ і сегменти відносно $\widetilde{G}$. Якщо множина сегментів порожня, то перейдемо до пункту 7.

2. Для кожного сегмента S визначимо множину Г(S).

3. Якщо існує сегмент S, для якого Г(S)=∅, то граф G не планарний. Кінець. Інакше перейдемо до п. 4.

4. Якщо існує сегмент S, для якого мається єдина припустима грань Г, то перейдемо до п. 6. Інакше до п. 5.

5. Для деякого сегмента S Г(S)>1. У цьому випадку вибираємо довільну припустиму грань Г.

6. Розмістимо довільний α- ланцюг L∈S у грань Г; замінимо $\widetilde{G}$ на $\widetilde{G} \cup L$ і перейдемо до п. 1.

7. Побудовано укладання $\widetilde{G}$ графа G на площині. Кінець.

Кроком алгоритму $\gamma$ будемо вважати приєднання до $\widetilde{G}$ α- ланцюга L.

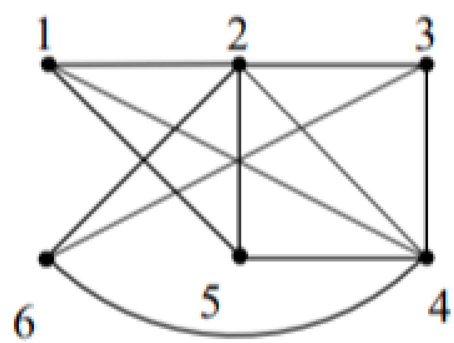


Рисунок 5.3

Приклад. Нехай граф G зображено на рисунку 5.3. Укладемо спочатку цикл C=[1, 2, 3, 4, 1], що розбиває площину на дві грані $Г_1$ і $Г_2$.

На рисунку 5.4 зображено граф $\widetilde{G} = C$ і сегменти $S_1$, $S_2$, $S_3$ відносно $\widetilde{G}$ з контактними вершинами, що обведені колами. Так як $Г(S_i) = \{Г_1, Г_2\}$ (i=1, 2, 3), то кожний α- ланцюг довільного сегмента можна укладати в будь-яку припустиму для нього грань.

Помістимо, наприклад, α- ланцюг L=[2, 5, 4] у $Г_1$. Виникає новий граф $\widetilde{G}$ і його сегменти (рис. 5.5). При цьому $Г(S_1)= \{Г_3\}$, $Г(S_2)=\{Г_1, Г_2\}$, $Г(S_3)=\{Г_1, Г_2, Г_3\}$.

Укладаємо L=(1, 5) у грань $Г_3$ (рис. 5.6).

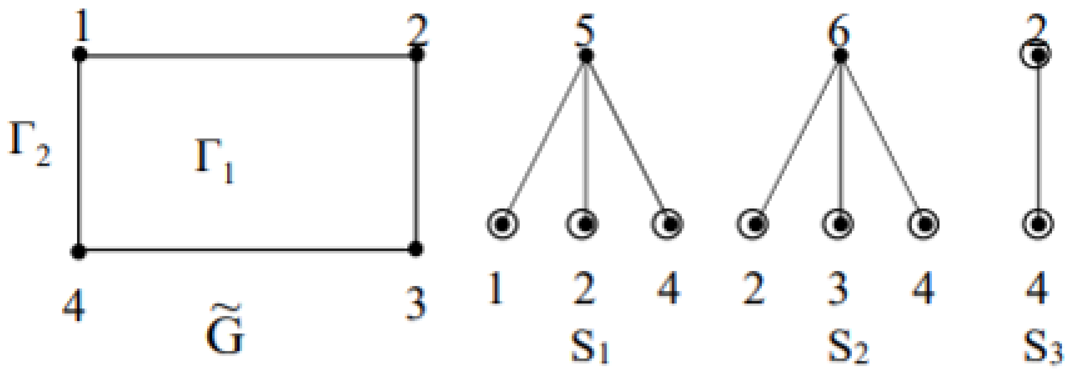



Рисунок 5.4

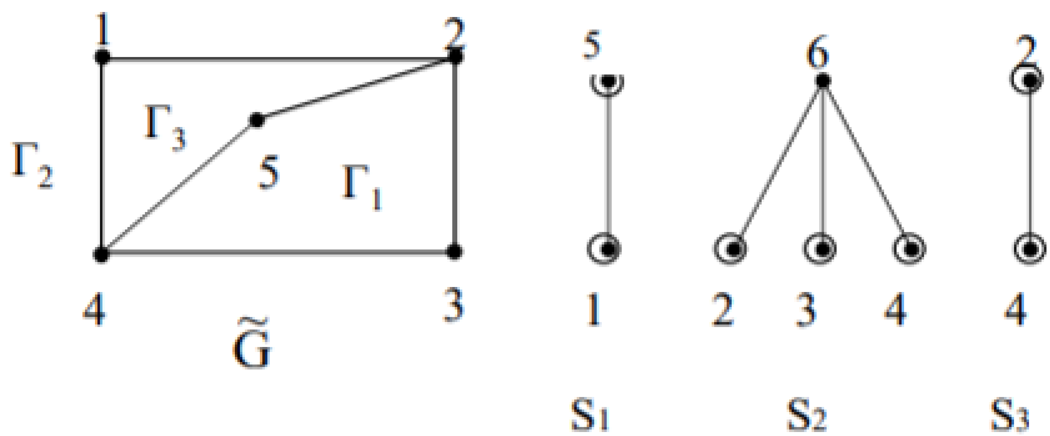
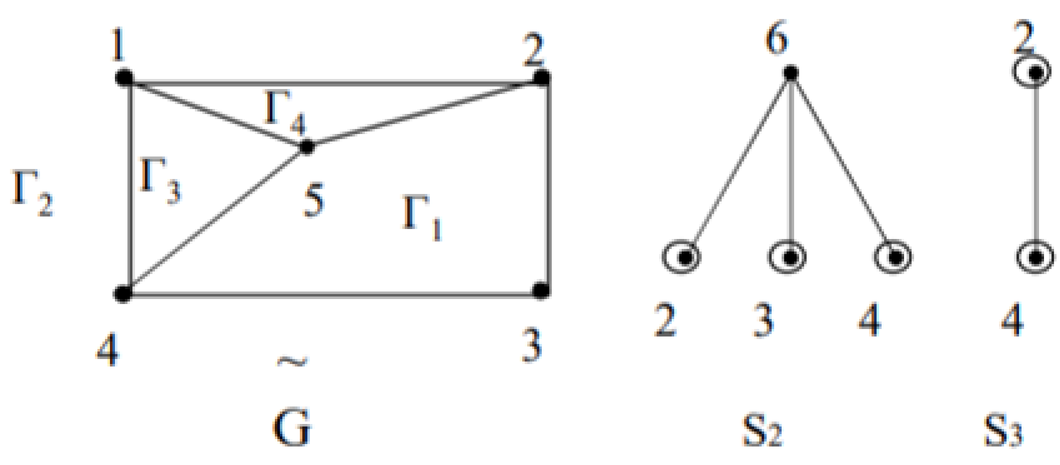



Рисунок 5.5

Тоді $\Gamma(S_1)=\{\Gamma_1, \Gamma_2\}$, $\Gamma(S_2)= \{\Gamma_1, \Gamma_2\}$. Далі укладемо α- ланцюг L=(2, 6, 4) сегмента $S_2$ у $\Gamma_1$ (рис. 5.7). У результаті маємо $\Gamma(S_2)=\{\Gamma_5\}$, $\Gamma(S_2)=\{\Gamma_1, \Gamma_2, \Gamma_5\}$. Нарешті, уклавши ребро (6, 3) у $\Gamma_5$, а ребро (2, 4) – наприклад, у $\Gamma_1$, отже $\Gamma(S_3)=\{\Gamma_1\}$. Одержуємо укладання графа G на площині (рис. 5.8).



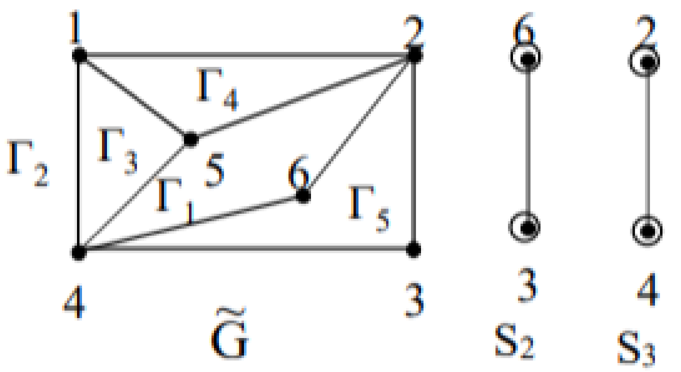



Рисунок 5.7

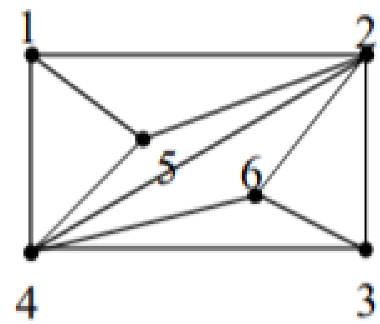


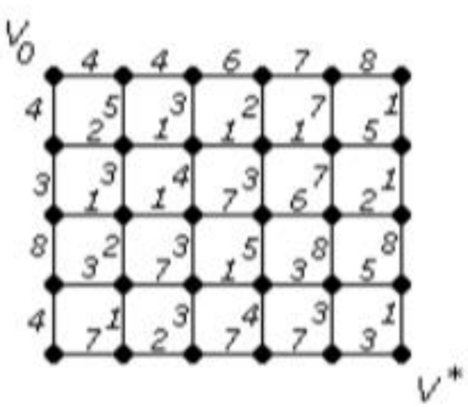

Рисунок 5.8

## Варіант №7

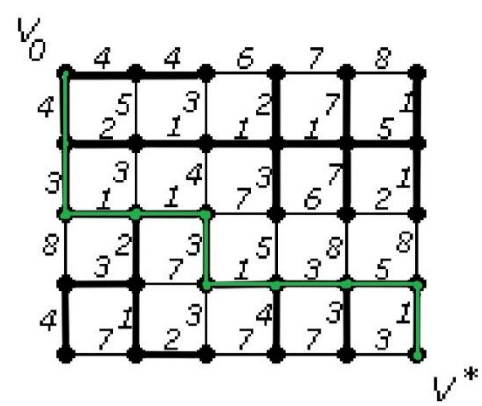
**Завдання № 1.** Розв'язати на графах наступні 2 задачі:

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

**7**




$\Rightarrow$

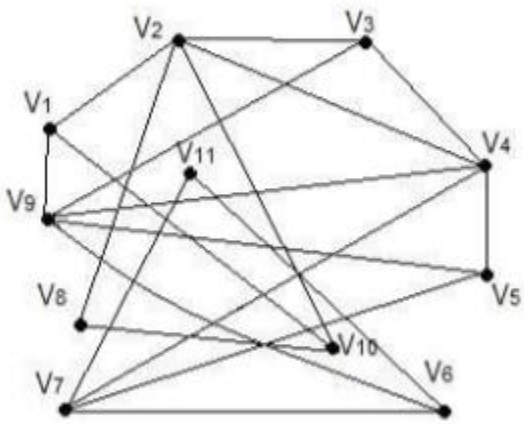



Шуканий ланцюг:{V0 V*} вага = 22

**2.** За допомогою $\gamma$-алгоритма зробити укладку графа у площині,

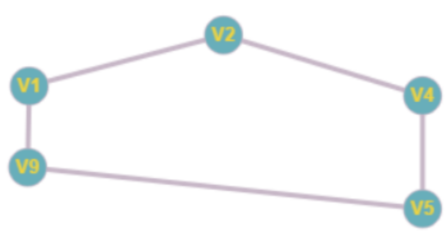
аб

**7**

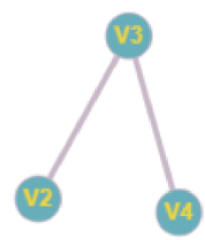



Розбиваємо на підграфи:

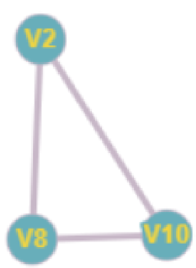
1)



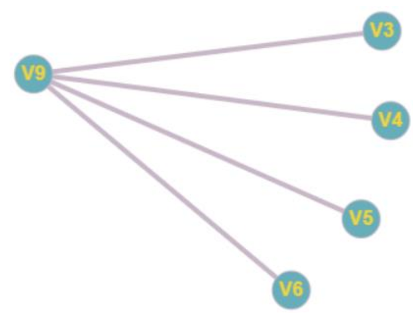

7)



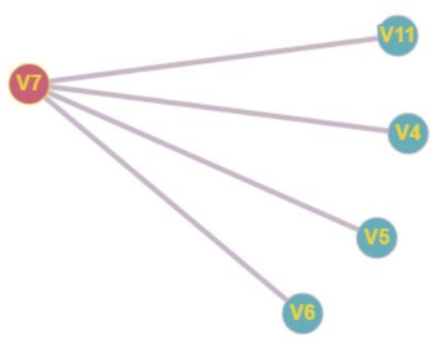

2)




3)



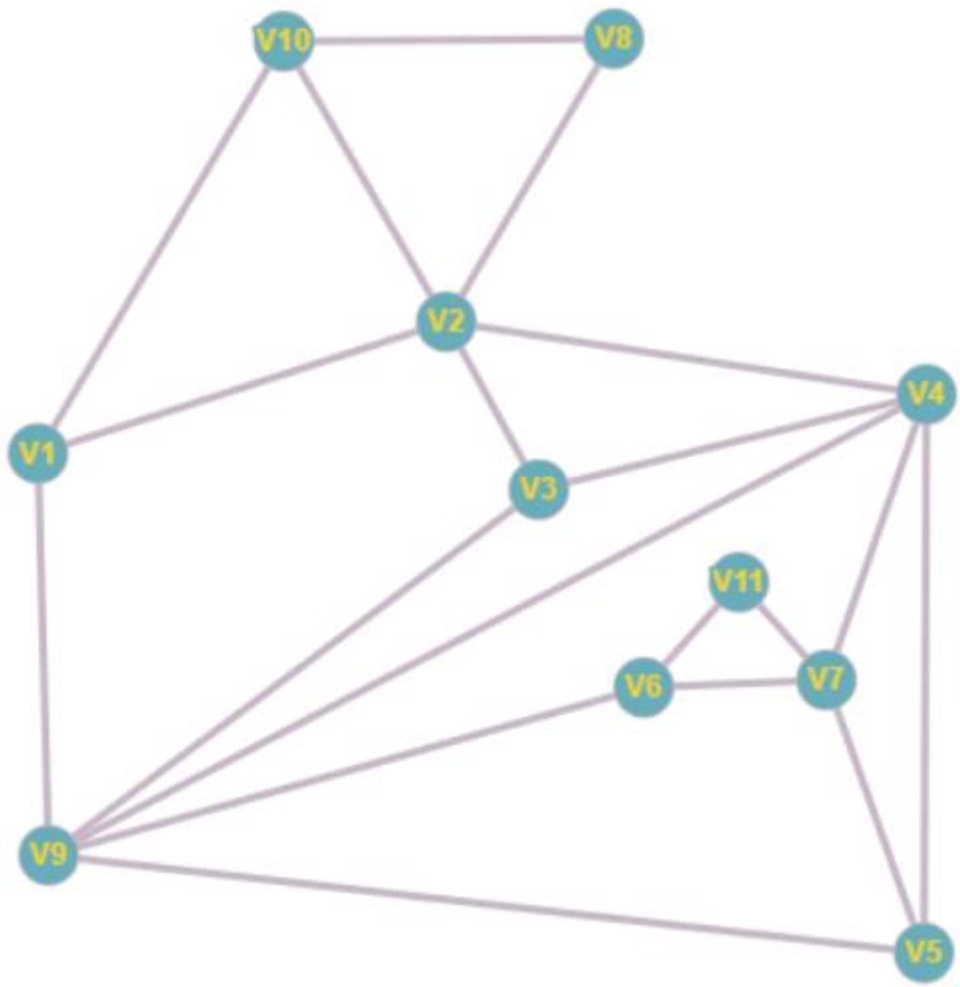

4)



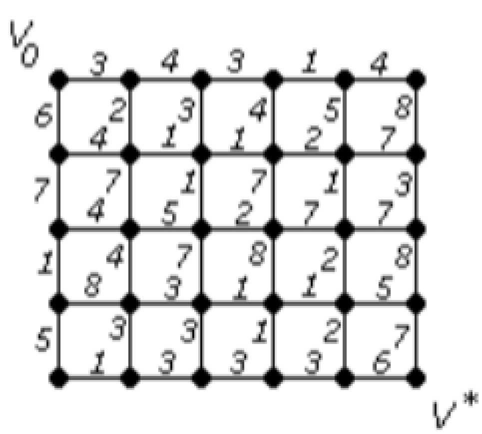

5)


6)


Методом проб і помилок),а також хитрими (і не дуже підстановками) отримуємо даний граф,який нормально розміщається в одній 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.



```c
#include <stdio.h>
#include <stdlib.h>
#define SIZE 30

int Array[SIZE][SIZE];
int distance[SIZE];
int vertex[SIZE];

void zeroArray()
{
    for(int i=0; i<SIZE; i++)
        for(int j=0; j<SIZE; j++)
           Array[i][j]=0;
}

void enterEdges()
{
    printf("Enter edges:\n");
    int r, c, n;
    for(int i=0; i<49; i++)
    {
        scanf("%d %d %d", &r, &c, &n);
        Array[r-1][c-1]=n;
        Array[c-1][r-1]=n;
    }
}
void initArray()
{
    for(int i=0; i<SIZE; i++)
    {
        distance[i]=10000;
        vertex[i]=1;
    }
    distance[0]= 0;
}

void printDistance()
{
    printf("\nShortest way to every vertex(Re(V0-V*),Img(1-30)): \n");
        for(int i=0; i<SIZE; i++)
        {
            printf("We are looking for [V*]([V29]) \n ");
            printf("V%d = %d),weight of way= %d; \n ", i, i+1, distance[i]);
            if((i+1)%30==0 && i!=0)
                printf("\n");
        }
}



int main(void)
{
    int temp;
    int minindex;
    int min;

    zeroArray();

    enterEdges();

    initArray();


    do
    {
        minindex=10000;
        min=10000;
```

```
    do
    {
        minindex=10000;
        min=10000;
        for (int i=0; i<SIZE; i++)
        {
            if ((vertex[i]==1) && (distance[i]<min))
            {
                min=distance[i];
                minindex=i;
            }
        }

        if(min!=10000)
        {
            for(int i=0; i<SIZE; i++)
            {
                if(Array[minindex][i]>0)
                {
                    temp=min+Array[minindex][i];
                    if(temp<distance[i])
                    {
                        distance[i]=temp;
                    }
                }
            }
            vertex[minindex]=0;
        }
    }
    while(minindex < 10000);

    printDistance();


    int ver[SIZE];
    int end = 29;
    ver[0] = end + 1;
    int k = 1;
    int weight = distance[end];

    while (end > 0)
    {
        for(int i=0; i<SIZE; i++)
            if (Array[end][i] != 0)
            {
                 temp = weight - Array[end][i];
                if (temp == distance[i])
                {
                    weight = temp;
                    end = i;
                    ver[k] = i + 1;
                    k++;
                }
            }
    }

    printf("\nThe shortest way:\n");
    for(int i = k-1; i>=0; i--)
        printf("%3d ", ver[i]);

    scanf("%d", &temp);
    return 0;
}
```

Висновок:на лабараторній роботі я навчився шукати найкоротший шлях з однієї вершини до іншої за допомогою алгоритму Дейкстри,а також робити укладку графа на площині.