МІНІСТЕРСТВО ОСВІТИ І НАУКИ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних наук та інформаційних технологій Кафедра систем штучного інтелекту

Лабораторна робота №4
з курсу “Дискретна математика”

Виконав: ст. гр. КН-113
Іванюшенко Нестор

Викладач: Мельникова Н.І.

[варіант 2]

Основні операції над графами. Знаходження остова мінімальної ваги за алгоритмом Пріма-Краскала

Мета роботи: набуття практичних вмінь та навичок з використання алгоритмів Пріма і Краскала.

Теоретичні відомості

Теорія графів дає простий, доступний і потужний інструмент побудови моделей прикладних задач, є ефективним засобом формалізації сучасних інженерних і наукових задач у різних областях знань.

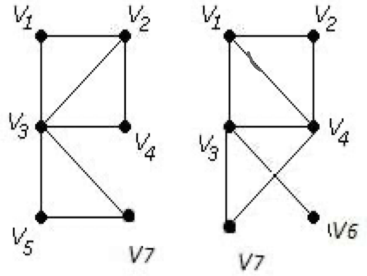
Графом G називається пара множин (V, E), де V − множина вершин, перенумерованих числами 1, 2, ..., n ? ? ; V ? ?? ?, E −

множина упорядкованих або неупорядкованих пар e = (v′,v″), v′∈V ,

v″∈V , називаних дугами або ребрами, E ={e}. При цьому не має примусового значення, як вершини розташовані в просторі або площині і які конфігурації мають ребра.

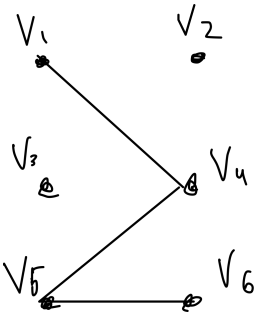
Завдання 1

2

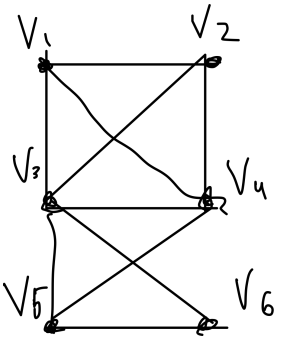


Виконати операції над графами:

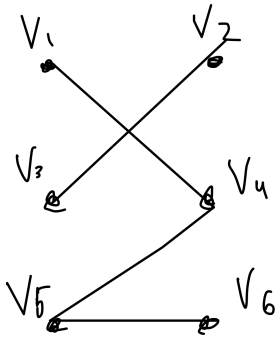
1. знайти доповнення до першого графу



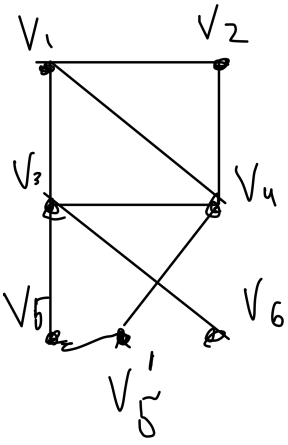
2. об’єднання графів



3. кільцеву суму G1 та G2 (G1+G2)

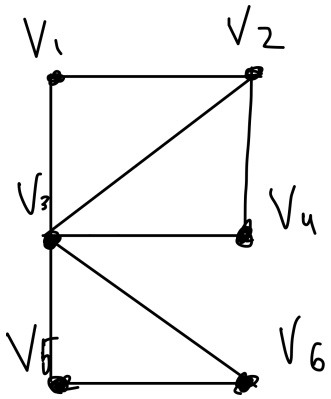


4. розщепити вершину у другому графі

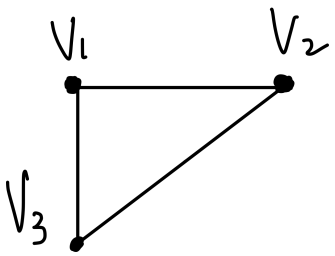


5. виділити підграф A, що складається з 3-х вершин в G1 і знайти стягнення A в G1 (G1\ A)

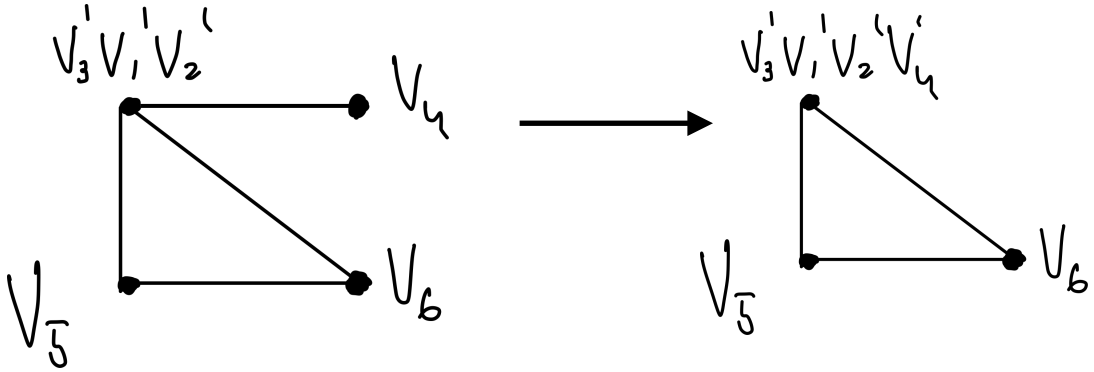
G1



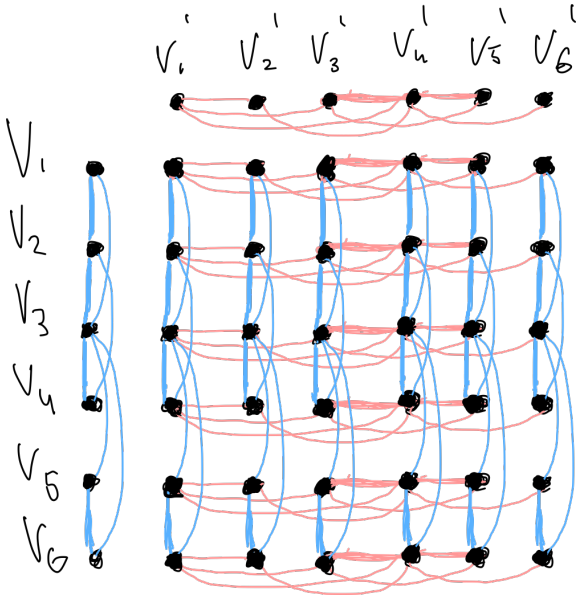
A



стягнення G1/A

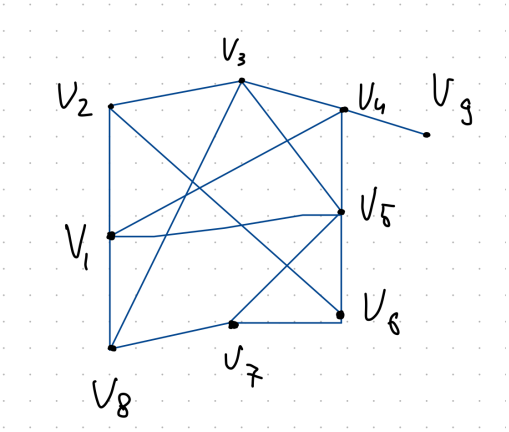


6. добуток графів.



Завдання 2

Знайти таблицю суміжності та діаметр графа



таблиця суміжності

таблиця відстаней

	V1	V2	V3	V4	V5	V6	V7	V8	V9
V1	0	1	0	1	1	0	0	1	0
V2	1	0	1	0	0	1	0	0	0
V3	0	1	0	1	1	0	0	1	0
V4	1	0	1	0	1	0	0	0	1
V5	1	0	1	1	0	1	1	0	0
V6	0	1	0	0	1	0	1	0	0
V7	0	0	0	0	1	1	0	1	0
V8	1	0	1	0	0	0	1	0	0
V9	0	0	0	1	0	0	0	0	0

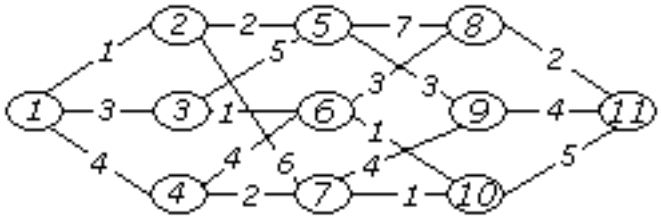
	V1	V2	V3	V4	V5	V6	V7	V8	V9
V1	0	1	2	1	1	2	2	1	2
V2	1	0	1	2	2	1	2	2	3
V3	2	1	0	1	1	2	2	1	2
V4	1	2	1	0	1	2	2	2	1
V5	1	2	1	1	0	1	1	2	2
V6	2	1	2	2	1	0	1	2	3
V7	2	2	2	2	1	1	0	1	3
V8	1	2	1	2	2	2	1	0	3
V9	2	3	2	1	2	3	3	3	0

найбільший ексцентриситет вершин - 3

отже діаметр графа - 3

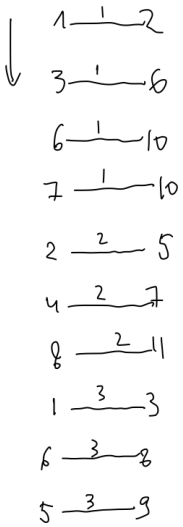
Завдання 3

Знайти двома методами (Краскала і Прима) мінімальне остове дерево графа.

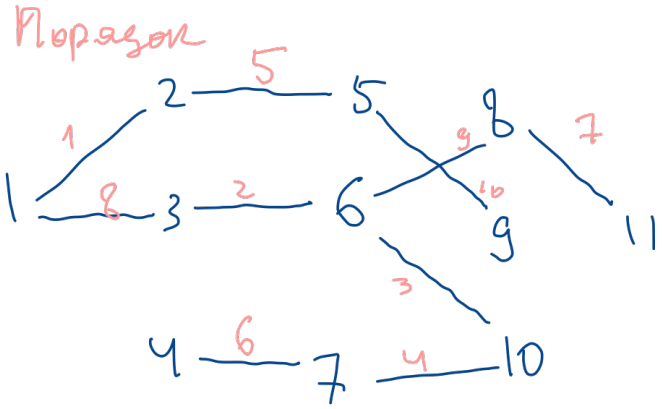


1) Краскала

Відсортовані ребра:



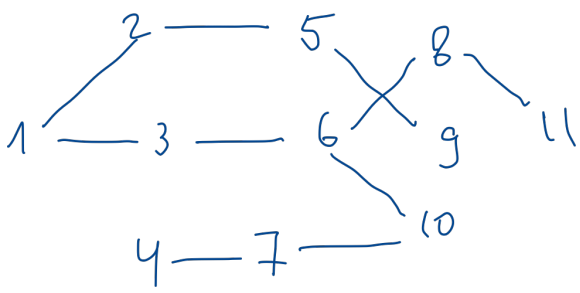
Порядок



2) Прима

Порядок:

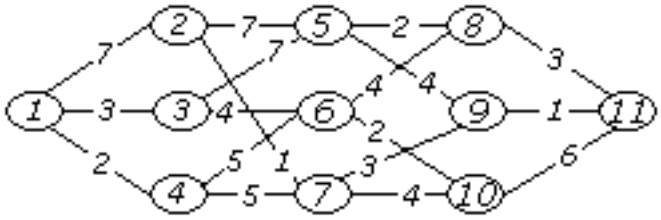
1
1–2
2–5
1–3
3–6
6–10
10–7
7–4
6–8
8–11
5–9



Частина 2

Написати програму, яка реалізує алгоритм знаходження остового дерева мінімальної ваги згідно свого варіанту.

За алгоритмом Краскала знайти мінімальне остове дерево графа. Етапи розв'язання задачі виводити на екран. Протестувати розроблену програму на наступному графі:



Програма:

```
#include ...

struct edge{
    int u, v, weight;
    edge(){};
    edge(int u_, int v_, int weight_){
        u = u_;
        v = v_;
        weight = weight_;
    };
};
void print_edge(edge e){
    std::cout << "Edge connecting " << e.u << " and " << e.v
    << ", weight: " << e.weight << std::endl;
}
int id[11];

int find(int p){
    while(p!=id[p]){
        p = id[p];
    }
    return p;
}
```

я використовую union - find алгоритм щоб перевірити граф на присутність циклу

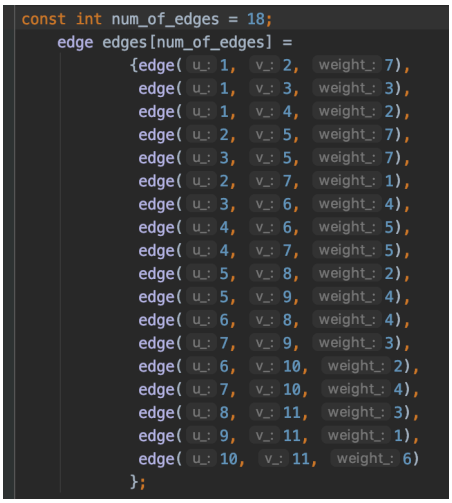
```
bool connected(int a, int b){
    return(find(a) == find(b));
}

void unio(int p, int q){
    int root1 = find(p);
    int root2 = find(q);
    if(root1 == root2){
        return;
    }
    // else merge
    id[root1] = root2;
}
```

масив айді - список груп до яких належать вершини. спочатку кожна вершина належить до власної групи

```
int main() {
    // keep track of parents / groups
    for (int i = 0; i < 12; i++) {
        id[i] = i;
    }
```

сам



граф:

сортуємо грані:

```
//sort edges by weight
bool sorted = false;
while (!sorted) {
    sorted = true;
    for (int i = 0; i < 17; i++) {
        for (int j = i + 1; j < 18; j++) {
            if (edges[j].weight < edges[i].weight) {
                sorted = false;
                edge tmp = edges[j];
                edges[j] = edges[i];
                edges[i] = tmp;
            }
        }
    }
}
```

друкуєм їх

```
    edge mst[num_of_edges];
    int mstsize = 0;
    for (edge e:edges) {
        print_edge(e);
    }

    for (edge e: edges) {
```

і починаєм алгоритм

```
     if(!connected(e.v, e.u)){
         unio(e.v, e.u);
           mst.push_back(e);
         mst[mstsize] = e;
         mstsize++;
     }
       if(mstsize >= 10){break;}
 }
```

кінець

```
    }
    std::cout << "\n\nAfter:\n";
    for (int i = 0; i<mstsize; i++) {
        print_edge(mst[i]);
    }

    return 0;
}
```

результат:

```
Edge connecting 2 and 7, weight: 1
Edge connecting 9 and 11, weight: 1
Edge connecting 5 and 8, weight: 2
Edge connecting 6 and 10, weight: 2
Edge connecting 1 and 4, weight: 2
Edge connecting 1 and 3, weight: 3
Edge connecting 8 and 11, weight: 3
Edge connecting 7 and 9, weight: 3
Edge connecting 5 and 9, weight: 4
Edge connecting 7 and 10, weight: 4
Edge connecting 6 and 8, weight: 4
Edge connecting 3 and 6, weight: 4
Edge connecting 4 and 6, weight: 5
Edge connecting 4 and 7, weight: 5
Edge connecting 10 and 11, weight: 6
Edge connecting 2 and 5, weight: 7
Edge connecting 3 and 5, weight: 7
Edge connecting 1 and 2, weight: 7


After:
Edge connecting 2 and 7, weight: 1
Edge connecting 9 and 11, weight: 1
Edge connecting 5 and 8, weight: 2
Edge connecting 6 and 10, weight: 2
Edge connecting 1 and 4, weight: 2
Edge connecting 1 and 3, weight: 3
Edge connecting 8 and 11, weight: 3
Edge connecting 7 and 9, weight: 3
Edge connecting 7 and 10, weight: 4
Edge connecting 3 and 6, weight: 4

Process finished with exit code 0
```

Висновок: я набув практичних вмінь та навичок з використання алгоритмів Пріма і Краскала.