МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Розрахунково-графічна робота

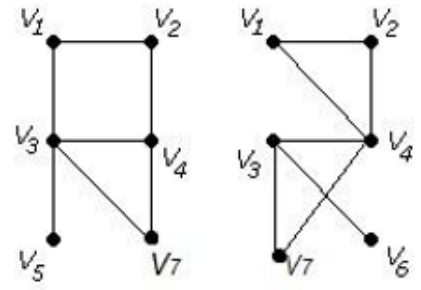
з дисципліни
«Дискретна математика»

Виконала:
студентка КН-113
Пеленська Софія
Перевірила:
Мельникова Н.І.

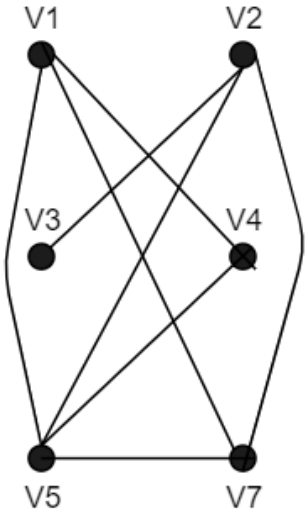
Львів – 2019 р.

Варіант № 13

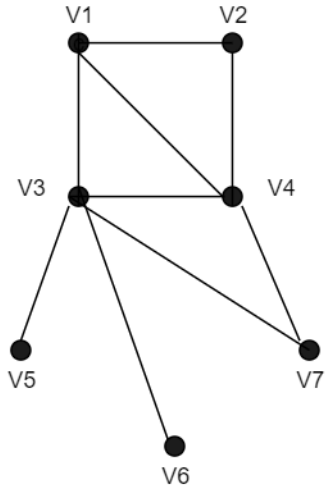
Завдання № 1 : Виконати наступні операції над графами: 1) знайти доповнення до першого графу, 2) об’єднання графів, 3) кільцеву сумму G1 та G2 (G1+G2), 4) розмножити вершину у другому графі, 5) виділити підграф А - що скадається з 3-х вершин в G1 6) добуток графів.



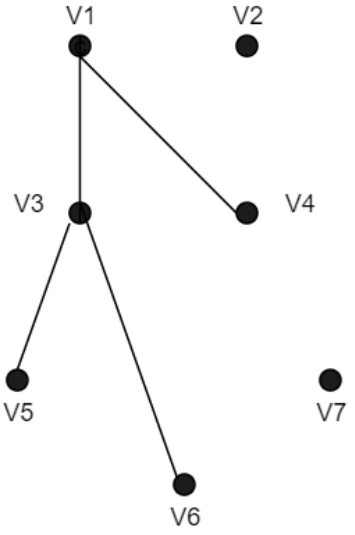
1) знайти доповнення до першого графу



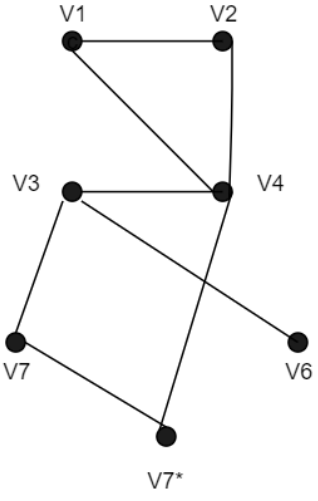
2) об’єднання графів



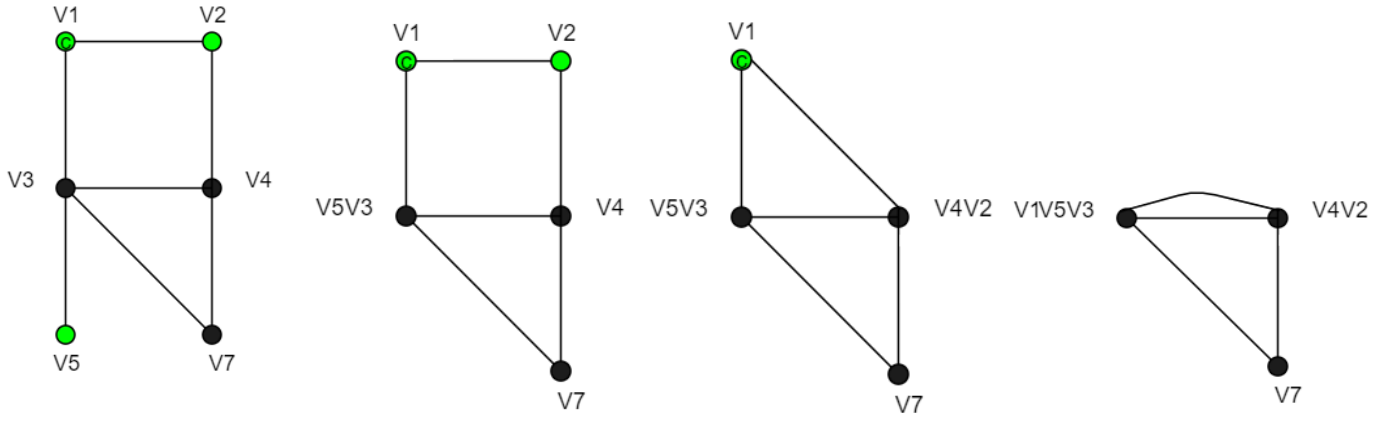
3) кільцеву сумму G1 та G2 (G1+G2)



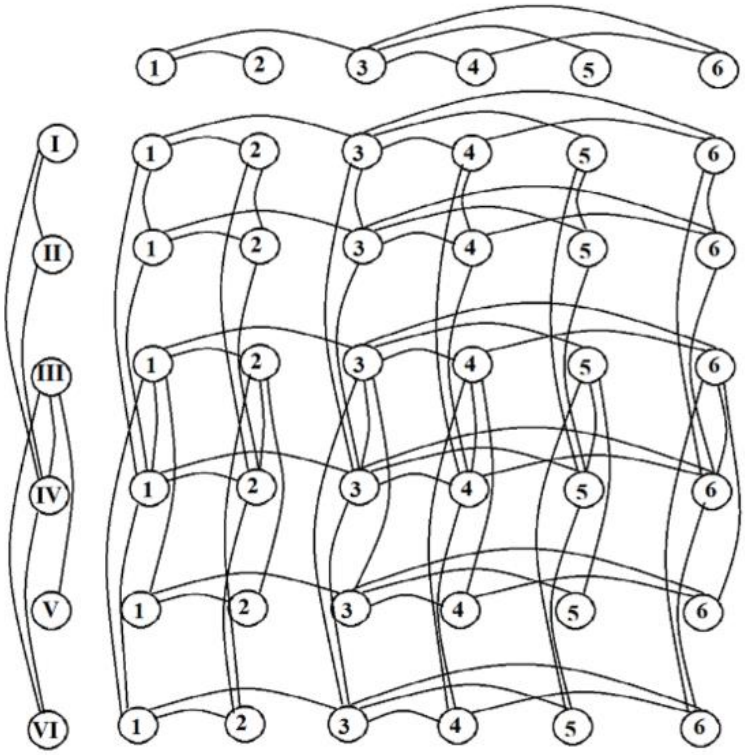
4) розмножити вершину у другому графі



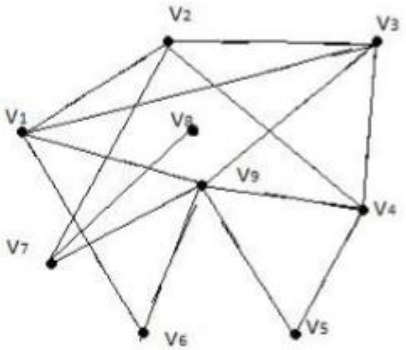
5)) виділити підграф А - що скадається з 3-х вершин в G1



6)множення графів



Завдання № 2: Скласти таблицю суміжності для неорграфа**.**



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

Завдання № 3: Для графа з другого завдання знайти діаметр.

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

Отже діаметр графа : 3

Завдання № 4: Для графа з другого завдання виконати обхід дерева **вглиб** (варіант закінчується на непарне число) або вшир (закінчується на парне число).

Вершина	номер	стек
V1	1	V1
V2	2	V1V2
V3	3	V1V2V3
V9	4	V1V2V3V4
V4	5	V1V2V3V4V9
V5	6	V1V2V3V4V9V5
-	-	V1V2V3V4V9
V6	7	V1V2V3V4V9V6
-	-	V1V2V3V4V9
V7	8	V1V2V3V4V9V7
V8	9	V1V2V3V4V9V7V8
-	-	V1V2V3V4V9V7
-	-	V1V2V3V4V9
-	-	V1V2V3V4
-	-	V1V2V3
-	-	V1V2
-	-	V1
-	-	-

```
Кількість ребер: 15
Кількість вершин: 9

Ребро 1: 1
Ребро 1: 2

Ребро 2: 1
Ребро 2: 3

Ребро 3: 1
Ребро 3: 9

Ребро 4: 1
Ребро 4: 6

Ребро 5: 2
Ребро 5: 3

Ребро 6: 2
Ребро 6: 4

Ребро 7: 2
Ребро 7: 7

Ребро 8: 3
Ребро 8: 4

Ребро 9: 3
Ребро 9: 9

Ребро 10: 4
Ребро 10: 9

Ребро 11: 4
Ребро 11: 5

Ребро 12: 5
Ребро 12: 9
```

```
Ребро 13: 6
Ребро 13: 9

Ребро 14: 7
Ребро 14: 9

Ребро 15: 7
Ребро 15: 8

Початок обходу з вершини:1
1 2
1 2 3
1 2 3 4
1 2 3 4 9
1 2 3 4 9 5
1 2 3 4 9
1 2 3 4 9 6
1 2 3 4 9
1 2 3 4 9 7
1 2 3 4 9 7 8
1 2 3 4 9 7
1 2 3 4 9
1 2 3 4
1 2 3
1 2
1
Стек пустий

Process returned 0 (0x0)   execution time : 73.623 s
Press any key to continue.
```

Код програми:

```
1   #include <iostream>
2   #include <string>
3   #include <sstream>
4
5   using namespace std;
6
7   struct Graf{
8     bool vglub = false;
9   };
10  struct edge{
11    int vert1;
12    int vert2;
13  };
14
15
16  int leng(string str){
17    int i = 0;
18    while (str[i] != '\0'){
19      i++;
20    }
21    return i;
22  }
23
24
25  int perevirka(int m, int n){
26    int c = 0;
27    bool check = false;
```

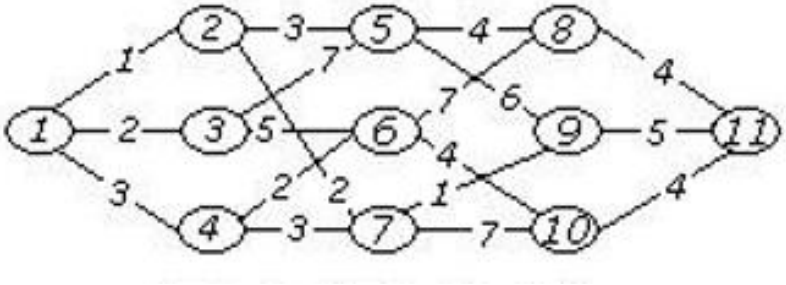
```
28   string str;
29   stringstream ss;
30   while (check == false)
31   {
32    cin >> str;
33    for (int i = 0; i < leng(str); i++){
34     if (!isdigit(str[i])){
35      if (i == 0 && str[i] == '-') check = true;
36      else{
37       check = false;
38       break;}
39     }
40     else
41      check = true;}
42
43    if (check == true){
44     ss << str;
45     ss >> c;
46     ss.clear();
47
48     if (c < m || c > n) check = false;
49     else
50      check = true;}
51    if (check == false) cout << "Error! Try again!" << endl;
52      str = "";}
53   return c;
54  }
```

```
55
56  void vvid(edge *reb, int n, int m){
57   for (int i = 0; i < n; i++){
58    cout << "Ребро " << i + 1 << ": ";
59    reb[i].vert1 = perevirka(1, m);
60    cout << "Ребро " << i + 1 << ": ";;
61    reb[i].vert2 = perevirka(1, m);
62    cout << endl;
63   }
64  }
65
66  int main()
67  {
68   setlocale(LC_ALL, "Ukrainian");
69   int n, m, p ,start;
70   int counter = 0;
71   int t = 0;
72   int head = 0;
73
74   cout << "Кількість ребер: ";
75   n = perevirka(1, 1000);
76   cout << "Кількість вершин: ";
77   m = perevirka(1, 1000);
78   cout << endl;
79
80   int *vec = new int[m];
81   edge *rbr = new edge[n];
```

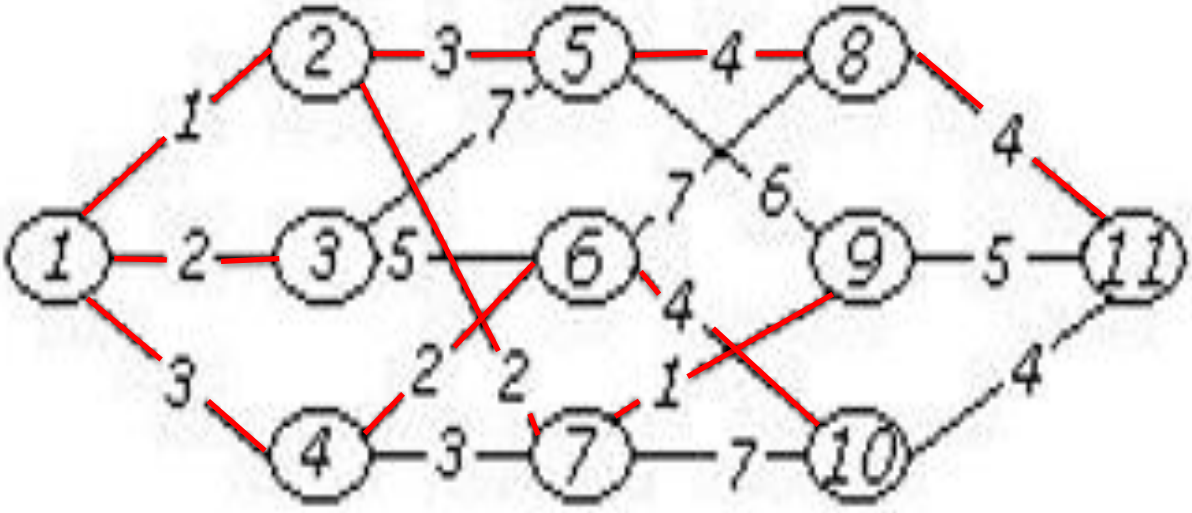
```
83
84  vvid(rbr, n, m);
85
86   cout << "Початок обходу з вершини:";
87   start = perevirka(1, m);
88
89   vec[0] = start;
90   graf[start - 1].vglub = true;
91   counter++;
92
93     while (counter != 0){
94      for (int i = 0; i < n; i++){
95  if((vec[counter - 1] == rbr[i].vert1 && graf[rbr[i].vert2 - 1].vglub == false) || (vec[counter - 1] == rbr[i].vert2 && graf[rbr[i].v
96        t++;}
97      }
98
99      if (t == 0) counter--;
100     else{
101      for (int i = 0; i < n; i++){
102       if (vec[counter - 1] == rbr[i].vert2 && graf[rbr[i].vert1 - 1].vglub == false){
103        vec[counter] = rbr[i].vert1;
104        graf[rbr[i].vert1 - 1].vglub = true;
105        counter++;
106        goto point;}
107
108      if (vec[counter - 1] == rbr[i].vert1 && graf[rbr[i].vert2 - 1].vglub == false){
```

```
109         vec[counter] = rbr[i].vert2;
110         graf[rbr[i].vert2 - 1].vglub = true;
111         counter++;
112         goto point;}
113       }
114     }
115   point:;
116    for (int i = 0; i < counter; i++)cout << vec[i] << " ";
117
118    if (counter != 0)cout << endl;
119    t = 0;}
120   cout << "Стек пустий" << endl;
121 }
122
```

Завдання № 5: Знайти двома методами (Краскала і Прима) мінімальне остове дерево графа.



1) Краскала:



V = {1, 2, 7, 9, 3, 4, 6, 5, 8,10,11}
E = {(1,2) (7,9) (1,3) (2,7) (4,6) (1,4) (2,5) (5,8) (6,10) (8,11)**}**

```
Number of edges : 18
Number of vertex: 11

Weight 1 edge : 1
Vertex 1 : 1
Vertex 2 : 2

Weight 2 edge : 2
Vertex 1 : 1
Vertex 2 : 3

Weight 3 edge : 3
Vertex 1 : 1
Vertex 2 : 4

Weight 4 edge : 3
Vertex 1 : 2
Vertex 2 : 5

Weight 5 edge : 2
Vertex 1 : 2
Vertex 2 : 7

Weight 6 edge : 7
Vertex 1 : 3
Vertex 2 : 5

Weight 7 edge : 5
Vertex 1 : 3
Vertex 2 : 6

Weight 8 edge : 2
Vertex 1 : 4
Vertex 2 : 6

Weight 9 edge : 3
Vertex 1 : 4
Vertex 2 : 7
```

```
Weight 10 edge : 4
Vertex 1 : 5
Vertex 2 : 8

Weight 11 edge : 6
Vertex 1 : 5
Vertex 2 : 9

Weight 12 edge : 7
Vertex 1 : 6
Vertex 2 : 8

Weight 13 edge : 4
Vertex 1 : 6
Vertex 2 : 10

Weight 14 edge : 1
Vertex 1 : 7
Vertex 2 : 9

Weight 15 edge : 7
Vertex 1 : 7
Vertex 2 : 10

Weight 16 edge : 4
Vertex 1 : 8
Vertex 2 : 11

Weight 17 edge : 5
Vertex 1 : 9
Vertex 2 : 11

Weight 18 edge : 4
Vertex 1 : 10
Vertex 2 : 11
```

```
"C:\Users\Lenovo\Desktop\1ъєЁё (1 ёхьхё€Ё)\–шёъЁх€ър ╥рсш\ъЁрёъ
1 -> 2
7 -> 9
1 -> 3
2 -> 7
4 -> 6
1 -> 4
2 -> 5
5 -> 8
6 -> 10
8 -> 11
Weight of tree : 26
Process returned 0 (0x0)   execution time : 90.255 s
Press any key to continue.
```

Код програми:

```cpp
#include <iostream>
#include <stdio.h>
using namespace std;
struct graf {

 int dov;
 int vert1;
 int vert2;
 bool vis = false;
};
struct masive {

 int arr[11] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
 int c = 0;
};

int vvid(graf *reb, int n) {
 for (int i = 0; i < n; i++)
 {
  cout << "Weight " << i + 1 << " edge : ";
  cin >> reb[i].dov;
  cout << "Vertex 1 : " ;
  cin >> reb[i].vert1;
  cout << "Vertex 2 : ";
  cin >> reb[i].vert2;
  cout << endl;
```

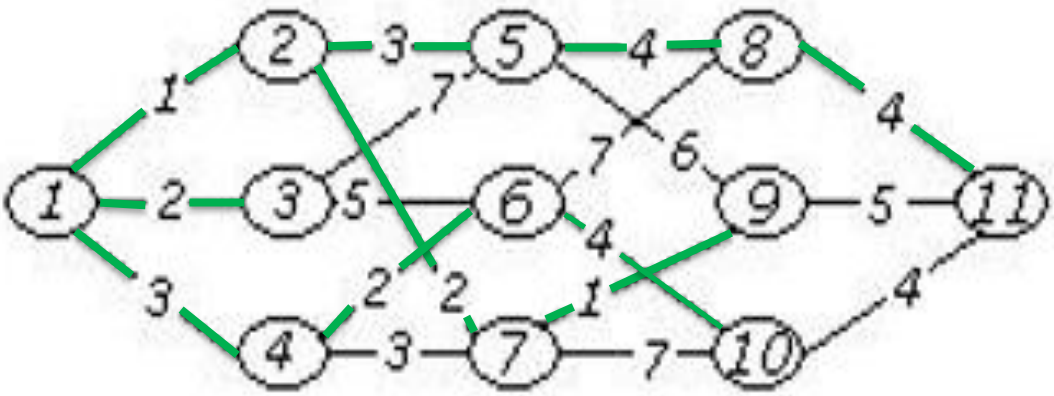
```
27    }
28    return 0;
29  }
30
31  int main() {
32    int x = 100;
33    int y = 100;
34    int n, z;
35    cout << "Number of edges : ";
36    cin >> n;
37    cout << "Number of vertex: ";
38    cin >> z;
39    cout << endl;
40
41    graf *rebro = new graf[n];
42    masive mas[5];
43
44    vvid(rebro, n);
45
46    for (int i = 0; i < n - 1; i++){
47     for (int j = 0; j < n - 1; j++){
48      if (rebro[j].dov > rebro[j + 1].dov) {
49          swap(rebro[j].dov, rebro[j + 1].dov);
50          swap(rebro[j].vert1, rebro[j + 1].vert1);
51          swap(rebro[j].vert2, rebro[j + 1].vert2); }
52    }
```

```
53    }
54
55    int c = -1;
56    for (int i = 0; i < n; i++){
57     for (int j = 0; j < 5; j++){
58      for (int k = 0; k < z; k++){
59       if (rebro[i].vert1 == mas[j].arr[k]) { x = j;
60       goto point0;;
61       }
62      }
63     }
64    point0:;
65
66     for (int j = 0; j < 5; j++){
67      for (int k = 0; k <z; k++){
68       if (rebro[i].vert2 == mas[j].arr[k]) { y = j;
69       goto point1;
70       }
71      }
72     }
73    point1:;
74     if (x != y && x == 100) {
75           mas[y].arr[mas[y].c] = rebro[i].vert1;
76           mas[y].c++;
77           }
78
```

```
79     if (x != y && y == 100) {
80           mas[x].arr[mas[x].c] = rebro[i].vert2;
81           mas[x].c++; }
82
83     if (x != y && x != 100 && y != 100) {
84      if (x < y) {
85       for (int l = 0; l < mas[y].c; l++){
86        mas[x].arr[mas[x].c+l] = mas[y].arr[l];
87        mas[y].arr[l] = 0;
88       }
89       mas[x].c += mas[y].c;
90       mas[y].c = 0;
91      }
92
93      if (y < x) {
94       for (int l = 0; l < mas[x].c; l++){
95        mas[y].arr[mas[y].c+l] = mas[x].arr[l];
96        mas[x].arr[l] = 0;
97       }
98       mas[y].c += mas[x].c;
99       mas[x].c = 0;
100     }
101    }
102    if (x == 100 && y == 100) {
103          c++;
104          mas[c].arr[mas[c].c] = rebro[i].vert1;
```

```
105         mas[c].arr[mas[c].c + 1] = rebro[i].vert2;
106         mas[c].c += 2;
107         }
108
109     rebro[i].vis = true;
110
111     if (x == y && x != 100) {
112         rebro[i].vis = false;
113     }
114     x = y = 100;
115
116   cout << "Result: " << endl;
117
118   int suma = 0;
119
120   for (int i = 0; i < n; i++){
121     if (rebro[i].vis == true) {
122         cout << rebro[i].vert1 << " -> " << rebro[i].vert2 << endl;
123         suma += rebro[i].dov; }
124   }
125   cout << "Weight of tree : " << suma;
126 }
127 }
```

2) Прима



V = {1, 2, 3, 7, 9, 4, 6, 5, 8, 10, 11}

E = {(1,2) (1,3) (2,7) (7,9) (1,4) (4,6) (2,5) (5,8) (6,10) (8,11)}

```
Tree :
1 -> 2
1 -> 3
2 -> 7
7 -> 9
1 -> 4
4 -> 6
2 -> 5
5 -> 8
6 -> 10
8 -> 11

Process returned 0 (0x0)   execution time : 0.152 s
Press any key to continue.
```

Код програми :

```cpp
#include <iostream>
#include <cstring>
using namespace std;

int main () {
    int siz = 11;

  int Graf[siz][siz] = {
  {0,1,2,3,0,0,0,0,0,0,0},
  {1,0,0,0,3,0,2,0,0,0,0},
  {2,0,0,0,7,5,0,0,0,0,0},
  {3,0,0,0,0,2,3,0,0,0,0},
  {0,3,7,0,0,0,0,4,6,0,0},
  {0,0,5,2,0,0,0,7,0,4,0},
  {0,2,0,3,0,0,0,0,1,7,0},
  {0,0,0,0,4,7,0,0,0,0,4},
  {0,0,0,0,6,0,1,0,0,0,5},
  {0,0,0,0,0,4,7,0,0,0,4},
  {0,0,0,0,0,0,0,4,5,4,0}
};


  int rebro = 0;
  int visited[siz];
  memset (visited, false, sizeof (visited));

  visited[0] = true;
  int x=0,y=0;

  cout << "Tree :"<<endl;
  while (rebro < siz - 1) {

  int minimum = 1000000000;

  for (int i = 0; i < siz; i++) {
    if (visited[i]) {
        for (int j = 0; j < siz; j++) {
            if (!visited[j] && Graf[i][j]) {
                if (minimum > Graf[i][j]) {
                    minimum = Graf[i][j];
                    x = i;
                    y = j;
                }

            }
        }
    }
  }
  cout << x+1 <<  " -> " << y+1<<endl;
  visited[y] = true;
  rebro++;
  }

return 0;
}
```

Завдання № 6: Розв’язати задачу комівояжера для повного 8-ми вершинного графа методом «іди у найближчий», матриця вагів якого має вигляд:

	1	2	3	4	5	6	7	8
1	∞	1	5	1	5	1	6	1
2	1	∞	7	5	6	1	2	3
3	5	7	∞	5	6	2	1	2
4	1	5	5	∞	6	5	1	5
5	5	6	6	6	∞	7	7	7
6	1	1	2	5	7	∞	1	1
7	6	2	1	1	7	1	∞	2
8	1	3	2	5	7	1	2	∞

Перевіримо декілька шляхів

	1	2	3	4	5	6	7	8
1	∞	1	5	1	5	1	6	1
2	1	∞	7	5	6	1	**2**	3
3	5	7	∞	5	6	2	1	2
4	1	5	5	∞	6	5	1	5
5	5	6	6	6	∞	7	7	7
6	1	1	2	5	7	∞	1	1
7	6	**2**	1	1	7	1	∞	2
8	1	3	2	5	7	1	2	∞

	1	3	4	5	6	7,2	8
1	∞	5	1	5	1	6	1
3	5	∞	5	6	2	**1**	2
4	1	5	∞	6	5	1	5
5	5	6	6	∞	7	7	7
6	1	2	5	7	∞	1	1
7,2	6	**1**	1	7	1	∞	2
8	1	2	5	7	1	2	∞

	1	2,7,3	4	5	6	8
1	∞	5	1	5	1	1
2,7,3	5	∞	5	6	**2**	2
4	1	5	∞	6	5	5
5	5	6	6	∞	7	7
6	1	**2**	5	7	∞	1
8	1	2	5	7	1	∞

	1	4	5	2,7,3,6	8
1	∞	1	5	1	1
4	1	∞	6	5	5
5	5	6	∞	7	7
2,7,3,6	1	5	7	∞	**1**
8	1	5	7	**1**	∞

	1	4	5	2,7,3,6,8
1	∞	1	5	**1**
4	1	∞	6	5
5	5	6	∞	7
2,7,3,6,8	**1**	5	7	∞

	2,7,3,6,8,1	4	5
2,7,3,6,8,1	∞	**1**	5
4	**1**	∞	6
5	5	6	∞

	2,7,3,6,8,1,4	5
2,7,3,6,8,1,4	∞	6
5	6	∞

Отже шлях :2 → 7 → 3 → 6 → 8 → 1 → 4 → 5 → 2
і його вага : 2+1+2+1+1+1+6+6=20

Перевіримо ще один шлях (почнемо його з іншої вершини):

	1	2	3	4	5	6	7	8
1	∞	1	5	1	5	1	6	1
2	1	∞	7	5	6	1	2	3
3	5	7	∞	5	6	2	1	2
4	1	5	5	∞	6	5	**1**	5
5	5	6	6	6	∞	7	7	7
6	1	1	2	5	7	∞	1	1
7	6	2	1	**1**	7	1	∞	2
8	1	3	2	5	7	1	2	∞

	1	2	3	5	6	4,7	8
1	∞	1	5	5	1	6	1
2	1	∞	7	6	1	**2**	3
3	5	7	∞	6	2	1	2
5	5	6	6	∞	7	7	7
6	1	1	2	7	∞	1	1
4,7	6	**2**	1	7	1	∞	2
8	1	3	2	7	1	2	∞

	1	4,7,2	3	5	6	8
1	∞	1	5	5	1	1
4,7,2	1	∞	7	6	**1**	3
3	5	7	∞	6	2	2
5	5	6	6	∞	7	7
6	1	**1**	2	7	∞	1
8	1	3	2	7	1	∞

	1	3	5	4,7,2,6	8
1	∞	5	5	1	1
3	5	∞	6	2	2
5	5	6	∞	7	7
4,7,2,6	1	2	7	∞	**1**
8	1	2	7	**1**	∞

	1	3	5	4,7,2,6,8
1	∞	5	5	**1**
3	5	∞	6	2
5	5	6	∞	7
4,7,2,6,8	**1**	2	7	∞

	4,7,2,6,8,1	3	5
4,7,2,6,8,1	∞	**5**	5
3	5	∞	6
5	**5**	6	∞

	4,7,2,6,8,1,3	5
4,7,2,6,8,1,3	∞	6
5	6	∞

Отже шлях :4 → 7 → 2 → 6 → 8 → 1 → 3 → 5 → 4
і його вага : 1+2+1+1+1+6+6+6=23

Перевіримо наступний цикл:

	1	2	3	4	5	6	7	8
1	∞	**1**	5	1	5	1	6	1
2	**1**	∞	7	5	6	1	2	3
3	5	7	∞	5	6	2	1	2
4	1	5	5	∞	6	5	1	5
5	5	6	6	6	∞	7	7	7
6	1	1	2	5	7	∞	1	1
7	6	2	1	1	7	1	∞	2
8	1	3	2	5	7	1	2	∞

	1,2	3	4	5	6	7	8
1,2	∞	7	5	6	**1**	2	3
3	7	∞	5	6	2	1	2
4	5	5	∞	6	5	1	5
5	6	6	6	∞	7	7	7
6	**1**	2	5	7	∞	1	1
7	2	1	1	7	1	∞	2
8	3	2	5	7	1	2	∞

	3	4	5	1,2,6	7	8
3	∞	5	6	2	1	2
4	5	∞	6	5	1	5
5	6	6	∞	7	7	7
1,2,6	2	5	7	∞	1	**1**
7	1	1	7	1	∞	2
8	2	5	7	**1**	2	∞

	3	4	5	7	1,2,6,8
3	∞	5	6	1	**2**
4	5	∞	6	1	5
5	6	6	∞	7	7
7	1	1	7	∞	2
1,2,6,8	**2**	5	7	2	∞

	1,2,6,8,3	4	5	7
1,2,6,8,3	∞	5	6	**1**
4	5	∞	6	1
5	6	6	∞	7
7	**1**	1	7	∞

	4	5	1,2,6,8,3,7
4	∞	6	**1**
5	6	∞	7
1,2,6,8,3,7	**1**	7	∞

	1,2,6,8,3,7,4	5
1,2,6,8,3,7,4	∞	6
5	6	∞

Отже шлях :1 → 2 → 6 → 8 → 3 → 7 → 4 → 5 → 1 і його вага : 1+1+1+2+1+1+6+5=18

Цей шлях є найменшим , так як циклів є велика кількість перевіримо усі можливі програмно :

```
7-> 2-> 5-> 4-> 1-> 6-> 8-> 3-> 7 (20)
7-> 2-> 5-> 4-> 1-> 8-> 3-> 6-> 7 (21)
7-> 2-> 5-> 4-> 1-> 8-> 6-> 3-> 7 (20)
7-> 2-> 5-> 4-> 3-> 1-> 6-> 8-> 7 (28)
7-> 2-> 5-> 4-> 3-> 1-> 8-> 6-> 7 (27)
7-> 2-> 5-> 4-> 3-> 6-> 1-> 8-> 7 (25)
7-> 2-> 5-> 4-> 3-> 6-> 8-> 1-> 7 (25)
7-> 2-> 5-> 4-> 3-> 8-> 1-> 6-> 7 (24)
7-> 2-> 5-> 4-> 3-> 8-> 6-> 1-> 7 (24)
7-> 2-> 5-> 4-> 6-> 1-> 3-> 8-> 7 (29)
7-> 2-> 5-> 4-> 6-> 1-> 8-> 3-> 7 (24)
7-> 2-> 5-> 4-> 6-> 3-> 1-> 8-> 7 (29)
7-> 2-> 5-> 4-> 6-> 3-> 8-> 1-> 7 (29)
7-> 2-> 5-> 4-> 6-> 8-> 1-> 3-> 7 (27)
7-> 2-> 5-> 4-> 6-> 8-> 3-> 1-> 7 (27)
7-> 2-> 5-> 4-> 8-> 1-> 3-> 6-> 7 (28)
7-> 2-> 5-> 4-> 8-> 1-> 6-> 3-> 7 (24)
7-> 2-> 5-> 4-> 8-> 3-> 1-> 6-> 7 (28)
7-> 2-> 5-> 4-> 8-> 3-> 6-> 1-> 7 (28)
7-> 2-> 5-> 4-> 8-> 6-> 1-> 3-> 7 (27)
7-> 2-> 5-> 4-> 8-> 6-> 3-> 1-> 7 (27)
7-> 2-> 5-> 6-> 1-> 3-> 4-> 8-> 7 (33)
7-> 2-> 5-> 6-> 1-> 3-> 8-> 4-> 7 (29)
7-> 2-> 5-> 6-> 1-> 4-> 3-> 8-> 7 (26)
7-> 2-> 5-> 6-> 1-> 4-> 8-> 3-> 7 (25)
7-> 2-> 5-> 6-> 1-> 8-> 3-> 4-> 7 (25)
7-> 2-> 5-> 6-> 1-> 8-> 4-> 3-> 7 (28)
7-> 2-> 5-> 6-> 3-> 1-> 4-> 8-> 7 (30)
7-> 2-> 5-> 6-> 3-> 1-> 8-> 4-> 7 (29)
7-> 2-> 5-> 6-> 3-> 4-> 1-> 8-> 7 (26)
7-> 2-> 5-> 6-> 3-> 4-> 8-> 1-> 7 (26)
7-> 2-> 5-> 6-> 3-> 8-> 1-> 4-> 7 (22)
7-> 2-> 5-> 6-> 3-> 8-> 4-> 1-> 7 (22)
7-> 2-> 5-> 6-> 4-> 1-> 3-> 8-> 7 (30)
7-> 2-> 5-> 6-> 4-> 1-> 8-> 3-> 7 (25)
7-> 2-> 5-> 6-> 4-> 3-> 1-> 8-> 7 (33)
7-> 2-> 5-> 6-> 4-> 3-> 8-> 1-> 7 (33)
7-> 2-> 5-> 6-> 4-> 8-> 1-> 3-> 7 (32)
7-> 2-> 5-> 6-> 4-> 8-> 3-> 1-> 7 (32)
7-> 2-> 5-> 6-> 8-> 1-> 3-> 4-> 7 (28)
7-> 2-> 5-> 6-> 8-> 1-> 4-> 3-> 7 (24)
```

```
"C:\Users\Lenovo\Desktop\1ъєЁё (1 ёхьхёЄЁ)\–шёъ
7-> 2-> 5-> 6-> 8-> 3-> 1-> 4-> 7 (25)
7-> 2-> 5-> 6-> 8-> 3-> 4-> 1-> 7 (25)
7-> 2-> 5-> 6-> 8-> 4-> 1-> 3-> 7 (28)
7-> 2-> 5-> 6-> 8-> 4-> 3-> 1-> 7 (28)
7-> 2-> 5-> 8-> 1-> 3-> 4-> 6-> 7 (32)
7-> 2-> 5-> 8-> 1-> 3-> 6-> 4-> 7 (29)
7-> 2-> 5-> 8-> 1-> 4-> 3-> 6-> 7 (25)
7-> 2-> 5-> 8-> 1-> 4-> 6-> 3-> 7 (25)
7-> 2-> 5-> 8-> 1-> 6-> 3-> 4-> 7 (25)
7-> 2-> 5-> 8-> 1-> 6-> 4-> 3-> 7 (28)
7-> 2-> 5-> 8-> 3-> 1-> 4-> 6-> 7 (29)
7-> 2-> 5-> 8-> 3-> 1-> 6-> 4-> 7 (29)
7-> 2-> 5-> 8-> 3-> 4-> 1-> 6-> 7 (25)
7-> 2-> 5-> 8-> 3-> 4-> 6-> 1-> 7 (25)
7-> 2-> 5-> 8-> 3-> 6-> 1-> 4-> 7 (22)
7-> 2-> 5-> 8-> 3-> 6-> 4-> 1-> 7 (22)
7-> 2-> 5-> 8-> 4-> 1-> 3-> 6-> 7 (29)
7-> 2-> 5-> 8-> 4-> 1-> 6-> 3-> 7 (25)
7-> 2-> 5-> 8-> 4-> 3-> 1-> 6-> 7 (32)
7-> 2-> 5-> 8-> 4-> 3-> 6-> 1-> 7 (32)
7-> 2-> 5-> 8-> 4-> 6-> 1-> 3-> 7 (32)
7-> 2-> 5-> 8-> 4-> 6-> 3-> 1-> 7 (32)
7-> 2-> 5-> 8-> 6-> 1-> 3-> 4-> 7 (28)
7-> 2-> 5-> 8-> 6-> 1-> 4-> 3-> 7 (24)
7-> 2-> 5-> 8-> 6-> 3-> 1-> 4-> 7 (25)
7-> 2-> 5-> 8-> 6-> 3-> 4-> 1-> 7 (25)
7-> 2-> 5-> 8-> 6-> 4-> 1-> 3-> 7 (28)
7-> 2-> 5-> 8-> 6-> 4-> 3-> 1-> 7 (28)
7-> 2-> 6-> 1-> 3-> 4-> 5-> 8-> 7 (29)
7-> 2-> 6-> 1-> 3-> 4-> 8-> 5-> 7 (33)
7-> 2-> 6-> 1-> 3-> 5-> 4-> 8-> 7 (28)
7-> 2-> 6-> 1-> 3-> 5-> 8-> 4-> 7 (28)
7-> 2-> 6-> 1-> 3-> 8-> 4-> 5-> 7 (29)
7-> 2-> 6-> 1-> 3-> 8-> 5-> 4-> 7 (25)
7-> 2-> 6-> 1-> 4-> 3-> 5-> 8-> 7 (25)
7-> 2-> 6-> 1-> 4-> 3-> 8-> 5-> 7 (26)
7-> 2-> 6-> 1-> 4-> 5-> 3-> 8-> 7 (21)
7-> 2-> 6-> 1-> 4-> 5-> 8-> 3-> 7 (21)
7-> 2-> 6-> 1-> 4-> 8-> 3-> 5-> 7 (25)
7-> 2-> 6-> 1-> 4-> 8-> 5-> 3-> 7 (24)
7-> 2-> 6-> 1-> 5-> 3-> 4-> 8-> 7 (27)
```

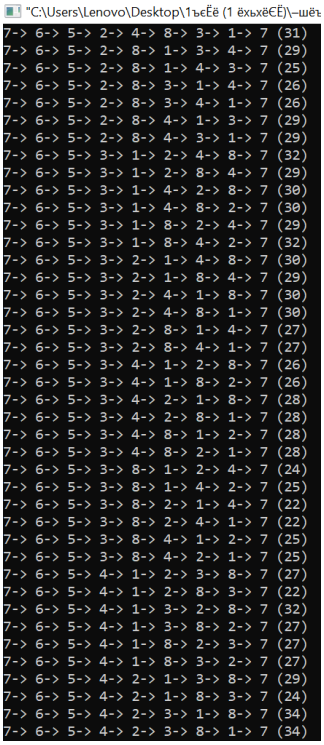
```
"C:\Users\Lenovo\Desktop\1ъєЁё (1 ёхьхёЄЁ)\–шёъ
7-> 3-> 2-> 4-> 8-> 6-> 1-> 5-> 7 (32)
7-> 3-> 2-> 4-> 8-> 6-> 5-> 1-> 7 (32)
7-> 3-> 2-> 5-> 1-> 4-> 6-> 8-> 7 (28)
7-> 3-> 2-> 5-> 1-> 4-> 8-> 6-> 7 (27)
7-> 3-> 2-> 5-> 1-> 6-> 4-> 8-> 7 (32)
7-> 3-> 2-> 5-> 1-> 6-> 8-> 4-> 7 (27)
7-> 3-> 2-> 5-> 1-> 8-> 4-> 6-> 7 (31)
7-> 3-> 2-> 5-> 1-> 8-> 6-> 4-> 7 (27)
7-> 3-> 2-> 5-> 4-> 1-> 6-> 8-> 7 (25)
7-> 3-> 2-> 5-> 4-> 1-> 8-> 6-> 7 (24)
7-> 3-> 2-> 5-> 4-> 6-> 1-> 8-> 7 (29)
7-> 3-> 2-> 5-> 4-> 6-> 8-> 1-> 7 (29)
7-> 3-> 2-> 5-> 4-> 8-> 1-> 6-> 7 (28)
7-> 3-> 2-> 5-> 4-> 8-> 6-> 1-> 7 (28)
7-> 3-> 2-> 5-> 6-> 1-> 4-> 8-> 7 (30)
7-> 3-> 2-> 5-> 6-> 1-> 8-> 4-> 7 (29)
7-> 3-> 2-> 5-> 6-> 4-> 1-> 8-> 7 (30)
7-> 3-> 2-> 5-> 6-> 4-> 8-> 1-> 7 (30)
7-> 3-> 2-> 5-> 6-> 8-> 1-> 4-> 7 (25)
7-> 3-> 2-> 5-> 6-> 8-> 4-> 1-> 7 (25)
7-> 3-> 2-> 5-> 8-> 1-> 4-> 6-> 7 (29)
7-> 3-> 2-> 5-> 8-> 1-> 6-> 4-> 7 (29)
7-> 3-> 2-> 5-> 8-> 4-> 1-> 6-> 7 (29)
7-> 3-> 2-> 5-> 8-> 4-> 6-> 1-> 7 (29)
7-> 3-> 2-> 5-> 8-> 6-> 1-> 4-> 7 (25)
7-> 3-> 2-> 5-> 8-> 6-> 4-> 1-> 7 (25)
7-> 3-> 2-> 6-> 1-> 4-> 5-> 8-> 7 (26)
7-> 3-> 2-> 6-> 1-> 4-> 8-> 5-> 7 (30)
7-> 3-> 2-> 6-> 1-> 5-> 4-> 8-> 7 (28)
7-> 3-> 2-> 6-> 1-> 5-> 8-> 4-> 7 (28)
7-> 3-> 2-> 6-> 1-> 8-> 4-> 5-> 7 (29)
7-> 3-> 2-> 6-> 1-> 8-> 5-> 4-> 7 (25)
7-> 3-> 2-> 6-> 4-> 1-> 5-> 8-> 7 (29)
7-> 3-> 2-> 6-> 4-> 1-> 8-> 5-> 7 (30)
7-> 3-> 2-> 6-> 4-> 5-> 1-> 8-> 7 (28)
7-> 3-> 2-> 6-> 4-> 5-> 8-> 1-> 7 (28)
7-> 3-> 2-> 6-> 4-> 8-> 1-> 5-> 7 (32)
7-> 3-> 2-> 6-> 4-> 8-> 5-> 1-> 7 (32)
7-> 3-> 2-> 6-> 5-> 1-> 4-> 8-> 7 (29)
7-> 3-> 2-> 6-> 5-> 1-> 8-> 4-> 7 (28)
7-> 3-> 2-> 6-> 5-> 4-> 1-> 8-> 7 (26)
```

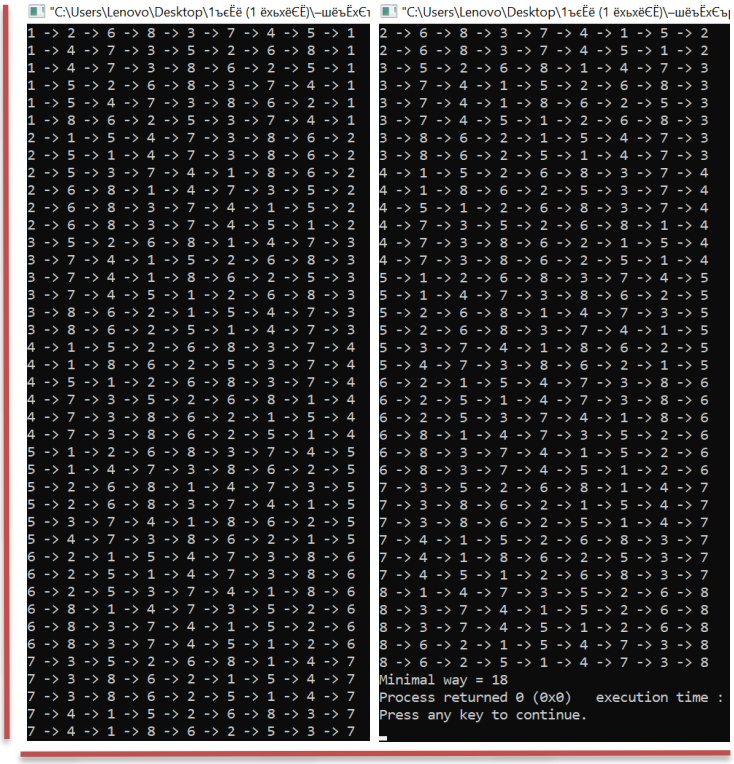
```
"C:\Users\Lenovo\Desktop\1ъєЁё (1 ёхьхёЄЁ)\–шёъЁ
7-> 3-> 4-> 8-> 1-> 6-> 5-> 2-> 7 (28)
7-> 3-> 4-> 8-> 2-> 1-> 5-> 6-> 7 (28)
7-> 3-> 4-> 8-> 2-> 1-> 6-> 5-> 7 (30)
7-> 3-> 4-> 8-> 2-> 5-> 1-> 6-> 7 (27)
7-> 3-> 4-> 8-> 2-> 5-> 6-> 1-> 7 (27)
7-> 3-> 4-> 8-> 2-> 6-> 1-> 5-> 7 (28)
7-> 3-> 4-> 8-> 2-> 6-> 5-> 1-> 7 (28)
7-> 3-> 4-> 8-> 5-> 1-> 2-> 6-> 7 (26)
7-> 3-> 4-> 8-> 5-> 1-> 6-> 2-> 7 (27)
7-> 3-> 4-> 8-> 5-> 2-> 1-> 6-> 7 (27)
7-> 3-> 4-> 8-> 5-> 2-> 6-> 1-> 7 (27)
7-> 3-> 4-> 8-> 5-> 6-> 1-> 2-> 7 (29)
7-> 3-> 4-> 8-> 5-> 6-> 2-> 1-> 7 (29)
7-> 3-> 4-> 8-> 6-> 1-> 2-> 5-> 7 (27)
7-> 3-> 4-> 8-> 6-> 1-> 5-> 2-> 7 (26)
7-> 3-> 4-> 8-> 6-> 2-> 1-> 5-> 7 (26)
7-> 3-> 4-> 8-> 6-> 2-> 5-> 1-> 7 (26)
7-> 3-> 4-> 8-> 6-> 5-> 1-> 2-> 7 (27)
7-> 3-> 4-> 8-> 6-> 5-> 2-> 1-> 7 (27)
7-> 3-> 5-> 1-> 2-> 4-> 6-> 8-> 7 (26)
7-> 3-> 5-> 1-> 2-> 4-> 8-> 6-> 7 (25)
7-> 3-> 5-> 1-> 2-> 6-> 4-> 8-> 7 (26)
7-> 3-> 5-> 1-> 2-> 6-> 8-> 4-> 7 (21)
7-> 3-> 5-> 1-> 2-> 8-> 4-> 6-> 7 (27)
7-> 3-> 5-> 1-> 2-> 8-> 6-> 4-> 7 (23)
7-> 3-> 5-> 1-> 4-> 2-> 6-> 8-> 7 (22)
7-> 3-> 5-> 1-> 4-> 2-> 8-> 6-> 7 (23)
7-> 3-> 5-> 1-> 4-> 6-> 2-> 8-> 7 (24)
7-> 3-> 5-> 1-> 4-> 6-> 8-> 2-> 7 (24)
7-> 3-> 5-> 1-> 4-> 8-> 2-> 6-> 7 (23)
7-> 3-> 5-> 1-> 4-> 8-> 6-> 2-> 7 (22)
7-> 3-> 5-> 1-> 6-> 2-> 4-> 8-> 7 (26)
7-> 3-> 5-> 1-> 6-> 2-> 8-> 4-> 7 (23)
7-> 3-> 5-> 1-> 6-> 4-> 2-> 8-> 7 (28)
7-> 3-> 5-> 1-> 6-> 4-> 8-> 2-> 7 (28)
7-> 3-> 5-> 1-> 6-> 8-> 2-> 4-> 7 (23)
7-> 3-> 5-> 1-> 6-> 8-> 4-> 2-> 7 (26)
7-> 3-> 5-> 1-> 8-> 2-> 4-> 6-> 7 (27)
7-> 3-> 5-> 1-> 8-> 2-> 6-> 4-> 7 (23)
7-> 3-> 5-> 1-> 8-> 4-> 2-> 6-> 7 (25)
7-> 3-> 5-> 1-> 8-> 4-> 6-> 2-> 7 (26)
```

```
"C:\Users\Lenovo\Desktop\1ъєЁё (1 ёхьхёЄЁ)\–шёъЁ
7-> 3-> 6-> 2-> 5-> 4-> 8-> 1-> 7 (20)
7-> 3-> 6-> 2-> 5-> 8-> 1-> 4-> 7 (20)
7-> 3-> 6-> 2-> 5-> 8-> 4-> 1-> 7 (20)
7-> 3-> 6-> 2-> 8-> 1-> 4-> 5-> 7 (22)
7-> 3-> 6-> 2-> 8-> 1-> 5-> 4-> 7 (20)
7-> 3-> 6-> 2-> 8-> 4-> 1-> 5-> 7 (25)
7-> 3-> 6-> 2-> 8-> 4-> 5-> 1-> 7 (25)
7-> 3-> 6-> 2-> 8-> 5-> 1-> 4-> 7 (21)
7-> 3-> 6-> 2-> 8-> 5-> 4-> 1-> 7 (21)
7-> 3-> 6-> 4-> 1-> 2-> 5-> 8-> 7 (25)
7-> 3-> 6-> 4-> 1-> 2-> 8-> 5-> 7 (27)
7-> 3-> 6-> 4-> 1-> 5-> 2-> 8-> 7 (25)
7-> 3-> 6-> 4-> 1-> 5-> 8-> 2-> 7 (26)
7-> 3-> 6-> 4-> 1-> 8-> 2-> 5-> 7 (26)
7-> 3-> 6-> 4-> 1-> 8-> 5-> 2-> 7 (25)
7-> 3-> 6-> 4-> 2-> 1-> 5-> 8-> 7 (28)
7-> 3-> 6-> 4-> 2-> 1-> 8-> 5-> 7 (29)
7-> 3-> 6-> 4-> 2-> 5-> 1-> 8-> 7 (27)
7-> 3-> 6-> 4-> 2-> 5-> 8-> 1-> 7 (27)
7-> 3-> 6-> 4-> 2-> 8-> 1-> 5-> 7 (29)
7-> 3-> 6-> 4-> 2-> 8-> 5-> 1-> 7 (29)
7-> 3-> 6-> 4-> 5-> 1-> 2-> 8-> 7 (25)
7-> 3-> 6-> 4-> 5-> 1-> 8-> 2-> 7 (25)
7-> 3-> 6-> 4-> 5-> 2-> 1-> 8-> 7 (24)
7-> 3-> 6-> 4-> 5-> 2-> 8-> 1-> 7 (24)
7-> 3-> 6-> 4-> 5-> 8-> 1-> 2-> 7 (25)
7-> 3-> 6-> 4-> 5-> 8-> 2-> 1-> 7 (25)
7-> 3-> 6-> 4-> 8-> 1-> 2-> 5-> 7 (28)
7-> 3-> 6-> 4-> 8-> 1-> 5-> 2-> 7 (27)
7-> 3-> 6-> 4-> 8-> 2-> 1-> 5-> 7 (29)
7-> 3-> 6-> 4-> 8-> 2-> 5-> 1-> 7 (29)
7-> 3-> 6-> 4-> 8-> 5-> 1-> 2-> 7 (28)
7-> 3-> 6-> 4-> 8-> 5-> 2-> 1-> 7 (28)
7-> 3-> 6-> 5-> 1-> 2-> 4-> 8-> 7 (28)
7-> 3-> 6-> 5-> 1-> 2-> 8-> 4-> 7 (25)
7-> 3-> 6-> 5-> 1-> 4-> 2-> 8-> 7 (26)
7-> 3-> 6-> 5-> 1-> 4-> 8-> 2-> 7 (26)
7-> 3-> 6-> 5-> 1-> 8-> 2-> 4-> 7 (25)
7-> 3-> 6-> 5-> 1-> 8-> 4-> 2-> 7 (28)
7-> 3-> 6-> 5-> 2-> 1-> 4-> 8-> 7 (25)
7-> 3-> 6-> 5-> 2-> 1-> 8-> 4-> 7 (24)
```

```
"C:\Users\Lenovo\Desktop\1ъєЁё (1 ёхьхёЄЁ)\–шёъ
7-> 3-> 6-> 8-> 2-> 4-> 1-> 5-> 7 (25)
7-> 3-> 6-> 8-> 2-> 4-> 5-> 1-> 7 (25)
7-> 3-> 6-> 8-> 2-> 5-> 1-> 4-> 7 (20)
7-> 3-> 6-> 8-> 2-> 5-> 4-> 1-> 7 (20)
7-> 3-> 6-> 8-> 4-> 1-> 2-> 5-> 7 (24)
7-> 3-> 6-> 8-> 4-> 1-> 5-> 2-> 7 (23)
7-> 3-> 6-> 8-> 4-> 2-> 1-> 5-> 7 (27)
7-> 3-> 6-> 8-> 4-> 2-> 5-> 1-> 7 (27)
7-> 3-> 6-> 8-> 4-> 5-> 1-> 2-> 7 (23)
7-> 3-> 6-> 8-> 4-> 5-> 2-> 1-> 7 (23)
7-> 3-> 6-> 8-> 5-> 1-> 2-> 4-> 7 (23)
7-> 3-> 6-> 8-> 5-> 1-> 4-> 2-> 7 (24)
7-> 3-> 6-> 8-> 5-> 2-> 1-> 4-> 7 (20)
7-> 3-> 6-> 8-> 5-> 2-> 4-> 1-> 7 (20)
7-> 3-> 6-> 8-> 5-> 4-> 1-> 2-> 7 (21)
7-> 3-> 6-> 8-> 5-> 4-> 2-> 1-> 7 (21)
7-> 3-> 8-> 1-> 2-> 4-> 5-> 6-> 7 (24)
7-> 3-> 8-> 1-> 2-> 4-> 6-> 5-> 7 (29)
7-> 3-> 8-> 1-> 2-> 5-> 4-> 6-> 7 (23)
7-> 3-> 8-> 1-> 2-> 5-> 6-> 4-> 7 (24)
7-> 3-> 8-> 1-> 2-> 6-> 4-> 5-> 7 (24)
7-> 3-> 8-> 1-> 2-> 6-> 5-> 4-> 7 (20)
7-> 3-> 8-> 1-> 4-> 2-> 5-> 6-> 7 (24)
7-> 3-> 8-> 1-> 4-> 2-> 6-> 5-> 7 (25)
7-> 3-> 8-> 1-> 4-> 5-> 2-> 6-> 7 (19)
7-> 3-> 8-> 1-> 4-> 5-> 6-> 2-> 7 (21)
7-> 3-> 8-> 1-> 4-> 6-> 2-> 5-> 7 (24)
7-> 3-> 8-> 1-> 4-> 6-> 5-> 2-> 7 (25)
7-> 3-> 8-> 1-> 5-> 2-> 4-> 6-> 7 (26)
7-> 3-> 8-> 1-> 5-> 2-> 6-> 4-> 7 (22)
7-> 3-> 8-> 1-> 5-> 4-> 2-> 6-> 7 (22)
7-> 3-> 8-> 1-> 5-> 4-> 6-> 2-> 7 (23)
7-> 3-> 8-> 1-> 5-> 6-> 2-> 4-> 7 (23)
7-> 3-> 8-> 1-> 5-> 6-> 4-> 2-> 7 (28)
7-> 3-> 8-> 1-> 6-> 2-> 4-> 5-> 7 (24)
7-> 3-> 8-> 1-> 6-> 2-> 5-> 4-> 7 (19)
7-> 3-> 8-> 1-> 6-> 4-> 2-> 5-> 7 (28)
7-> 3-> 8-> 1-> 6-> 4-> 5-> 2-> 7 (24)
7-> 3-> 8-> 1-> 6-> 5-> 2-> 4-> 7 (25)
7-> 3-> 8-> 2-> 1-> 4-> 5-> 6-> 7 (22)
```

Мінімальні шляхи з усіх вершин





Код програми :

```
#include <iostream>
#include <stdio.h>
#include <string>
#include <fstream>

using namespace std;
ifstream fin;
string path = "Matrix.txt";

struct massiveays{
  int massiveayl[9];
};

int** vvid() {
  int counter = 8;

string str;
str = "";

fin.open(path);
int **massive;
massive = new int*[counter];
for (int i = 0; i < counter; i++)
 massive[i] = new int[counter];

for (int i = 0; i < counter; i++){
 for (int j = 0; j < counter; j++)
  massive[i][j] = 0;
}
for (int i = 0; i < counter; i++){
 for (int j = i + 1; j < counter; j++){
  getline(fin, str);
  massive[i][j] = atoi(str.c_str());
  massive[j][i] = atoi(str.c_str());
 }
}
fin.close();

return massive;
}
```

```
43  bool comp(int* massive, int counter){
44   int* massiveayl = new int[counter];
45   for (int i =0; i < counter; i++){
46    massiveayl[i] = counter -i;
47   }
48
49   for (int i = 0; i < counter; i++){
50    if (massiveayl[i] != massive[i]){
51     return true;
52    }
53    else
54     continue;
55   }
56   return false;
57  }
58
59
60  bool repeat(int* massiveayl, int size){
61   bool k = true;
62
63   for (int i = 0; i < size; i++){
64    for (int j = 0; j < size; j++){
65     if (massiveayl[i] == massiveayl[j] && i != j)return false;
66    }
67   }
68   return true;
69  }
```

```
70
71
72  int way(int** mat, int* massive){
73   int counter = 0;
74   for (int i = 0; i < 7; i++){
75    counter += mat[massive[i] - 1][massive[i + 1] - 1];
76   }
77   counter += mat[massive[7] - 1][massive[0] - 1];
78   return counter;
79  }
80
81  int main() {
82   int const counter = 8;
83   int **massive;
84   massive = vvid();
85   int var = counter - 1;
86   bool k = true;
87   int *massiveayl = new int[counter];
88
89   int* minmas = new int[9];
90   int min = 1000;
91   int leng = 0;
92
93   int m = 0;
94
95   for (int i = 0; i < counter; i++){
96    massiveayl[i] = 1;
97    minmas[i] = 1;
98   }
99
100  while (comp(massiveayl, counter)){
101   while (massiveayl[var] != counter){
102    massiveayl[var]++;
103
104    if (repeat(massiveayl, counter)){
105     leng = way(massive, massiveayl);
106
107     for (int i = 0; i < counter; i++){
108      cout << massiveayl[i] << "-> ";}
109     cout << massiveayl[0] << " (" << leng << ") ";
110     cout << endl;
111
112     if (leng < min){
113      min = leng;
114      m = 1;}
115     if (leng == min){
116      m++;}
117    }
118   }
119
120   while (massiveayl[var] == counter){
121    massiveayl[var] = 1;
122    var--;}
123   massiveayl[var]++;
```

```
124
125     if (repeat(massiveayl, counter)){
126      for (int i = 0; i < counter; i++){
127       cout << massiveayl[i] << "-> ";}
128      cout << massiveayl[0] << " (" << leng << ") ";
129      cout << endl;
130
131      leng = way(massive, massiveayl);
132
133     if (leng < min){
134      min = leng;
135      m = 1;}
136     if (leng == min){
137      m++;}
138    }
139    var = counter - 1;
140   }
141
142   for (int i = 0; i < counter; i++){
143    massiveayl[i] = 1;
144    minmas[i] = 1;
145   }
146   massiveays *rez = new massiveays[m];
147   int iteration= 0;
148
149   while (comp(massiveayl, counter)){
150    while (massiveayl[var] != counter){
151     massiveayl[var]++;
152
153     if (repeat(massiveayl, counter)){
154      leng = way(massive, massiveayl);
155
156      if (leng == min){
157       for (int i = 0; i < counter; i++){
158        rez[iteration].massiveayl[i] = massiveayl[i];
159       }
160       rez[iteration].massiveayl[counter] = massiveayl[0];
161       iteration++;
162      }
163     }
164    }
165    while (massiveayl[var] == counter){
166     massiveayl[var] = 1;
167     var--;}
168    massiveayl[var]++;
169
170    if (repeat(massiveayl, counter)){
171     leng = way(massive, massiveayl);
172
173     if (leng == min){
174      for (int i = 0; i < counter; i++){
175       rez[iteration].massiveayl[i] = massiveayl[i];
176      }
177      rez[iteration].massiveayl[counter] = massiveayl[0];
177      rez[iteration].massiveayl[counter] = massiveayl[0];
178      iteration++;
179     }
180
181    }
182    var = counter - 1;
183   }
184
185   cout << "Ways: " << endl;
186
187   for (int i = 0; i < iteration - 1; i++){
188    for (int j = 0; j <= counter; j++){
189     if (j != 0){
190      cout << "-> ";}
191     cout << rez[i].massiveayl[j] << " ";}
192    cout << endl;}
193   cout << "Minimal way = " << min;
194
195   return 0;
196  }
197
```

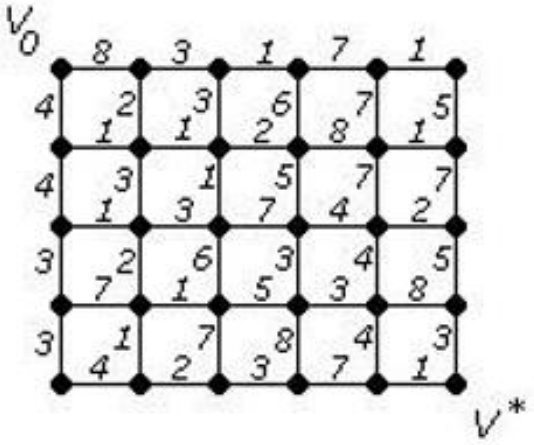
Записуємо у файл верхній трикутник матриці без головної діагоналі :

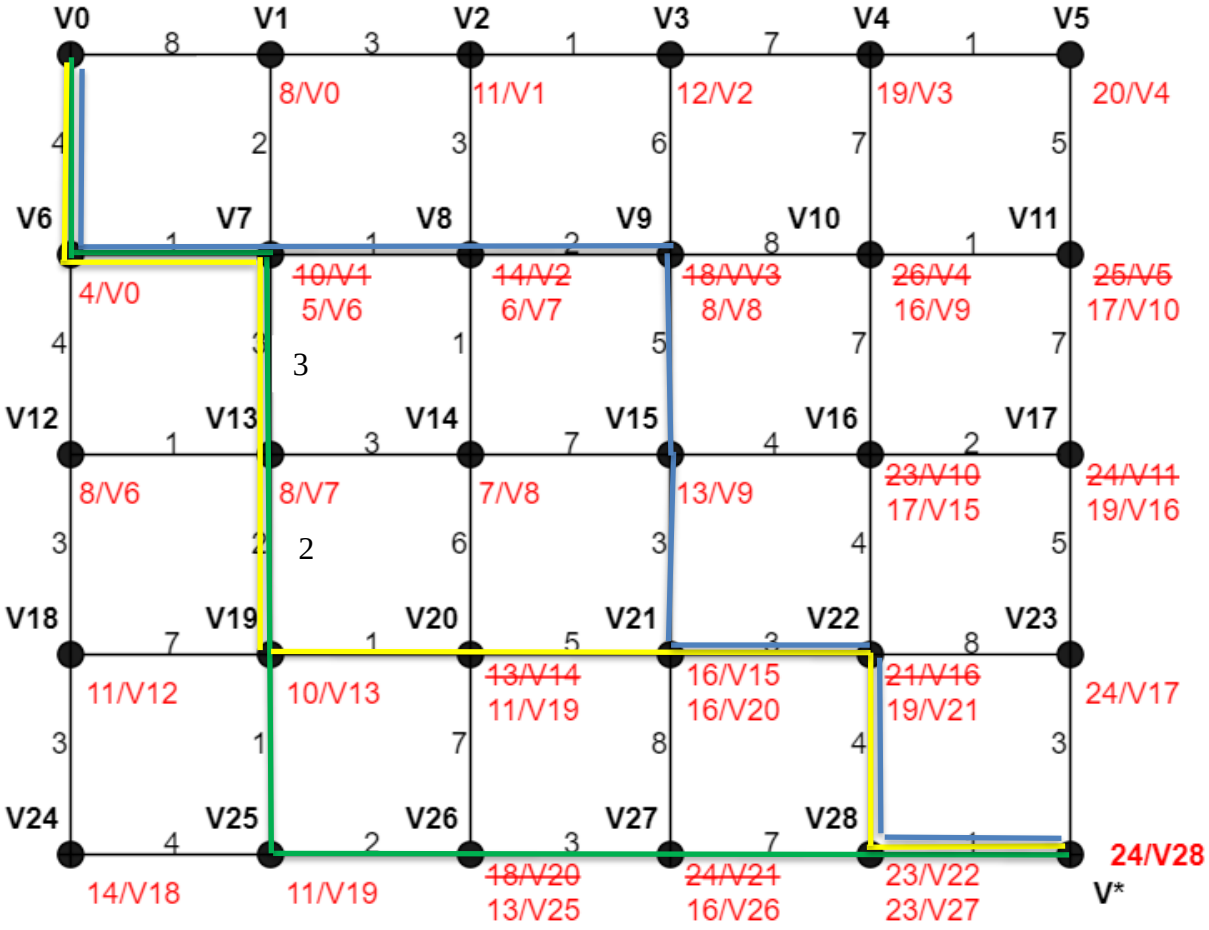
Matrix: Блокнот

Файл Редагування Формат Вигляд Довідка

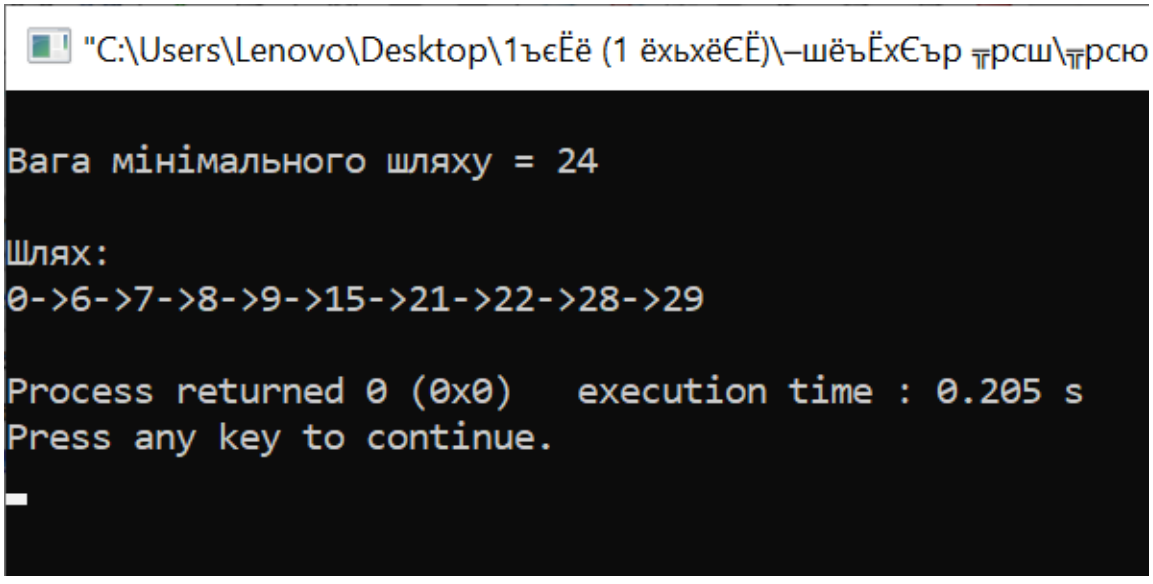
1
5
1
5
1
6
1
7
5
6
1
2
3
5
6
2
1
2
6
5
1
5
7
7
7
1
1
2

Завдання № 7: За допомогою алгоритму Дейкстри знайти найкоротший шлях у графі між парою вершин V0 і V*





Отже довжина мінімального шляху : 24



Код програми:

```
#include <bits/stdc++.h>
using namespace std;

    int main()
    {
        setlocale(LC_ALL,"Ukrainian");
        int n,i,j;
        n=30;//кількість вершин
        int matrix[n][n];
        for (i=0;i<n;i++)
        {
            for (j=0;j<n;j++){
                matrix[i][j]=0;
            }
        }

        matrix[0][1]=matrix[1][0]=8;
        matrix[0][6]=matrix[6][0]=4;
        matrix[1][2]=matrix[2][1]=3;
        matrix[1][7]=matrix[7][1]=2;
        matrix[2][3]=matrix[3][2]=1;
        matrix[2][8]=matrix[8][2]=3;
        matrix[3][4]=matrix[4][3]=7;
        matrix[3][9]=matrix[9][3]=6;
        matrix[4][5]=matrix[5][4]=1;
        matrix[4][10]=matrix[10][4]=7;
        matrix[5][11]=matrix[11][5]=5;
        matrix[6][7]=matrix[7][6]=1;
        matrix[6][12]=matrix[12][6]=4;
        matrix[7][8]=matrix[8][7]=1;
        matrix[7][13]=matrix[13][7]=3;
        matrix[8][9]=matrix[9][8]=2;
        matrix[8][14]=matrix[14][8]=1;
        matrix[9][10]=matrix[10][9]=8;
        matrix[9][15]=matrix[15][9]=5;
        matrix[10][11]=matrix[11][10]=1;
        matrix[10][16]=matrix[16][10]=7;
        matrix[11][17]=matrix[17][11]=7;
        matrix[12][13]=matrix[13][12]=1;
        matrix[12][18]=matrix[18][12]=3;
        matrix[13][14]=matrix[14][13]=3;
        matrix[13][19]=matrix[19][13]=2;
        matrix[14][15]=matrix[15][14]=7;
        matrix[14][20]=matrix[20][14]=6;
        matrix[15][16]=matrix[16][15]=4;
        matrix[15][21]=matrix[21][15]=3;
        matrix[16][17]=matrix[17][16]=2;
        matrix[16][22]=matrix[22][16]=4;
        matrix[17][23]=matrix[23][17]=5;
        matrix[18][19]=matrix[19][18]=7;
        matrix[18][24]=matrix[24][18]=3;
        matrix[19][20]=matrix[20][19]=1;
        matrix[19][25]=matrix[25][19]=1;
```

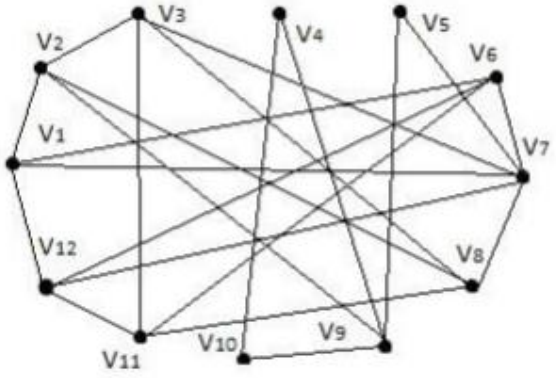
```
        matrix[20][21]=matrix[21][20]=5;
        matrix[20][26]=matrix[26][20]=7;
        matrix[21][22]=matrix[22][21]=3;
        matrix[21][27]=matrix[27][21]=8;
        matrix[22][28]=matrix[28][22]=4;
        matrix[22][23]=matrix[23][22]=8;
        matrix[23][29]=matrix[29][23]=3;
        matrix[24][25]=matrix[25][24]=4;
        matrix[25][26]=matrix[26][25]=2;
        matrix[26][27]=matrix[27][26]=3;
        matrix[27][28]=matrix[28][27]=7;
        matrix[28][29]=matrix[29][28]=1;
```

```
        int numb[n]{-1};
        int rebra=0;
        for (i=0;i<n;i++)
        {
            for (j=0;j<n;j++)
            {
                if (matrix[i][j]!=0)
                    rebra++;// кількість ребер
            }
        }
        int weight[n]; //ваги
        bool visited[n]; //пройдені
        for (i=0;i<n;i++)
        {
            weight[i]=10000000;
            visited[i]=0;
        }
        weight[0]=0;
        visited[0]=1;
        int nmin,Vmin1,Vmin2;
        while (rebra!=0)
        {
            nmin=10000000;
            for (i=0;i<n;i++)
            {
                if (visited[i]==1)//якщо була
```

```
                if (visited[i]==1)//якщо була пройдена
                {
                    for (j=0;j<n;j++)
                    {
                        if (weight[i]+matrix[i][j]<nmin&&matrix[i][j]!=0)//до рядку і шукає мін
                        {
                            nmin=weight[i]+matrix[i][j];
                            Vmin1=i;
                            Vmin2=j;
                        }
                    }
                }
            }
            if (weight[Vmin2]>nmin)
            {
                weight[Vmin2]=nmin;
                numb[Vmin2]=Vmin1;
            }
            visited[Vmin2]=1;
            matrix[Vmin1][Vmin2]=matrix[Vmin2][Vmin1]=0;
            rebra-=2;//віднімає ребра
        }

        int endd=29;
        int way[n];
        i=0;
        cout<<"\nВага мінімального шляху = "<<weight[endd]<<endl;
```

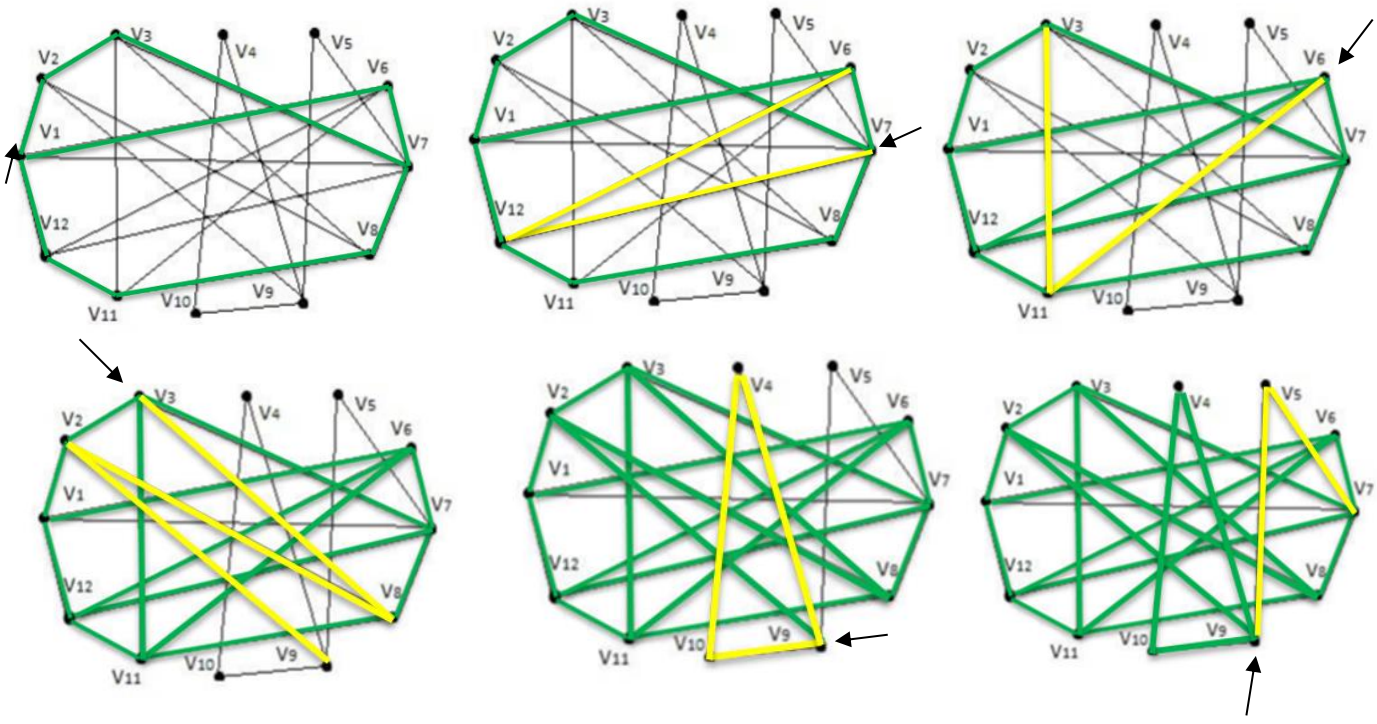
```
119
120         cout<<"\nШлях:"<<endl;
121         while (endd!=0)
122         {
123             way[i]=endd;
124             endd=numb[endd];
125             i++;
126         }
127         way[i]=0;
128
129         for (i;i>=0;i--)
130             {
131                 cout<<way[i];
132                 if (i!=0) cout<<"->";
133             }
134         cout<<endl;
135     }
```

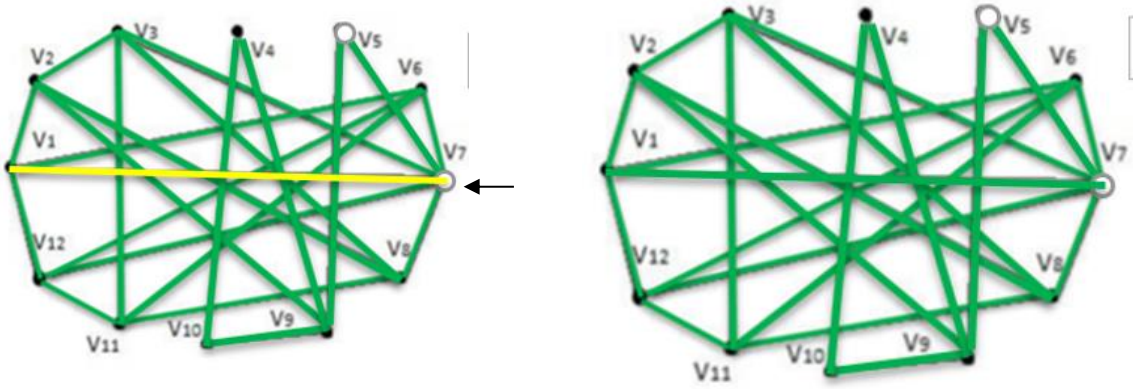
Завдання № 7: Знайти ейлеровий цикл в ейлеровому графі двома методами: а) Флері; б) елементарних циклів.



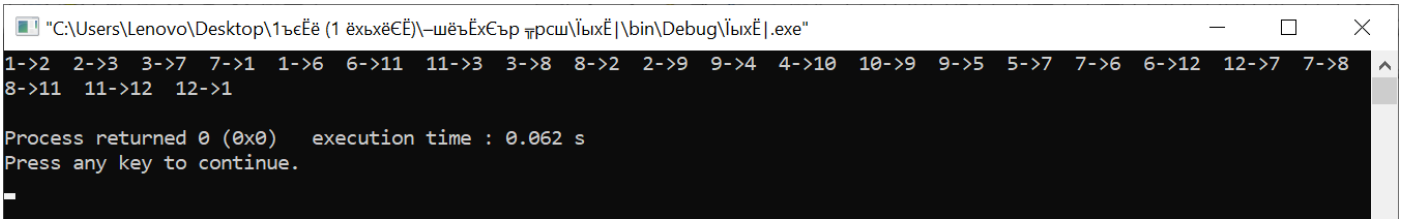
а)Флері

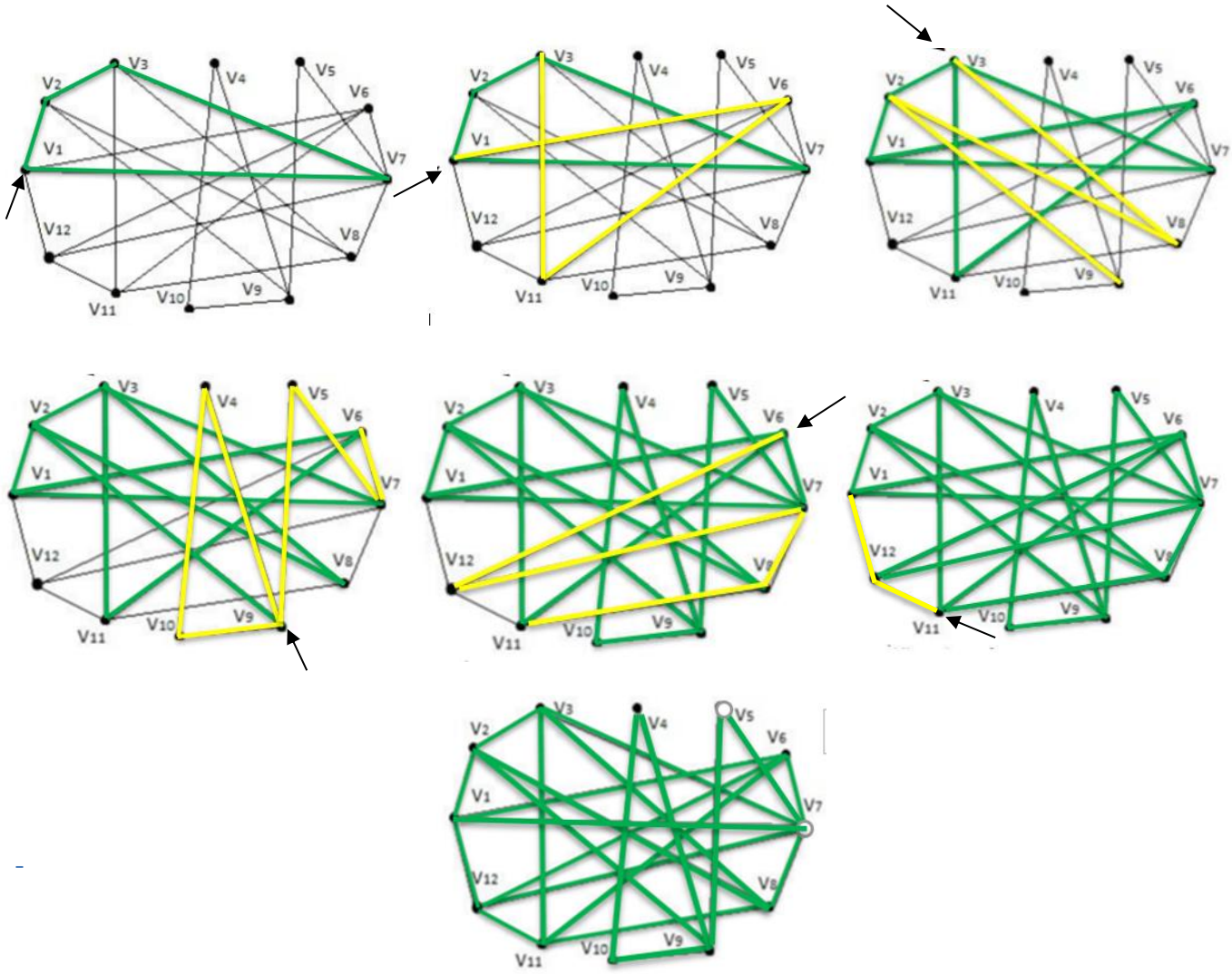
$$V1 \to V2 \to V3 \to V7 \to V8 \to V11 \to V12 \to V1 \to V6 \to V7 \to V12 \to V6 \to V11 \to V3 \to V8 \to V2 \to V9 \to V10 \to V4 \to V9 \to V5 \to V7 \to V1$$





Ще один із варіантів ейлерового циклу в даному графі :



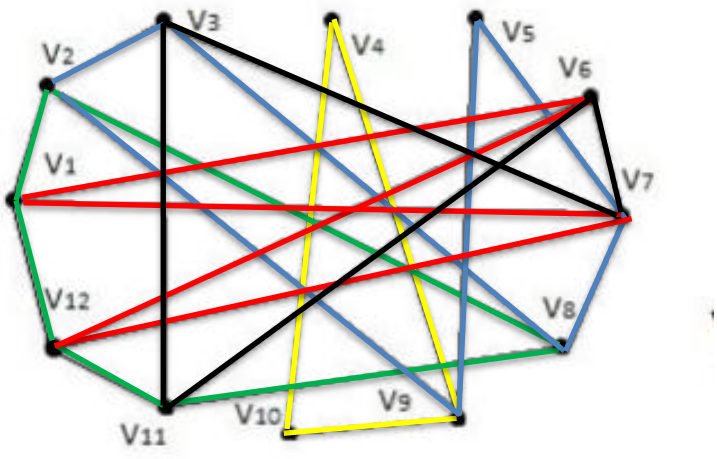


Код програми:

```cpp
#include <iostream>
#include <string.h>
#include <algorithm>
#include <list>
using namespace std;

class Graph
{
  int V;
  list<int>* adj;
public:
  Graph(int V) { this->V = V; adj = new list<int>[V+1]; }
  ~Graph() { delete[] adj; }

  void addEdge(int u, int v) { adj[u].push_back(v); adj[v].push_back(u); }
  void rmvEdge(int u, int v);

  void printEulerTour();
  void printEulerUtil(int s);

  int DFSCount(int v, bool visited[]);

  bool isValidNextEdge(int u, int v);
};

void Graph::printEulerTour()
{
  int u = 1;
  for (int i = 1; i <= V; i++)
    if (adj[i].size() & 1)
    {
      u = i; break;
    }
  printEulerUtil(u);
  cout << endl;
}

void Graph::printEulerUtil(int u)
{
  list<int>::iterator i;
  for (i = adj[u].begin(); i != adj[u].end(); ++i)
  {
    int v = *i;
    if (v != -1 && isValidNextEdge(u, v))
    {
      cout << u << "->" << v << "  ";
      rmvEdge(u, v);
      printEulerUtil(v);
    }
  }
}

bool Graph::isValidNextEdge(int u, int v)
{
  int count = 0;
  list<int>::iterator i;
  for (i = adj[u].begin(); i != adj[u].end(); ++i)
    if (*i != -1)
      count++;
  if (count == 1)
    return true;
  bool visited[20];
  memset(visited, false, V);
  int count1 = DFSCount(u, visited);

  rmvEdge(u, v);
  memset(visited, false, V);
  int count2 = DFSCount(u, visited);

  addEdge(u, v);

  return (count1 > count2) ? false : true;
}

void Graph::rmvEdge(int u, int v)
{
  list<int>::iterator iv = find(adj[u].begin(), adj[u].end(), v);
  *iv = -1;

  list<int>::iterator iu = find(adj[v].begin(), adj[v].end(), u);
  *iu = -1;
```

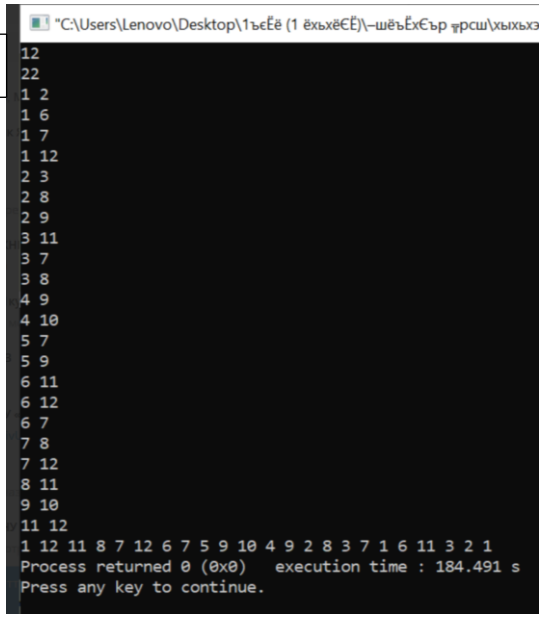
```
}

int Graph::DFSCount(int v, bool visited[])
{
  visited[v] = true;
  int count = 1;

  list<int>::iterator i;
  for (i = adj[v].begin(); i != adj[v].end(); ++i)
    if (*i != -1 && !visited[*i])
      count += DFSCount(*i, visited);

  return count;
}

int main()
{
  Graph g1(20);
  g1.addEdge(1, 2);
  g1.addEdge(1, 6);
  g1.addEdge(1, 7);
  g1.addEdge(1, 12);
  g1.addEdge(2, 3);
  g1.addEdge(2, 8);
  g1.addEdge(2, 9);
  g1.addEdge(3, 7);
  g1.addEdge(3, 8);
  g1.addEdge(3, 11);
  g1.addEdge(4, 9);
  g1.addEdge(4, 10);
  g1.addEdge(5, 7);
  g1.addEdge(5, 9);
  g1.addEdge(6, 11);
  g1.addEdge(6, 12);
  g1.addEdge(6, 7);
  g1.addEdge(7, 12);
  g1.addEdge(7, 8);
  g1.addEdge(8, 11);
  g1.addEdge(9, 10);
  g1.addEdge(11, 12);

  g1.printEulerTour();

  return 0;
}
```

б)елементарні цикли



V1V2V8V11V12V1
V1V2V3V8V7V5V9V2V8V11V12V1
V1V2V3V8V7V1V6V12V7V5V9V2V8V11V12V1
V1V2V3V8V7V1V6V11V3V7V6V12V7V5V9V2V8V11V12V1
V1V2V3V8V7V1V6V11V3V7V6V12V7V5V9V4V10V9V2V8V11V12V1

Кількість вершин
Кількість ребер



Код програми:

```cpp
#include <iostream>
#include <vector>
#include <stack>
#include <algorithm>
#include <list>
using namespace std;

    vector < list<int> > graph;
    vector <int> deg;
    stack<int> head,tail ;


int main()
{
    int n, a, x,y ;
    cin >> n >> a;
    graph.resize(n+1);
    deg.resize(n+1);
    for(;a--;){
        cin >> x >> y;
        graph[x].push_back(y);
        graph[y].push_back(x);
        ++deg[x];
        ++deg[y];
    }

    if(any_of(deg.begin()+1,deg.end(),[](int i){return i&1;})) cout << "-1";
```

```
28      else
29      {
30          head.push(1);
31          while(!head.empty()){
32              while(deg[head.top()]){
33                  int v = graph[head.top()].back();
34                  graph[head.top()].pop_back();
35                  graph[v].remove(head.top());
36                  --deg[head.top()] ;
37                  head.push(v);
38                  --deg[v];
39                  }
40
41              while(!head.empty()&&!deg[head.top()]){
42                  tail.push( head.top());
43                  head.pop();
44                  }
45              }
46
47          while(!tail.empty()){
48              cout << tail.top() <<' ';
49              tail.pop();
50          }
51      }
52  }
```

Завдання № 9: Спростити формули (привести їх до скороченої ДНФ).

$$x\bar{y} \vee x\bar{z} \vee z$$

z	0	0	1	1
x/y	0	1	1	0
0	0	1	1	1
1	0	1	1	1

Отже згідно карти Карно ми бачимо , що ДНФ цієї формули : $x \vee z$