МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”
Кафедра систем штучного інтелекту

Розрахунково-графічна Робота

з дисципліни
«Дискретна математика»

Виконав:
студент групи КН-114
Сиротюк Владислав
Викладач:
Мельникова Н.І.

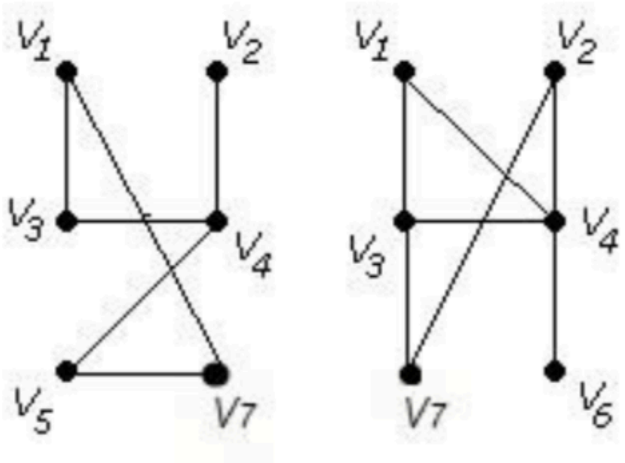
Львів – 2019 р.

ІНДИВІДУАЛЬНІ ЗАВДАННЯ

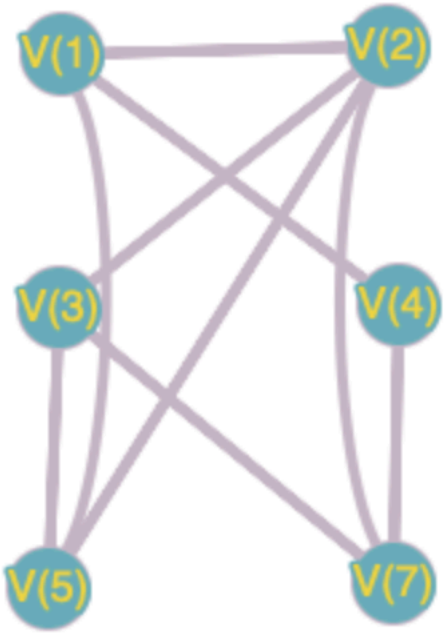
Завдання № 1

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.

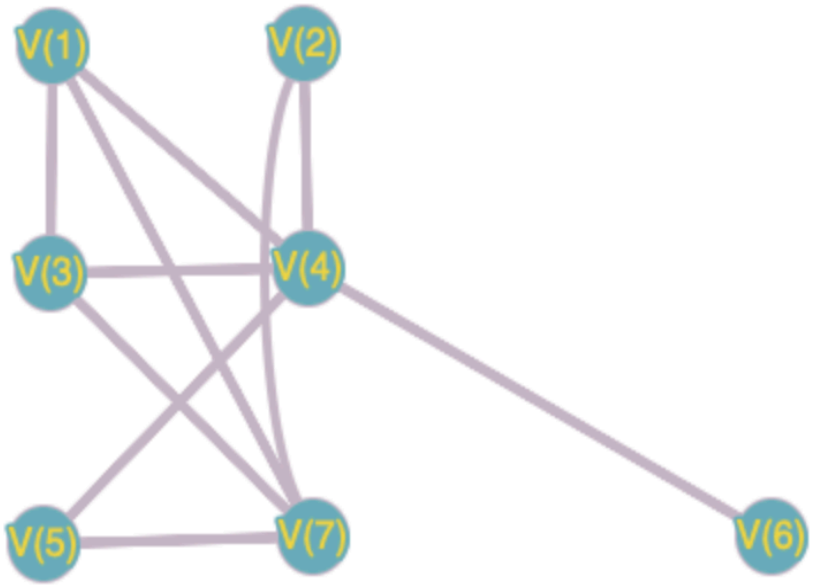
17)



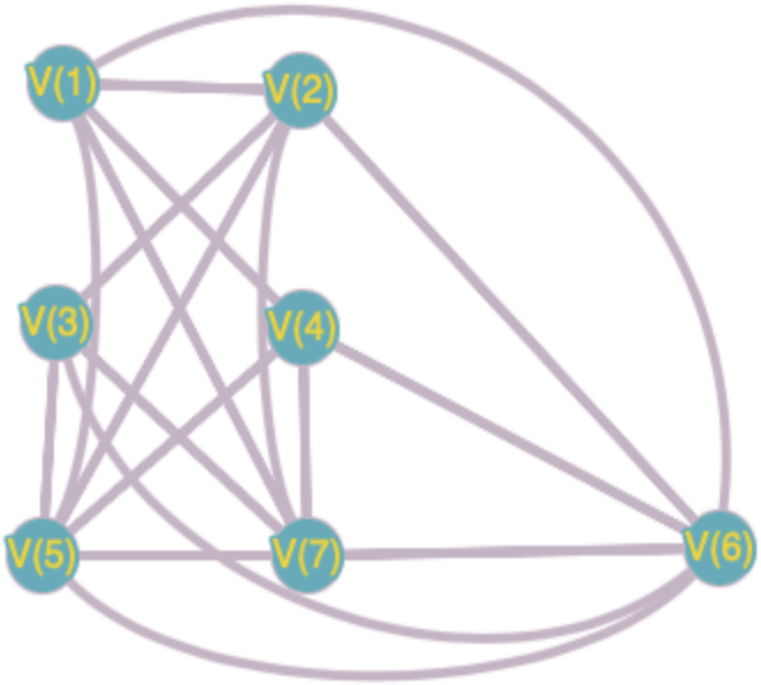
1)



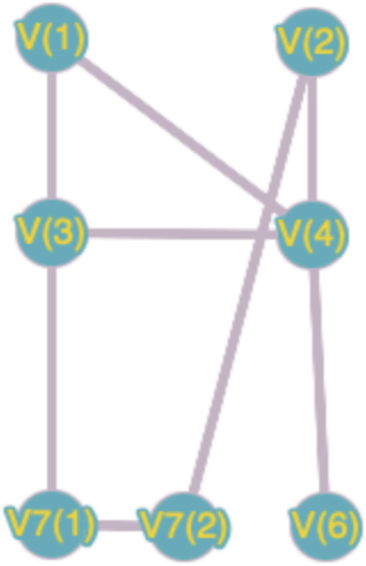
2)



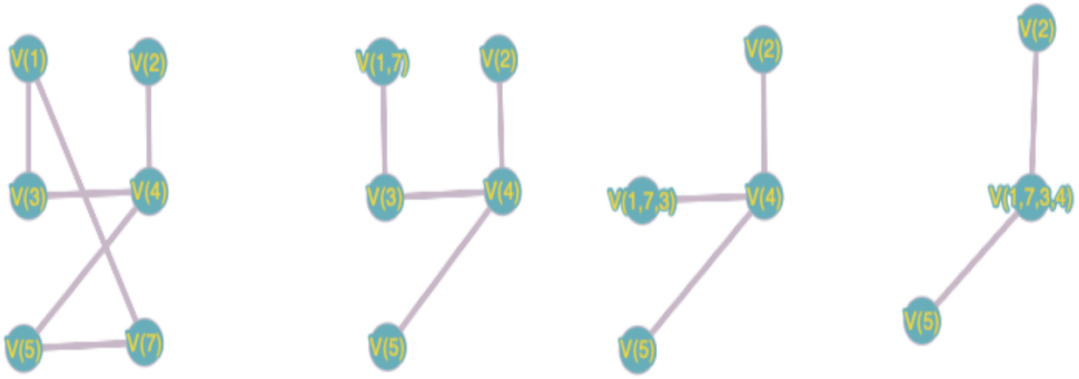
3)



4)



5)



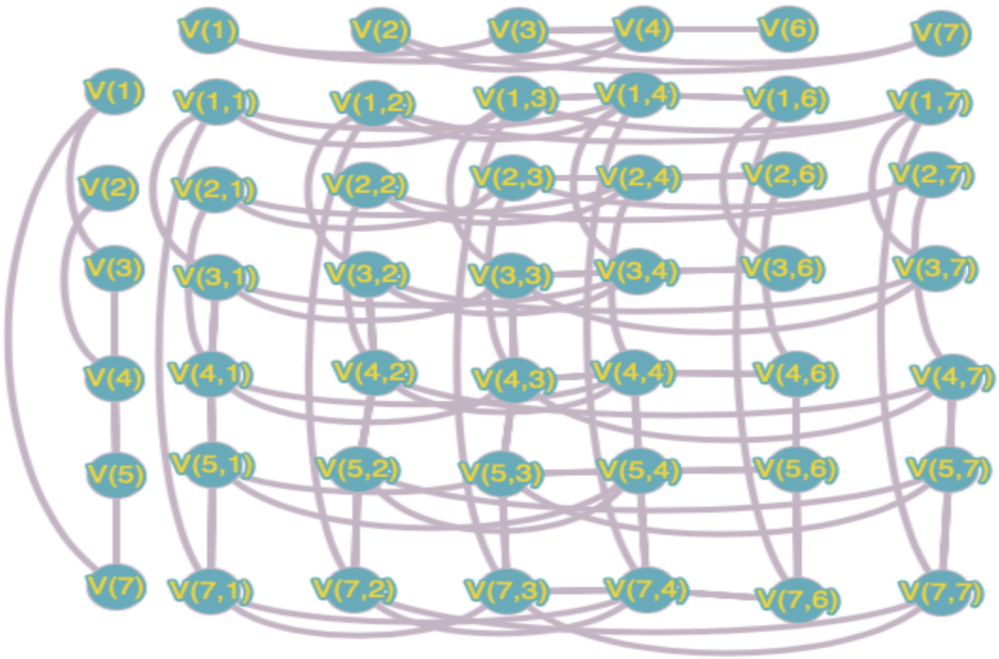
1.Стягуємо V(7)

2.Стягуємо V(1,7)

3.Стягуємо V(1,7,3)

4.Вийшов підграф A,який складається з 3-ох вершин.

6)



G = G1 x G2

V(G) = V1 x V2 = V(G1) x V(G2)

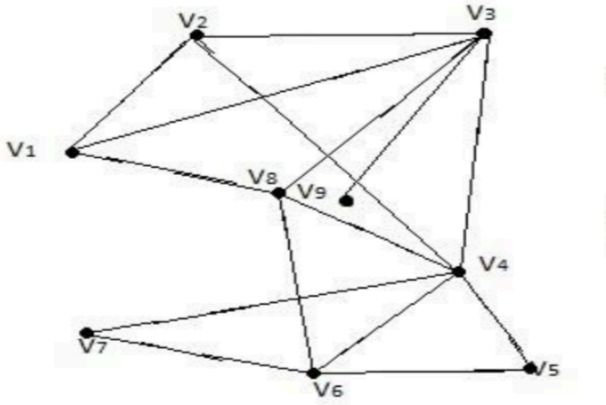
E(G) = E1 x E2 = E(G1 x G2)

V(G1) = {1,2,3,4,6,7}
V(G2) = {1,2,3,4,5,7}
V(G) = { (1;1),(1;2),(1;3),(1;4),(1;6),(1;7);
(2;1),(2;2),(2;3),(2;4),(2;5),(2;7);
(3;1),(3;2),(3;3),(3;4),(3;5),(3;7);
(4;1),(4;2),(4;3),(4;4),(4;5),(4;7);
(6;1),(6;2),(6;3),(6;4),(6;5),(6;7);
(7;1),(7;2),(7;3),(7;4),(7;5),(7;7);}

Завдання № 2

Скласти таблицю суміжності для орграфа.

17)



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

Завдання № 3

Для графа з другого завдання знайти діаметр.

1->2->4->5

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	DFS-номер	Стек
V9	1	(V9)
V3	2	(V9)(V3)
V2	3	(V9)(V3)(V2)
V1	4	(V9)(V3)(V2)(V1)
V8	5	(V9)(V3)(V2)(V1)(V8)
V4	6	(V9)(V3)(V2)(V1)(V8)(V4)
V5	7	(V9)(V3)(V2)(V1)(V8)(V4)(V5)
V6	8	(V9)(V3)(V2)(V1)(V8)(V4)(V5)(V6)
V7	9	(V9)(V3)(V2)(V1)(V8)(V4)(V5)(V6)(V7)
-	-	(V9)(V3)(V2)(V1)(V8)(V4)(V5)(V6)(V7)
-	-	(V9)(V3)(V2)(V1)(V8)(V4)(V5)(V6)
-	-	(V9)(V3)(V2)(V1)(V8)(V4)(V5)
-	-	(V9)(V3)(V2)(V1)(V8)(V4)
-	-	(V9)(V3)(V2)(V1)(V8)
-	-	(V9)(V3)(V2)(V1)
-	-	(V9)(V3)(V2)
-	-	(V9)(V3)
-	-	(V9)
		Ø

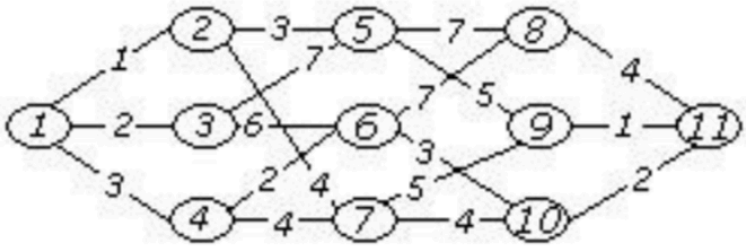
Пошук в глубину:

```
#include <iostream>
using namespace std;
const int n = 5;
int i, j;
bool* visited = new bool[n];
int graph[n][n] =
{
{0, 1, 0, 0, 1},
{1, 0, 1, 1, 0},
{0, 1, 0, 0, 1},
{0, 1, 0, 0, 1},
{1, 0, 1, 1, 0}
};
void DFS(int st)
{
  int r;
  cout << st + 1 << " ";
  visited[st] = true;
  for (r = 0; r <= n; r++)
    if ((graph[st][r] != 0) && (!visited[r]))
      DFS(r);
}
int main()
{

  int start;
  cout << "Матриця суміжності графа: " << endl;
  for (i = 0; i < n; i++)
  {
    visited[i] = false;
    for (j = 0; j < n; j++)
      cout << " " << graph[i][j];
    cout << endl;
  }
  cout << "Стартовая вершина >> "; cin >> start;

  bool* vis = new bool[n];
  cout << "Порядок обходу: ";
  DFS(start - 1);
  delete[]visited;
  return 0;
}
```

Завдання № 5

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

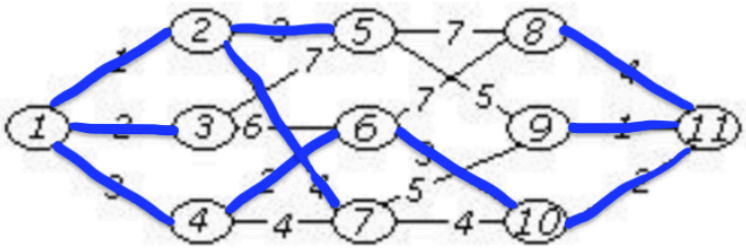
17)



1)За методом Краскала:

2)

17)



1)(1;2),(9;11)
2)(1;3),(4;6),(10;11)
3)(1;4)(2;5)(6;10)
4)(2;7),(8;11)
Вага = 1+1+2+2+2+3+3+3+4+4= 25

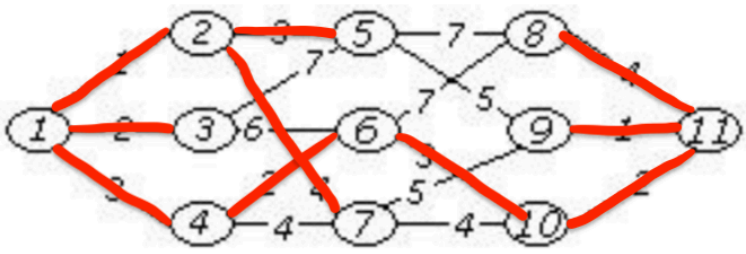
```
#include <iostream>
#include<algorithm>
#include <vector>
using namespace std;

int main()
{
  int n,m,weight,x,y;
  cout << "kil versun:";cin >> n;
  cout << "kil reber:";cin >> m;
  vector < pair < int, pair<int, int> > > g;

  for (int i = 0; i < m; i++) {
    cout << "Rebro[" << i << "] = "<<endl;
    cout << "ver1:";cin >> x;
    cout << "ver2:";cin >> y;
    cout << "weight:";cin >> weight;
    g.push_back({weight, {--x,--y}});
  }
```

```
  int cost = 0;
  vector < pair<int, int> > res;

  sort(g.begin(), g.end());
  vector<int> tree_id(n);


  for (int i = 0; i < n; ++i)
    tree_id[i] = i;
  for (int i = 0; i < m; ++i)
  {
    int a = g[i].second.first, b = g[i].second.second, l = g[i].first;
    if (tree_id[a] != tree_id[b])
    {
      cost += l;
      res.push_back(make_pair(a, b));
      int old_id = tree_id[b], new_id = tree_id[a];
      for (int j = 0; j < n; ++j)
        if (tree_id[j] == old_id)
          tree_id[j] = new_id;
    }
  }


  for (auto index : res) {
    cout << index.first +1<< " - " << index.second +1<< endl;;
  }
    return 0;
}
```

```
0 - 1
8 - 10
0 - 2
3 - 5
9 - 10
0 - 3
1 - 4
5 - 9
1 - 6
7 - 10
Program ended with exit code: 0
```

3)За методом Прима:

17)



1)(1;2)
2)(1;4)

2)(2;5)
3)(2;7)
4)(1;3)
5)(4;6)
6)(6;10)
7)(10;11)
6)(9;11)
8)(11;8)
Вага = 1+3+3+4+2+2+3+2+1+4 = 25

```
#define _CRT_SECURE_NO_WARNINGS
#include<iostream>
#include<iomanip>

using namespace std;
int main()
{
  setlocale(LC_ALL, "ukr");
  int INF = 10000;

  int vertex, edges;
  cout << "Кількість вершин: ";
  cin >> vertex;
  cout << "Кількість ребер: ";
  cin >> edges;
  int** a = new int* [vertex];


  int* d = new int[vertex];
  int* visited = new int[vertex];
  int temp, minindex, min;




    for (int i = 0; i < vertex; i++)
    {
      a[i] = new int[vertex];
    }


  for (int i = 0; i < vertex; i++)
  {
    for (int j = 0; j < vertex; j++) {
      a[i][j] = 0;
    }
  }


  for (int i = 0; i < edges; i++)
  {
    int x, y, weight;
    cout << "Вершина(1): ";
    cin >> x;
    cout << "Вершина(2): ";
    cin >> y;
```

```
  cout << "Вага: ";
  cin >> weight;
  a[--x][--y] = weight;
  a[y][x] = weight;

}



cout << "Матриця суміжності: " << endl;
for (int i = 0; i < vertex; i++)
{
  for (int j = 0; j < vertex; j++)
    cout << a[i][j] << " ";
  cout << endl;
}
cout << endl;


for (int i = 0; i < vertex; i++)
{
  d[i] = INF;
  visited[i] = 1;
}

int start, finish;
cout << "Початкова вершина :";
cin >> start;
start--;
cout << "Кінцева вершина: ";
cin >> finish;
finish--;
int begin_index = start;
d[begin_index] = 0;


  do {
  minindex = INF;
  min = INF;
  for (int i = 0; i < vertex; i++)
  {
    if ((visited[i] == 1) && (d[i] < min))
    {
      min = d[i];
      minindex = i;
    }
  }
  if (minindex != INF)
  {
    for (int i = 0; i < vertex; i++)
    {
      if (a[minindex][i] > 0)
      {
        temp = min + a[minindex][i];
        if (temp < d[i])
        {
```

```
            d[i] = temp;
          }
        }
      }
      visited[minindex] = 0;
    }


  } while (minindex < INF);


  cout << "Мінімальні шляхи до вершин: " << endl;
  for (int i = 0; i < vertex; i++)cout << d[i] << " ";


  if (d[finish] != INF) {

    int* ver = new int[vertex];
    int end = finish;
    ver[0] = end + 1;
    int k = 1;
    int weight = d[end];

    while (end != begin_index)
    {
      for (int i = 0; i < vertex; i++)
        if (a[end][i] != 0)
        {
          int temp = weight - a[end][i];
          if (temp == d[i])
          {
            weight = temp;
            end = i;
            ver[k] = i + 1;
            k++;
          }
        }
    }


    cout << endl << "Мінімальний шлях від " << ++start << " до " <<
++finish << ": " << endl;
    for (int i = k - 1; i >= 0; i--)cout << ver[i] << " ";


  }
  else { cout << endl << "Не існує такого шляху"; }
  return 0;
}
```

Завдання № 6

Розв’язати задачу комівояжера для повного 8-ми вершинного графа методом «іди у найближчий», матриця вагів якого має вигляд:

17)

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

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**min**
1	**-**								**1**
2		**-**							**1**
3			**-**						**5**
4				**-**					**1**
5					**-**				**1**
6						**-**			**1**
7							**-**		**1**
8								**-**	**2**

1)Редукція за рядками:

	1	2	3	4	5	6	7	8
1	-	5	5	5	0	2	0	2
2	5	-	4	4	0	5	0	4
3	1	0	-	2	2	2	2	0
4	5	4	6	-	5	4	0	1
5	0	0	6	5	-	5	5	5
6	2	5	6	4	5	-	0	1
7	0	0	6	0	5	0	-	1
8	1	3	3	0	4	0	0	-
min	0	0	3	0	0	0	0	

2)Редукція за стовпцями:

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	**-**	**5**	**2**	**5**	**0**	**2**	**0**	**2**
2	**5**	**-**	**1**	**4**	**0**	**5**	**0**	**4**
3	**1**	**0**	**-**	**2**	**2**	**2**	**2**	**0**
4	**5**	**4**	**3**	**-**	**5**	**4**	**0**	**1**
5	**0**	**0**	**3**	**5**	**-**	**5**	**5**	**5**
6	**2**	**5**	**3**	**4**	**5**	**-**	**0**	**1**
7	**0**	**0**	**3**	**0**	**5**	**0**	**-**	**1**
8	**1**	**3**	**0**	**0**	**4**	**0**	**0**	**-**

Метод спроб і помилок:

Фіксуємо 0 в (1;7),решту 0 – викреслюємо в 1 рядку 7 стовбці

Фіксуємо 0 в (2;5),решту 0 – викреслюємо в 2 рядку 5 стовбці

Фіксуємо 0 в (3;8),решту 0 – викреслюємо в 3 рядку 8 стовбці

	1	2	3	4	5	6	7	8
1	-	5	2	5		2	0	2
2	5	-	1	4	0	5		4
3	1		-	2	2	2	2	0
4	5	4	3	-	5	4		1
5	0	0	3	5	-	5	5	5
6	2	5	3	4	5	-		1
7	0	0	3	0	5	0	-	1
8	1	3	0	0	4	0		-

Не існує системи із 8 незакінчених циклів – рішення недопустиме.

3)Модифікація матриці:
Викреслюємо рядки і стовпці з найбільшою кількістю нулів: 3,5,7,8 рядки, 2,5,7 стовпці.

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

Min(2,5,1,3,4) = 1.

Віднімаємо 1 від всіх елементів:

	1	2	3	4	5	6	7	8
1	-		1	4		1		1
2	4		0	3		4		3
3								
4	4		2	-		3		0
5								
6	1		2	3		-		0
7								
8								

До мінімального додаємо елементи,які лежать на перетині вершин рядків і стовпчиків:

	1	2	3	4	5	6	7	8
1	-	5	1	4	0	1	0	1
2	4	-	0	3	0	4	0	3
3	1	1	-	2	3	2	3	0
4	4	4	2	-	5	3	0	0
5	0	1	3	5	-	5	6	5
6	1	5	2	3	5	-	0	0
7	0	1	3	0	6	0	-	1
8	1	4	0	0	5	0	1	-

Редукція за рядками:

	1	2	3	4	5	6	7	8
1	-	4	1	4	0	1	0	1
2	4	-	0	3	0	4	0	3
3	1	0	-	2	3	2	3	0
4	4	3	2	-	5	3	0	0
5	0	0	3	5	-	5	6	5
6	1	4	2	3	5	-	0	0
7	0	0	3	0	6	0	-	1
8	1	3	0	0	5	0	1	-

Метод спроб і помилок:
Фіксуємо:(1;5).Решту нулів в рядку 1 і 5 стовпці викреслюємо.
Фіксуємо:(2;3).Решту нулів в рядку 2 і 3 стовпці викреслюємо.
Фіксуємо:(3;2).Решту нулів в рядку 3 і 2 стовпці викреслюємо.
Фіксуємо:(4;7).Решту нулів в рядку 4 і 7 стовпці викреслюємо.
Фіксуємо:(5;1).Решту нулів в рядку 5 і 1 стовпці викреслюємо.
Фіксуємо:(6;8).Решту нулів в рядку 6 і 8 стовпці викреслюємо.
Фіксуємо:(7;4).Решту нулів в рядку 7 і 4 стовпці викреслюємо.
Фіксуємо:(8;6).Решту нулів в рядку 8 і 6 стовпці викреслюємо.

	1	2	3	4	5	6	7	8
1	-	4	1	3	0	1		1
2	4	-	0	3		4		3
3	1	0	-	2	3	2	3	
4	3	3	2	-	5	3	0	
5	0		3	5	-	5	6	5
6	1	4	2	3	5	-		0
7			3	0	6		-	1
8	1	3			5	0	1	-

Кількість нулів – 8.Отримали

	1	2	3	4	5	6	7	8
1	-	4	1	4	0	1	0	1
2	4	-	0	3	0	4	0	3
3	1	0	-	2	3	2	3	0
4	4	3	2	-	5	3	0	0
5	0	1	3	5	-	5	6	5
6	1	4	2	3	5	-	0	0
7	0	0	3	0	6	0	-	1
8	1	3	0	0	5	0	1	-

C min = 2+1+5+5+1+1+1+2 = 18
Шлях - (4,8),(1,5),(2,3),(3,2),(5,1),(6,7),(7,4),(8,6)

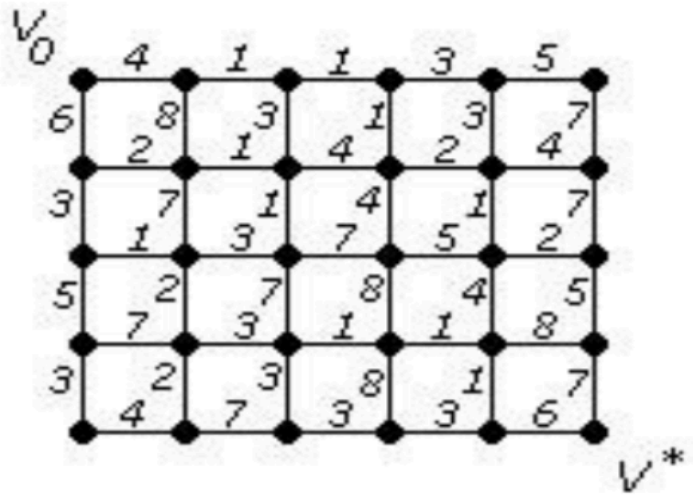
```
#include <vector>
#include<iostream>
using namespace std;
#define vr 4
int TSP(int grph[][vr], int p)
   vector<int> ver;
   for (int i = 0; i < vr; i++)
      if (i != p)
         ver.push_back(i);
         int m_p = INT_MAX;
   do {
      int cur_pth = 0;
      int k = p;
      for (int i = 0; i < ver.size(); i++) {
         cur_pth += grph[k][ver[i]];
         k = ver[i];
      }
      cur_pth += grph[k][p];
      m_p = min(m_p, cur_pth);
   }
   while (next_permutation(ver.begin(), ver.end()));
   return m_p;
}
int main() {
   int grph[][vr] = { { 0, 5, 10, 15 },
      { 5, 0, 20, 30 },
      { 10, 20, 0, 35 },
      { 15, 30, 35, 0 }
   };
```

```
    int p = 0;
    cout<< "\n The result is: "<< TSP(grph, p) << endl;
    return 0;
}
```

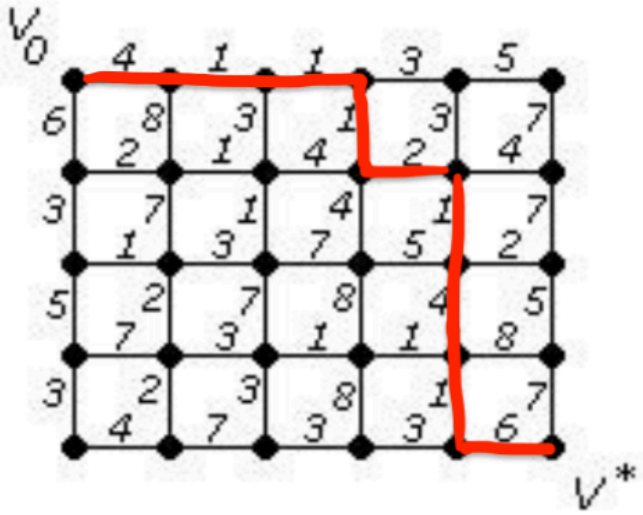
Завдання № 7

За допомогою алгоритму Дейкстри знайти найкоротший шлях у графі між парою вершин V_0 і V^*.

17)



17)



(V1,V2)→(V2,V3)→(V3,V4)→(V4,V10)→(V10,V11)→(V11, V17)→(V17,V23)→(V23,V29)→(V29,V30)
Довжина = 4+1+1+1+2+1+4+1+6 = 21.

```
#define _CRT_SECURE_NO_WARNINGS
#include<iostream>
using namespace std;
int main()
{
  int vertex,edges;
  cout << "Count of vertex: ";
  cin >> vertex;
  cout << "Count of edges: ";
  cin >> edges;
  int **a=new int*[vertex];
  int *d=new int[vertex];
  int *visited=new int[vertex];
  int temp, minindex, min;
  int start, finish;
  cout << "From vertex :";
  cin >> start;
  start--;
  cout << "To: ";
  cin >> finish;
  finish--;
  int begin_index = start;
  for (int i = 0; i < vertex; i++)
  {
    a[i] = new int [vertex];
  }
  for (int i = 0; i < vertex; i++)
  {
    for (int j = 0; j < vertex; j++) {
a[i][j] = 0; }
}
  for (int i = 0; i < edges; i++)
  {
    int x, y,weight;
    cout << "Edge[" << i << "]=" << endl;
  cout << "Vertex1: ";
  cin >> x;
  cout << "Vertex2: ";
  cin >> y;
  cout << "Weight: ";
  cin >> weight;
  a[--x][--y] =weight;
  a[y][x] = weight;
}
for (int i = 0; i < vertex; i++)
{
  for (int j = 0; j < vertex; j++)
    cout<< a[i][j]<<" ";
  cout << endl;
}
```

```
for (int i = 0; i < vertex; i++)
{
d[i] = 10000;
  visited[i] = 1;
}
d[begin_index] = 0;
do {
  minindex = 10000;
  min = 10000;
  for (int i = 0; i < vertex; i++)
  {
    if ((visited[i] == 1) && (d[i] < min))
{
min = d[i];
      minindex = i;
    }
  }
  if (minindex != 10000)
  {
    for (int i = 0; i < vertex; i++)
    {
      if (a[minindex][i] > 0)
      {
        temp = min + a[minindex][i];
        if (temp < d[i])
{
d[i] = temp;
} }
}
    visited[minindex] = 0;
  }
} while (minindex < 10000);

cout<<"Minimal ways to vertex: "<<endl;
for (int i = 0; i < vertex; i++)cout << d[i] << " ";
bool flag = false;
for (int i = 0; i < vertex; i++)if(d[i]!=0&&d[i]!=10000)flag = true;
if (flag) {
    int* ver = new int[vertex];
    int end = finish;
    ver[0] = end + 1;
    int k = 1;
    int weight = d[end];
    while (end != begin_index)
    {
      for (int i = 0; i < vertex; i++)
        if (a[end][i] != 0)
        {
          int temp = weight - a[end][i];
          if (temp == d[i])
```

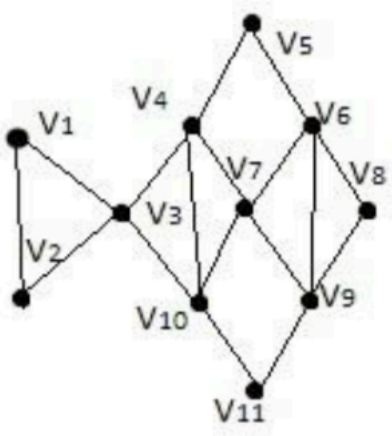
```
            {
              weight = temp;
              end = i;
              ver[k] = i + 1;
              k++;
} }
    }
    cout << endl << "Print minimal way" << endl;
for (int i = k - 1; i >= 0; i--)cout << ver[i] << " ";
} else {
    cout <<endl<< "There isn`t such way";
  }
return 0; }
```

```
Minimal ways to vertex:
0 4 5 6 9 14 6 8 8 7 9 13 9 9 11 10 15 17 14 11 14 15 16 22 17 13 17 20 17 23
Print minimal way
1 2 3 9 14 20 21 22 23 29 30 Program ended with exit code: 0
```

Завдання № 8

Знайти ейлеровий цикл в ейлеровому графі двома методами: а) Флері; б) елементарних циклів.

17)



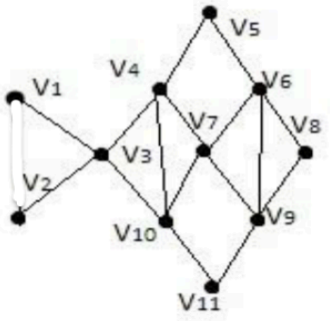
Це Ейлеровий граф, тому що степінь кожної вершини є парним числом. Тому ми можемо використати алгоритм Флері. Тобто нам треба пройти по всіх ребрах і ми можемо пройти по кожному ребру лише один раз.

Починаємо від вершини V1:

А)

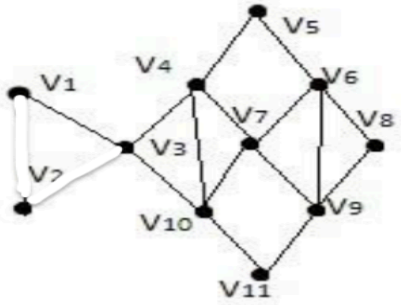
1→2

17)



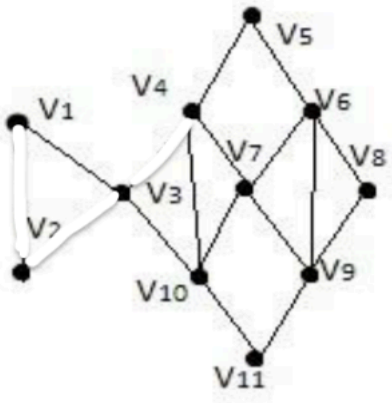
2→3

17)



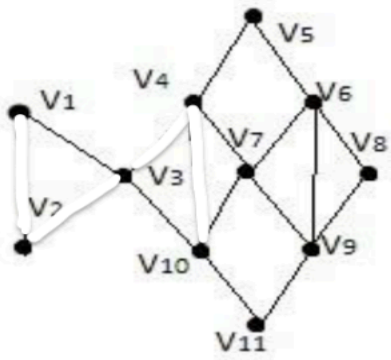
3→4

17)



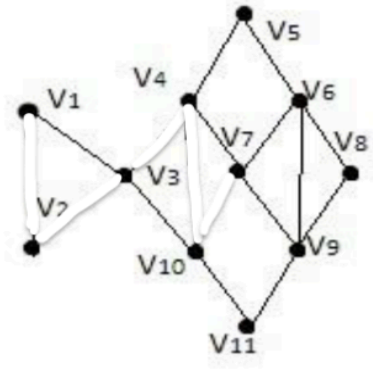
4→10

17)



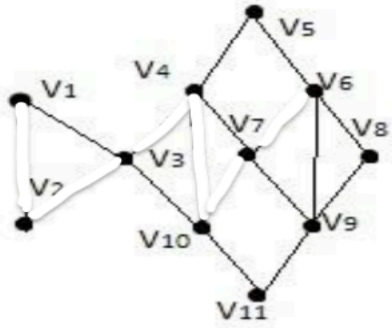
10→7

17)



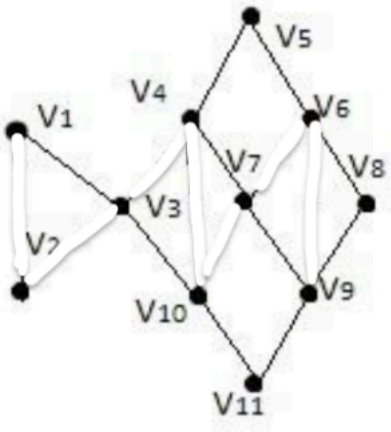
7→6

17)



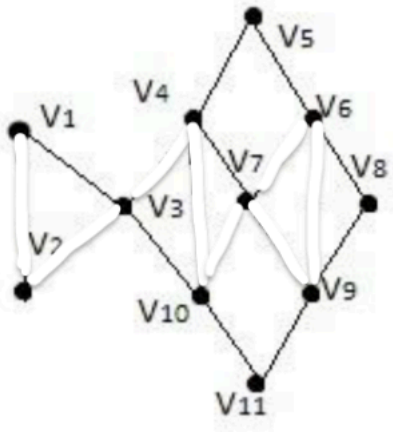
6→9

17)



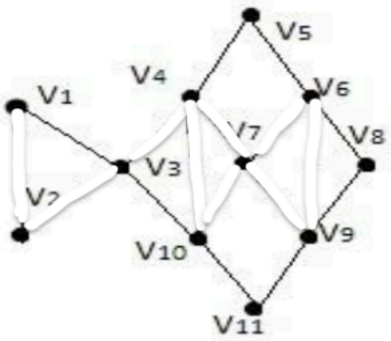
9→7

17)



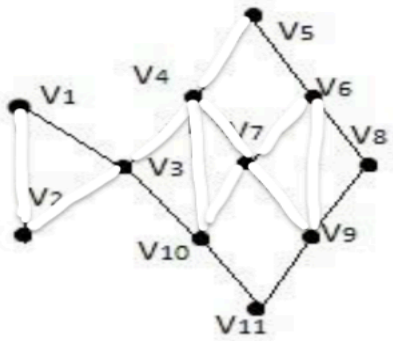
7→4

17)



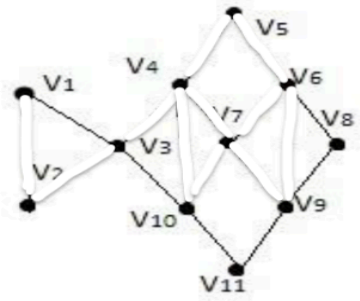
4→5

17)



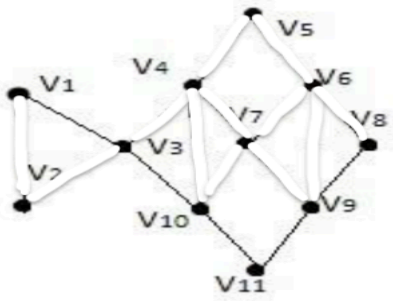
5→6

17)



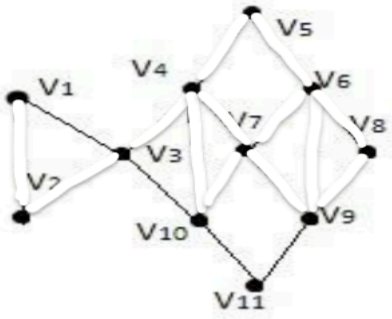
6→8

17)



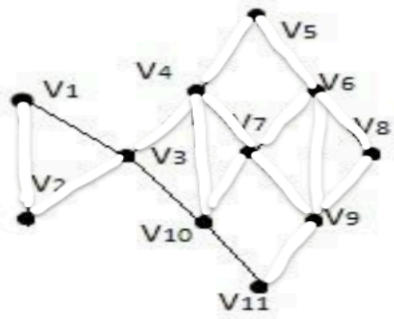
8→9

17)



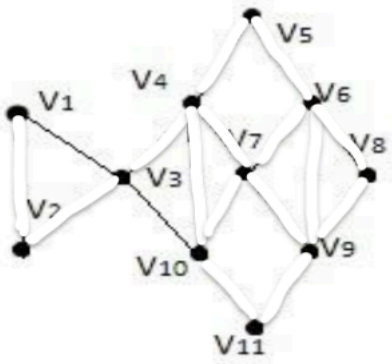
9→11

17)



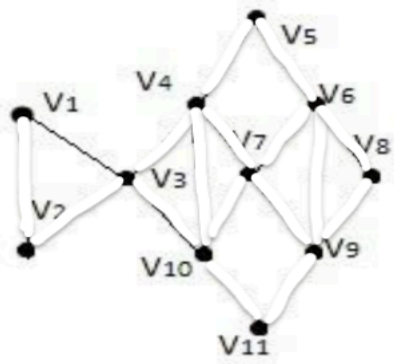
11→10

17)



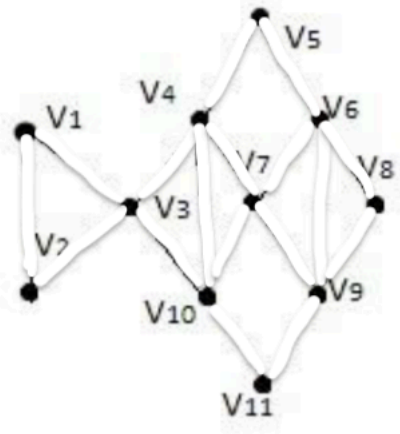
10→3

17)



3→1

17)



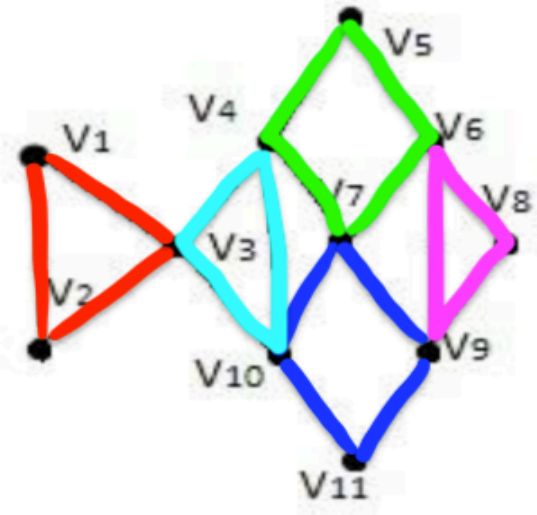
1→2→3→4→10→7→6→9→7→4→5→6→8→9→11→10→3→1

Довжина = 17

Б)

Метод елементарних циклів:

17)



Алгоритм Флері:

```
#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];  }
  ~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);
};


   degree vertex (if there is any) and then calls printEulerUtil()
   to print the path */
void Graph::printEulerTour()
{

  int u = 0;
  for (int i = 0; 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;
}
```

Завдання №9

Спростити формули (привести їх до скороченої ДНФ).

17. $x\bar{y} \vee \bar{x}\bar{z} \vee yz$

x,y,z	¬y	x¬y	¬x	¬z	¬x¬z	x¬y V ¬x¬z	yz	x¬y V ¬x¬z V yz
0,0,0	1	0	1	1	1	1	0	1
0,0,1	1	0	1	0	0	0	0	0
0,1,0	0	0	1	1	1	1	0	1
0,1,1	0	0	1	0	0	0	1	1
1,0,0	1	1	0	1	0	1	0	1
1,0,1	1	1	0	0	0	1	0	1
1,1,0	0	0	0	1	0	0	0	0
1,1,1	0	0	0	0	0	0	1	1

f(x,y,z) = 1,при {0,0,0};{0,1,0};{0,1,1};{1,0,0};{1,0,1},{1,1,1}

K1:{0,0,0} → ¬x¬y¬z
K2:{0,1,0} → ¬xy¬z
K3:{0,1,1} → ¬xyz
K4:{1,0,0} → x¬y¬z
K5:{1,0,1} → x¬yz
K6:{1,1,1} → xyz

K1 v K2 v K3 v K4 v K5 v K6 = ¬x¬y¬z V ¬xy¬z V ¬xyz V x¬y¬z V x¬yz V xyz.zz

Напишіть алгоритм

64. «Іди в найближчий» для розв'язання задачі комівояжера.

```
#include <iostream>
using namespace std;
#define vr 4
int TSP(int grph[][vr], int p)
   vector<int> ver; //
   for (int i = 0; i < vr; i++)
      if (i != p)
         ver.push_back(i);
         int m_p = INT_MAX;
   do {
      int cur_pth = 0;
      int k = p;
      for (int i = 0; i < ver.size(); i++) {
         cur_pth += grph[k][ver[i]];
         k = ver[i];
      }
      cur_pth += grph[k][p];
      m_p = min(m_p, cur_pth);
   }
   while (next_permutation(ver.begin(), ver.end()));
   return m_p;
}
int main() {
   int grph[][vr] = { { 0, 5, 10, 15 },
      { 5, 0, 20, 30 },
      { 10, 20, 0, 35 },
      { 15, 30, 35, 0 }
   };
   int p = 0;
   cout<< "\n The result is: "<< TSP(grph, p) << endl;
   return 0;
}
```