МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
Інститут комп'ютерних наук та інформаційних технологій

Кафедра систем штучного інтелекту

Розрахункова робота
з курсу “Дискретна математика”

Виконала:
студентка групи КН-114
Церковник Оксана

Викладач:
Мельникова Н.І.

Львів – 2019

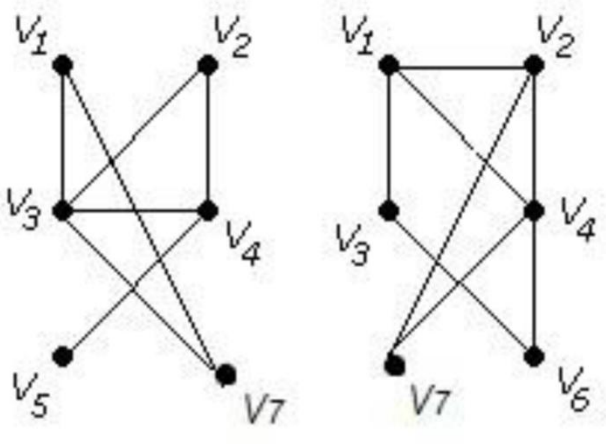
Варіант - 18

Завдання No 1

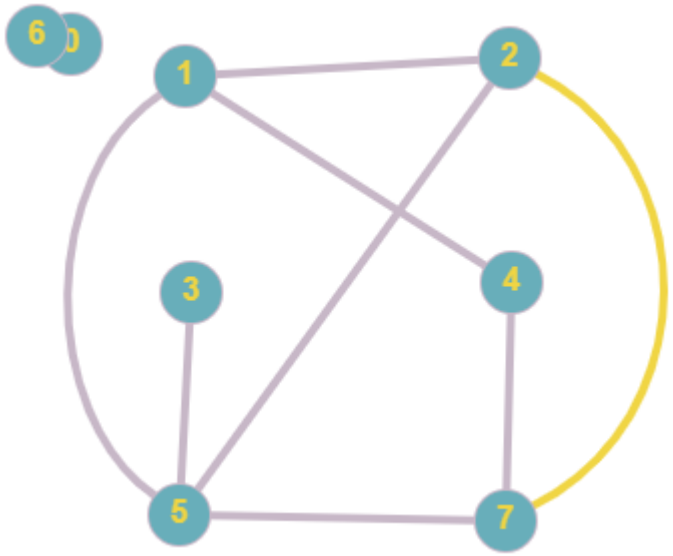
Виконати наступні операції над графами:

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.

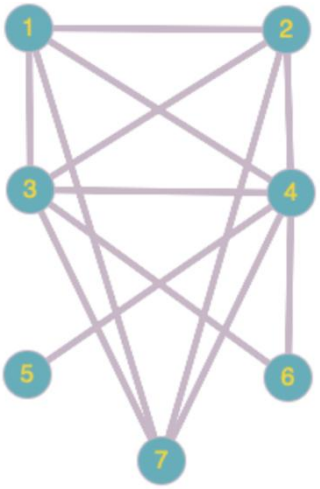
Задані графи



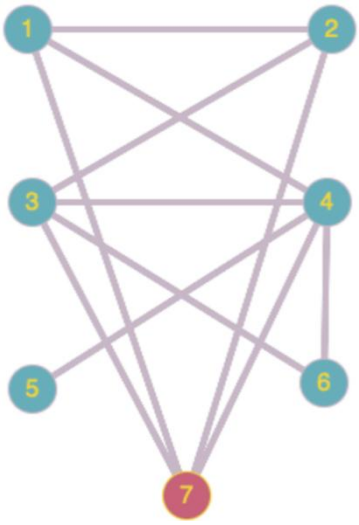
1)



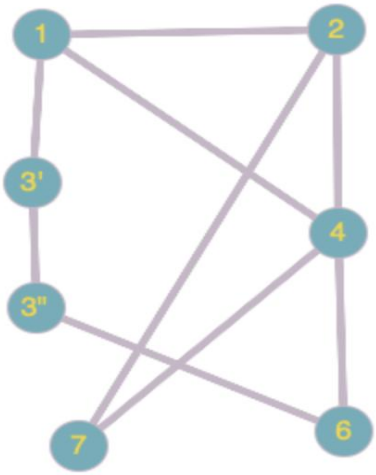
2)



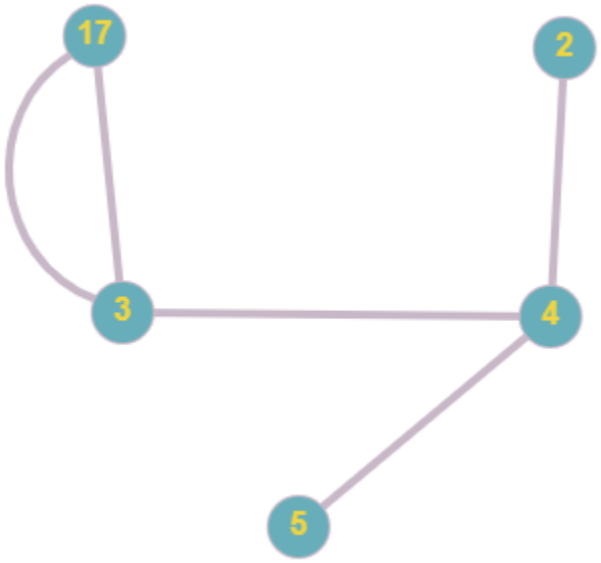
3)

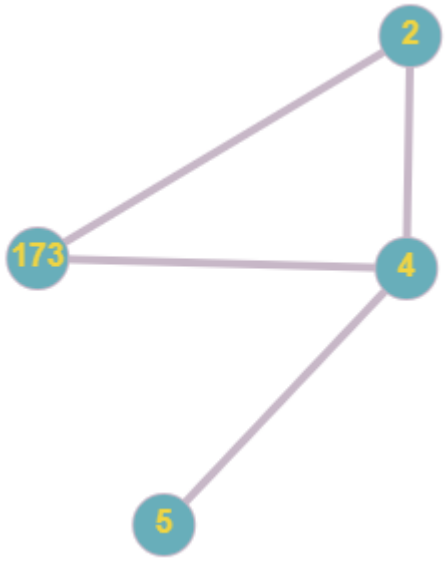


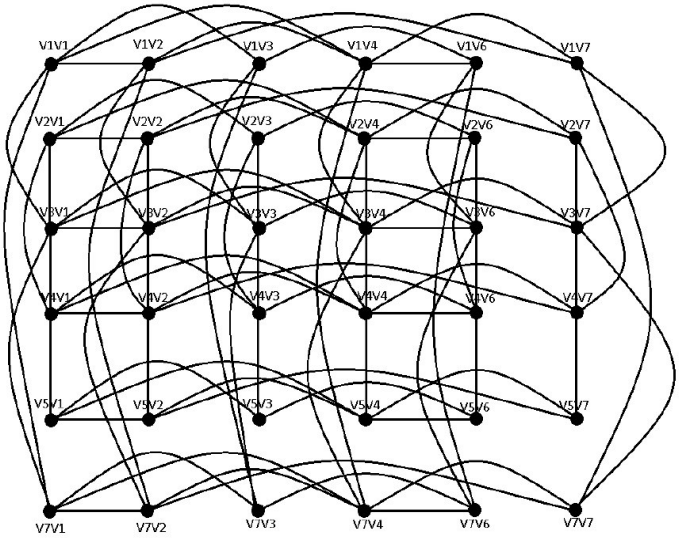
4)



5)





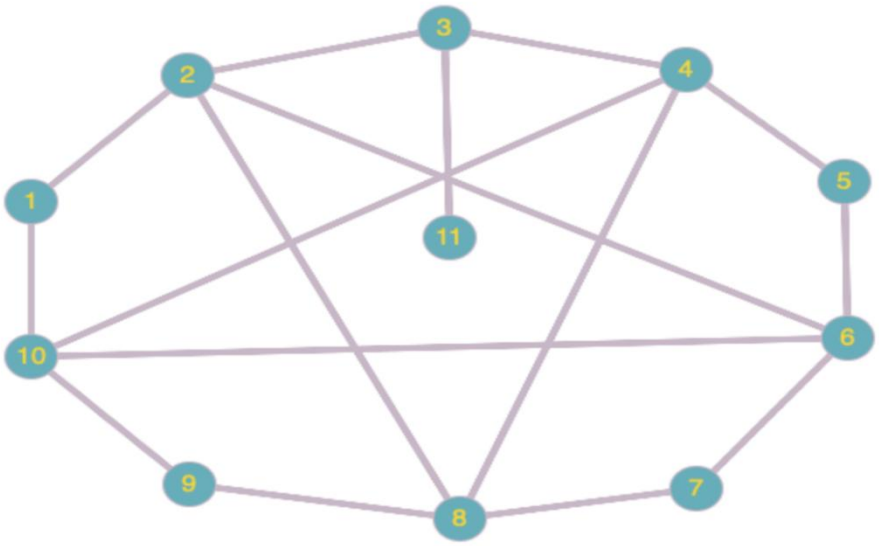


6)

Завдання N2

Скласти таблицю суміжності для орграфа.

Заданий граф



Таблиця суміжності:

	1	2	3	4	5	6	7	8	9	10	11
1	0	1	0	0	0	0	0	0	0	1	0
2	1	0	1	0	0	1	0	1	0	0	0
3	0	1	0	1	0	0	0	0	0	0	1
4	0	0	1	0	1	0	0	1	0	1	0
5	0	0	0	1	0	1	0	0	0	0	0
6	0	1	0	0	1	0	1	0	0	1	0
7	0	0	0	0	0	1	0	1	0	0	0
8	0	1	0	1	0	0	1	0	1	0	0
9	0	0	0	0	0	0	0	1	0	1	0
10	1	0	0	1	0	1	0	0	1	0	0
11	0	0	1	0	0	0	0	0	0	0	0

Завдання No 3

Для графа з другого завдання знайти діаметр.

Діаметр = 4

Завдання 4

Обхід дерева вшир

V1	1	V1
V2	2	V1V2
V10	3	V1V2V10
-	-	V2V10

V3	4	V2V10V3
V6	5	V2V10V3V6
V8	6	V2V10V3V6V8
-	-	V10V3V6V8
V4	7	V10V3V6V8V4
V9	8	V10V3V6V8V4V9
-	-	V3V6V8V4V9
V11	9	V3V6V8V4V9V11
-	-	V6V8V4V9V11
V7	10	V6V8V4V9V11V7
V5	11	V6V8V4V9V11V7V5
-	-	V8V4V9V11V7V5
-	-	V4V9V11V7V5
-	-	V9V11V7V5
-	-	V11V7V5
-	-	V7V5
-	-	V5
-	-	∅

```
#include <iostream>

#include <queue>



using namespace std;

bool check(int *Check,int j,int N) {//перевіряємо чи не було вже такої вершини в черзі

 for (int i = 0; i < N; i++) {

  if (Check[i] == j) {

   return false;

  }

 }

 return true;

}

int main()

{

 int Start, N,k=1;

 int** Graf;

 queue <int> qq;
```

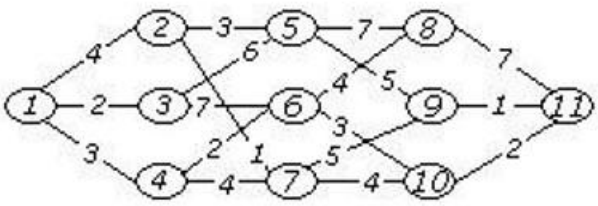
```
cin >> N >> Start;

Graf = new int* [N];

int* Check = new int[4*N];//масив з елементами для перевірки(4N бо тут багато зайвих елементів)



qq.push(Start-1);//наш стек починається із заданої вершини

Check[0] = Start-1;

 for (int i = 0; i < N; i++) {

  Graf[i] = new int[N];

 }



 for (int i = 0; i < N; i++) {

  for (int j = 0; j < N; j++) {

   cin >> Graf[i][j];

  }

 }



 for (int i = 0; i < N; i++) {

  for (int j = 0; j < N; j++) {

   if (Graf[qq.front()][j]) {



    if (check(Check, j, 4 * N)) {

     qq.push(j);//якщо  вершину ще не проходили додаємо її в чергу
```

```
    }



    Check[k] = j;

    k++

   }

  }

  cout << ++qq.front()<<" ";



  qq.pop();//після того як закінчились суміжні вершини видаляєм перший елемент черги

 }



return 0;

}
```

Завдання No 5

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

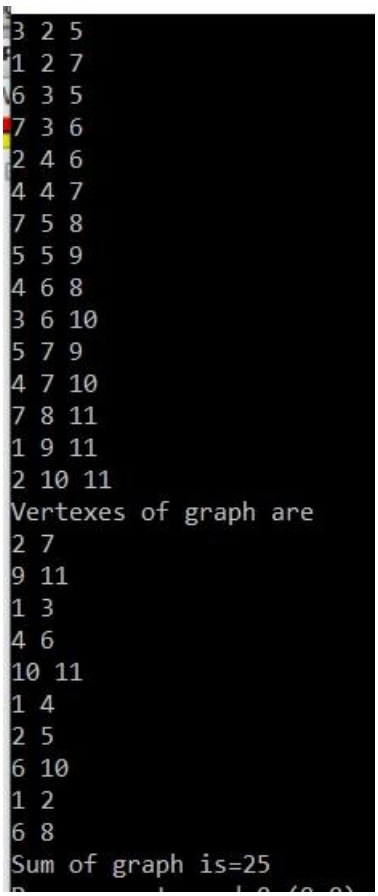
18)



Метод Краскала

1. Ребро 2-7 , Вага=1
2. Ребро 9-11, Вага=1
3. Ребро 1-3, Вага=2
4. Ребро 10-11, Вага=2
5. Ребро 4-6, Вага=2
6. Ребро 6-10, Вага=3
7. Ребро 1-4, Вага=3
8. Ребро 2-5, Вага=3
9. Ребро 4-7, Вага=4
10. Ребро 6-8, Вага=4

```
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;
int main() {
   int m=18,n=11;
   vector < pair < int, pair<int,int> > > g (m); // вага - вершина 1 - вершина 2
   cout<<"Input your graph(1 is weight)(2 & 3 are vertexes:)\n";
      for(int i=0;i<m;i++)
      cin>>g[i].first>> g[i].second.first>>g[i].second.second;

   int cost = 0,l=0;
   vector < pair<int,int> > res;
   sort (g.begin(), g.end());
   vector<int> tree_id (n);
   for (int i=0; i<n-1; ++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[a],  new_id = tree_id[b];
         for (int j=0; j<n; ++j)
            if (tree_id[j] == old_id)
               tree_id[j] = new_id;
      }
   }
   cout<<"Vertexes of graph are\n";
   for (int i = 0; i < n -1 ; i++)
     cout << res[i].first << " " << res[i].second<<endl;
   cout<<"Sum of graph is="<<cost;
}
```



Метод Прима

1. Ребро 1-3, Вага=2
2. Ребро 1-4, Вага=3
3. Ребро 4-6, Вага=2
4. Ребро 6-10, Вага=3
5. Ребро 10-11, Вага=2
6. Ребро 11-9, Вага=1
7. Ребро 1-2, Вага=4
8. Ребро 2-7, Вага=1
9. Ребро 2-5, Вага=3
10. Ребро 6-8, Вага=4

Шлях=25

```
#include<iostream>
using namespace std;
int main()
{
    int a,b,u,v,n,i,j,ne=1;
int visited[20]={0},min,mincost=0,cost[20][20];
 int path[100]={0};
 int path_index=0;

 cout<<"Enter the number of vertices: ";
 cin>>n;
 cout<<"Enter the adjacency matrix: " << endl;

  for(i=1;i<=n;i++){
 for(j=1;j<=n;j++)
 {
  cin>>cost[i][j];
  if(cost[i][j]==0)
    cost[i][j]=99;
 }
  }
 visited[1]=1;
 cout<<endl;;

 while(ne < n)
 {
  for(i=1,min=99;i<=n;i++)
  for(j=1;j<=n;j++)
  if(cost[i][j]< min)
  if(visited[i]!=0)
  {
    min=cost[i][j];
    a=u=i;
    b=v=j;
  }
  if(visited[u]==0 || visited[v]==0)
  {
    path[path_index]=b;
    path_index++;
    ne++;
    mincost+=min;
    visited[b]=1;

  }
  cost[a][b]=cost[b][a]=99;
 }
```

```
 cout<<endl;

 cout<<1<<" -> ";
 for (int i=0;i<n-1;i++)
 {
  cout<<path[i];
  if (i<n-2) cout<<" -> ";
 }

 cout<< endl << "The least way is: "<<mincost;


 cin.get();
 cin.get();
 return 0;
}
```

```
11
0 4 2 3 0 0 0 0 0 0 0
  4 0 0 0 3 0 1 0 0 0 0
  2 0 0 0 6 7 0 0 0 0 0
  3 0 0 0 0 2 4 0 0 0 0
  0 3 6 0 0 0 0 7 5 0 0
  0 0 7 2 0 0 0 4 0 3 0
  0 1 0 4 0 0 0 0 5 4 0
  0 0 0 0 7 4 0 0 0 0 7
  0 0 0 0 5 0 5 0 0 0 1
  0 0 0 0 0 3 4 0 0 0 2
  0 0 0 0 0 0 0 7 1 2 0
1 3
1 4
4 6
6 10
10 11
11 9
1 2
2 7
2 5
6 8
```

Завдання No 6

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

18)

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

	14	2	3	5	6	7	8
14	N	3	5	5	1	2	3
2	3	N	3	2	1	2	2
3	5	3	N	4	5	4	5
5	5	2	4	N	2	2	2
6	1	1	5	2	N	5	6
7	2	2	4	2	5	N	2
8	3	2	5	2	6	2	N

	2	3	5	146	7	8
2	N	3	2	1	2	2
3	3	N	4	5	4	5
5	2	4	N	2	2	2
146	1	5	2	N	5	6
7	2	4	2	5	N	2
8	2	5	2	6	2	N

	1462	3	5	7	8
1462	N	3	2	2	2

3	3	N	4	4	5
5	2	4	N	2	2
7	2	4	2	N	2
8	2	5	2	2	N

	3	14625	7	8
3	N	4	4	5
14625	4	N	2	2
7	4	2	N	2
8	5	2	2	N

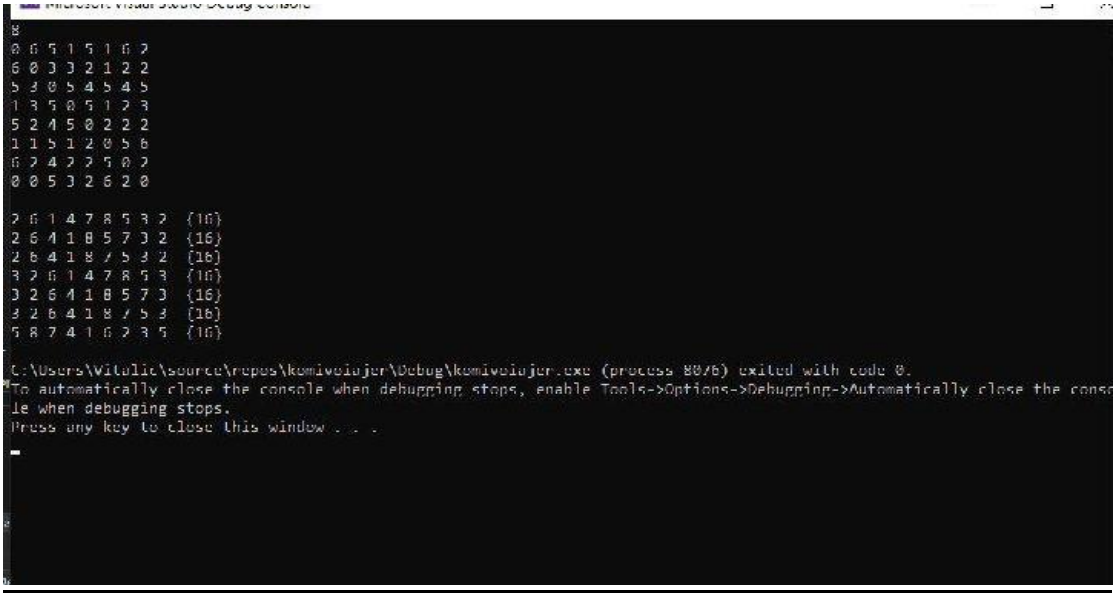
	3	146257	8
3	N	4	5
146257	4	N	2
8	5	2	N

	3	1462578
3	N	5
1462578	5	N

Довжина маршруту =19

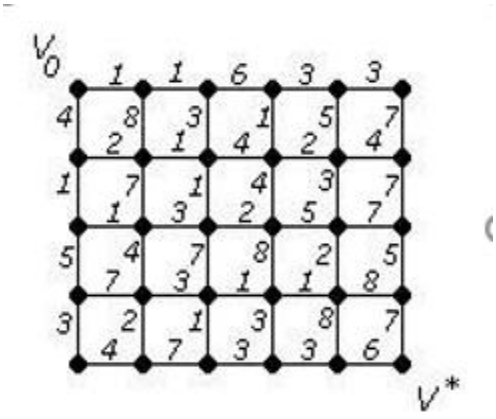
```
#include<iostream>
#include<vector>
using namespace std;
int counter = 0, Inf = 9999;

bool check(vector<int> q, int Node);

int F_Min(vector<int>* q, int** arr, int n, int i);

void Find(vector<int>* q, int** arr, int n, int pos, vector<int>* qq);

int main()
{
 int n;
 cin >> n;
 int** arr = new int* [n];
 for (int i = 0; i < n; i++) {
  arr[i] = new int[n];
 }

 for (int i = 0; i < n; i++) {
  for (int j = 0; j < n; j++) {
   cin >> arr[i][j];
  }
 }

 vector<int> q;
 vector<int> qq;
 cout << endl;

 for (int i = 0; i < n; i++) {
  q.clear();
  q.push_back(i);
  Find(&q, arr, n, i, &qq);
 }

 for (int i = 1; i <= qq.size(); i++) {
  if (i != 0 && i % (n + 2) == 0)
   cout << " {" << qq[i - 1] << "}" << endl;
  else
   cout << qq[i - 1] + 1 << " ";
 }
 return 0;
}




bool check(vector<int> q, int Node) {
 for (auto i = q.begin(); i != q.end(); i++)
  if (*i == Node)return false;
 return true;
}


int F_Min(vector<int>* q, int** arr, int n, int i) {
 int min = 999;
 for (int j = 0; j < n; j++) {
```

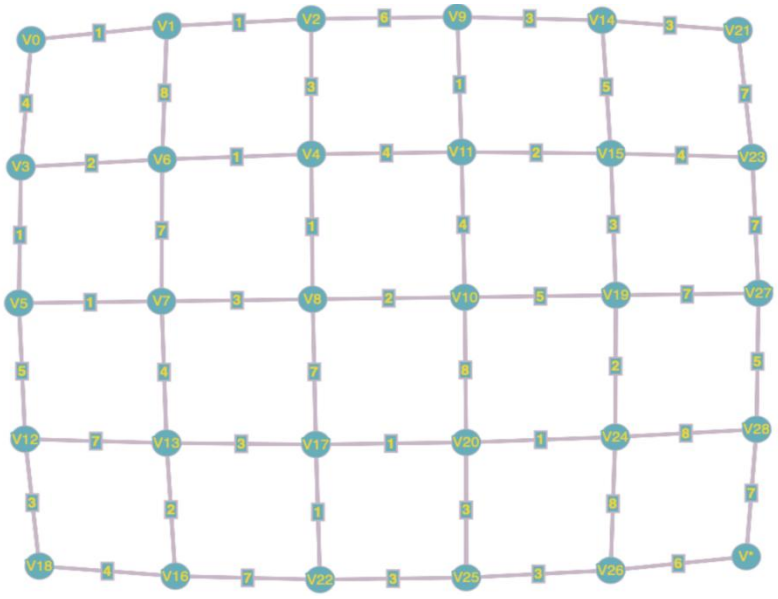
```
  if (arr[i][j] < min && arr[i][j] != 0 && check((*q), j))min = arr[i][j];
 }
 return min;
}

void Find(vector<int>* q, int** arr, int n, int pos, vector<int>* qq) {
 int min;
 for (int i = pos, k = 0; k < 1; i++, k++) {
  min = F_Min(q, arr, n, i);
  for (int j = 0; j < n; j++) {
   if (arr[i][j] == min && check((*q), j)) {
    (*q).push_back(j);
    Find(q, arr, n, j, qq);
   }
  }
  if (q->size() == n) {
   (*q).push_back((*q)[0]);
   counter = 0;

    for (int l = 1; l <= n; l++) {
     counter += arr[(*q)[l - 1]][(*q)[l]];
    }

    if (Inf == counter) {
     for (int b = 0; b <= n; b++) {
      (*qq).push_back((*q)[b]);
     }
     (*qq).push_back(counter);
    }

    else if (Inf > counter) {
     (*qq).clear();

     for (int b = 0; b <= n; b++) {
      (*qq).push_back((*q)[b]);

     }
     (*qq).push_back(counter);

     Inf = counter;
    }
   q->pop_back();
  }
 }
 q->pop_back();
}
```



Завдання No 7

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





V1=1 | V2=2 | V3=4 | V4=5 | V5=5;

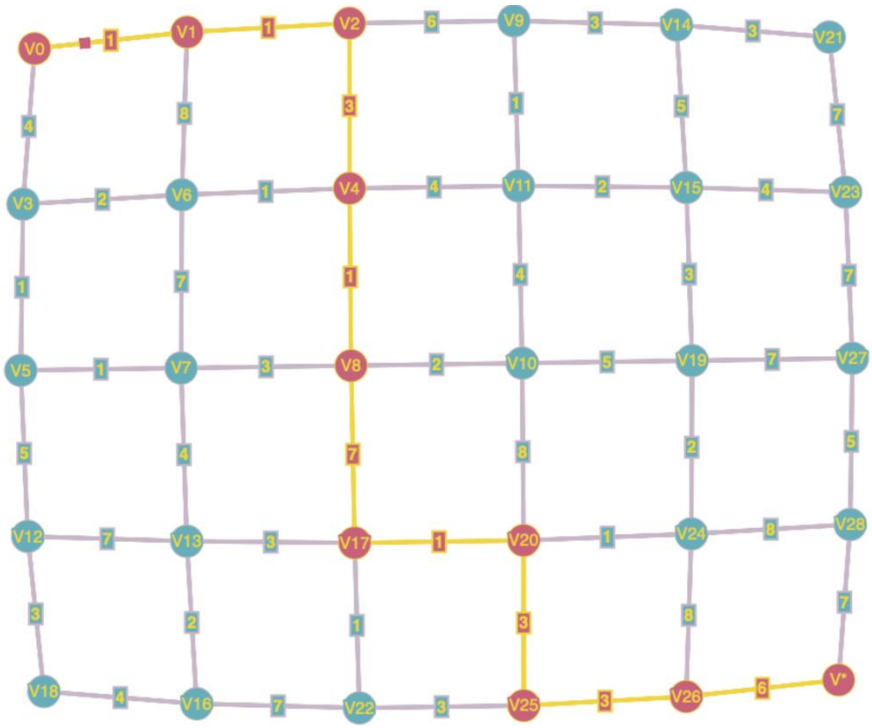
V6=6 | V7=6 | V8=6 | V9=8 | V10=8;

V11=9 | V12=10 | V13=10 | V14=11 | V15=11;

V16=12 | V17=13 | V18=13 | V19=13 | V20=14;

V21=14 | V22=14 | V23=15 | V24=15 | V25=17;

V26=20 | V27=20 | V28=23 | V*=26




```
#include<iostream>
using namespace std;
int n;
int i, j, q;
int dist[40];
bool visited[40];
int pred[40];
void createGraph(int c[40][40])
{
  int g1, g2;
  cout << "Enter the number of vertices: ";
  cin >> n;
  for (int i = 0; i < n; i++) {

    for (int j = 0; j < n; j++)
    {
      c[i][j] = 0;
    }
  }
  cout << "Enter size of (n*m) : ";
  cin >> g1 >> g2;

  for (i = 0; i < n; i++) {
    for (j = i + 1; j < n; j++)
    {
      if (j == i + 1 || j == i + g1) {
        cout << "Enter the length from x" << i+1 << " to x" << j+1 << ": ";
        cin >> c[i][j];
      }
      else {
```

```
            c[i][j] = 0;
        }
    }
}
int minDistance()
{
  int minimum = 10000, minDist;
  for (int v = 0; v < n; v++)
    if (visited[v] == false && dist[v] <= minimum)
    {
      minimum = dist[v];
      minDist = v;
    }
  return minDist;
}
void printPath(int j)
{
  if (pred[j] == -1)
    return;
  printPath(pred[j]);
  cout << "X" << j+1 << " -> ";
}
void dijkstra(int c[40][40])
{
    int start;
    cout << "Enter the first node : ";
    cin >> start;
    for (int i = 0; i < n; i++)
    {
      pred[0] = -1;
      dist[i] = 10000;
      visited[i] = false;
    }
    dist[start-1] = 0;
  for (int count = 0; count < n - 1; count++)
    {
      int u = minDistance();
      visited[u] = true;
      for (int v = 0; v < n; v++)
        if (!visited[v] && c[u][v] &&
            dist[u] + c[u][v] < dist[v])
          {
            pred[v] = u;
            dist[v] = dist[u] + c[u][v];
          }
    }
    cout << "The least way is: ";
    cout << dist[29] << endl;
    cout << "The way is: ";
    cout << "X1 -> ";
    printPath(29);
    cout << "The end!)" << endl;
```

```
    //}
}
int main()
{
 int c[40][40];
 createGraph(c);
 dijkstra(c);
 return 0;
}
```

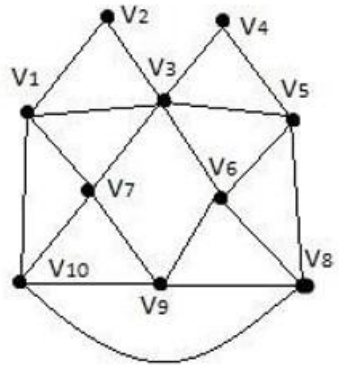
```
Enter the number of vertices: 30
Enter size of (n*m) : 6 5
Enter the length from x1 to x2: 1
Enter the length from x1 to x7: 4
Enter the length from x2 to x3: 1
Enter the length from x2 to x8: 8
Enter the length from x3 to x4: 6
Enter the length from x3 to x9: 3
Enter the length from x4 to x5: 3
Enter the length from x4 to x10: 1
Enter the length from x5 to x6: 3
Enter the length from x5 to x11: 5
Enter the length from x6 to x7: 0
Enter the length from x6 to x12: 7
Enter the length from x7 to x8: 2
Enter the length from x7 to x13: 1
Enter the length from x8 to x9: 1
Enter the length from x8 to x14: 7
Enter the length from x9 to x10: 4
Enter the length from x9 to x15: 1
Enter the length from x10 to x11: 22
Enter the length from x10 to x16: 4
Enter the length from x11 to x12: 4
Enter the length from x11 to x17: 3
Enter the length from x12 to x13: 0
Enter the length from x12 to x18: 7
Enter the length from x13 to x14: 1
Enter the length from x13 to x19: 5
Enter the length from x14 to x15: 3
Enter the length from x14 to x20: 4
Enter the length from x15 to x16: 2
Enter the length from x15 to x21: 7
Enter the length from x16 to x17: 5
Enter the length from x16 to x22: 8
```

```
Enter the length from x16 to x17: 5
Enter the length from x16 to x22: 8
Enter the length from x17 to x18: 7
Enter the length from x17 to x23: 2
Enter the length from x18 to x19: 0
Enter the length from x18 to x24: 5
Enter the length from x19 to x20: 7
Enter the length from x19 to x25: 3
Enter the length from x20 to x21: 3
Enter the length from x20 to x26: 2
Enter the length from x21 to x22: 1
Enter the length from x21 to x27: 1
Enter the length from x22 to x23: 1
Enter the length from x22 to x28: 3
Enter the length from x23 to x24: 8
Enter the length from x23 to x29: 8
Enter the length from x24 to x25: 0
Enter the length from x24 to x30: 7
Enter the length from x25 to x26: 4
Enter the length from x26 to x27: 7
Enter the length from x27 to x28: 3
Enter the length from x28 to x29: 3
Enter the length from x29 to x30: 6
Enter the first node : 1
The least way is: 26
The way is: X1 -> X2 -> X3 -> X9 -> X15 -> X21 -> X27 -> X28 -> X29 -> X30 -> The end!)
```

Завдання No 8

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

А)



1 => 10 => 9 => 8 => 10 => 7 => 9 => 6 => 8 => 5 => 6 => 3 => 5 => 4 => 3 => 7 => 1 => 3 => 2 => 1

```
#include<iostream>
#include<vector>
using namespace std;
void Search(int v, vector < vector<int> >* G, int N)
{
int i;
for (i = 0; i < N; i++) {
if ((*G)[v][i])
{
(*G)[v][i] = (*G)[i][v] = 0;
Search(i, G, N);
}
}
cout << v + 1 << " => ";
}
int main()
{
int N = 0;
cin >> N;
vector < vector<int> > G(N, vector<int>(N));
for (int i = 0; i < N; ++i) {
for (int j = 0; j < N; ++j) {
cin >> G[i][j];
}
}
int count, p, q, sum;
count = 1;
for (p = 0; p < N; p++){
sum = 0;
for (q = 0; q < N; q++){
sum += G[p][q];
}
if (sum % 2) count = 0;
}
cout << endl;
if (count)
Search(0, &G, N);
else
cout << "again\n";
cout << endl;
return 0;
}
```

```
10
 0 1 1 0 0 0 1 0 0 1
 1 0 1 0 0 0 0 0 0 0
 1 1 0 1 1 1 1 0 0 0
 0 0 1 0 1 0 0 0 0 0
 0 0 1 1 0 1 0 1 0 0
 0 0 1 0 1 0 0 1 1 0
 1 0 1 0 0 0 0 0 1 1
 0 0 0 0 1 1 0 0 1 1
 0 0 0 0 0 1 1 1 0 1
 1 0 0 0 0 0 1 1 1 0

1 => 10 => 9 => 8 => 10 => 7 => 9 => 6 => 8 => 5 => 6 => 3 => 5 => 4 => 3 => 7 => 1 => 3 => 2 => 1

Process returned 0 (0x0)   execution time : 19.450 s
Press any key to continue.
```

Б)

```
1 2 3 7 1
2 1 7 3 2
3 1 10 7 3
4 3 6 5 4
5 3 6 8 5
6 3 4 5 6
7 1 2 3 7
8 5 3 6 8
9 6 3 7 9
10 1 3 7 10
```

```
#include <iostream>
#include <vector>
using namespace std;
vector<int> Vcon;
int Inf = 999;
bool check(vector<int> V, int pork) {
	for (auto i = V.begin(); i != V.end(); i++) {
		if (*i == pork)return false;
	}
	return true;
}
void Find(vector<int>* V, int** arr, int n, int pos, int start_pork) {
	for (int i = pos, k = 0; k < 1; i++, k++) {
		for (int j = 0; j < n; j++)
			if (arr[i][j] == 1 && check((*V), j)) {
				if (j == start_pork && (*V).size() > 2) {
					if (Inf > V->size()) {
						Vcon.clear();
						Vcon.push_back(start_pork + 1);
						for (auto it = (*V).begin(); it != (*V).end();
							it++)
							Vcon.push_back(*it + 1);
```

```
                                        Vcon.push_back(start_pork + 1);
                                        Inf = V->size();
                                        break;
                                }
                        }
                        else {
                                (*V).push_back(j);
                                Find(V, arr, n, j, start_pork);
                        }
                }
        }
        if (V->size() != 0)
                V->pop_back();
}
int main() {
        int n;
        cout << "Enter number of porks: ";
        cin >> n;
        int** arr = new int* [n];
        for (int i = 0; i < n; i++) {
                arr[i] = new int[n];
        }
        for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                        cin >> arr[i][j];
                }
        }
        vector<int> V;
        vector<int> WAS;
        cout << endl;
        int count, p, q, sum;
        count = 1;
        for (p = 0; p < n; p++)
        {
                sum = 0;
                for (q = 0; q < n; q++)
                {
                        sum += arr[p][q];
                }
                if (sum % 2) count = 0;
        }
        cout << endl;
        if (count) {
                for (int j = 0; j < n; j++) {
                        Inf = 999;
                        Find(&V, arr, n, j, j);
                        for (int i = 1; i <= Vcon.size(); i++) {
                                cout << Vcon[i - 1] << " ";
                        }
                        cout << endl;
                        Vcon.clear();
                }
}
        else
                cout << "not correct \n";
        cout << endl;
        return 0;
}
```

Завдання N9

Спростити формули (привести їх до скороченої ДНФ).

$$(X \vee \bar{Y} \vee \bar{Z})(\bar{X} \vee \bar{Z})$$

$$X\bar{X} \vee X\bar{Z} \vee \bar{Y}X \vee \bar{Y}\bar{Z} \vee \bar{Z}X \vee \bar{Z}\bar{Z}$$

$$0 \vee X\bar{Z} \vee \bar{Y}X \vee \bar{Y}\bar{Z} \vee \bar{Z}X \vee \bar{Z}\bar{Z}$$

$$X\bar{Z} \vee \bar{Y}X \vee \bar{Y}\bar{Z} \vee \bar{Z}X \vee \bar{Z}\bar{Z}$$

$$\bar{X}\bar{Y} \vee X\bar{Z} \vee \bar{Z}X \vee \bar{Z}\bar{Z}$$

$$\bar{X}\bar{Y} \vee \bar{Z} \vee \bar{Z}\bar{Z}$$

$$\bar{X}\bar{Y} \vee \bar{Z} \vee \bar{Z}$$

$$\bar{X}\bar{Y} \vee \bar{Z}$$