МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ
“ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Розрахунково-графічна робота

з дисципліни

«Дискретна математика»

Виконав:

студент групи КН-112

Стаськів Максим

Викладач:

Мельникова Н.І.

Львів – 2019р.

Варіант 14

Завдання №1

Виконати наступні операції над графами:

1) знайти доповнення до першого графу,

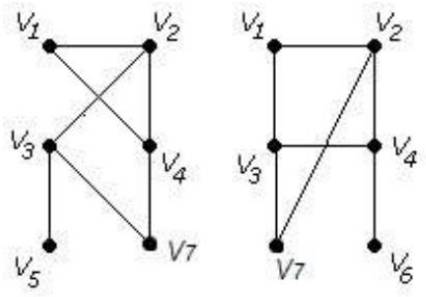
2) об’єднання графів,

3) кільцеву сумму G1 та G2 (G1+G2),

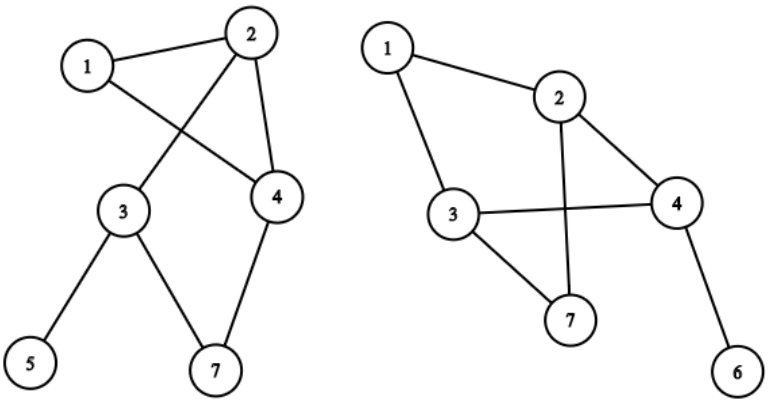
4) розмножити вершину у другому графі,

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

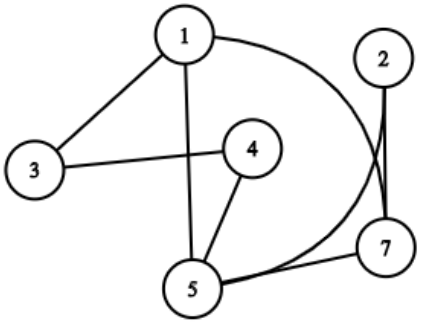
6) добуток графів.



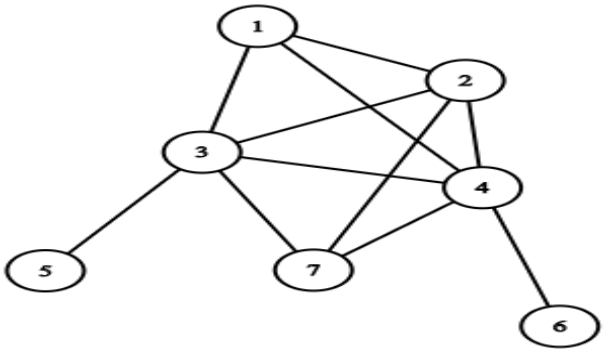
G1: G2:



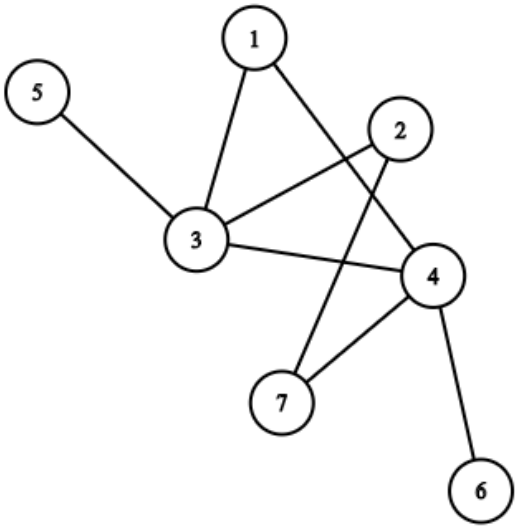
1) Доповнення до першого графу:



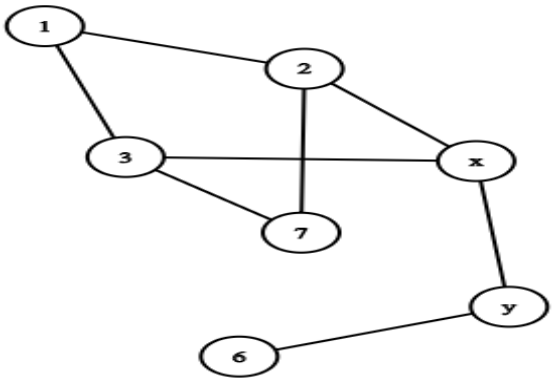
2) Об’єднання графів:



3) Кільцева сума G1 та G2 (G1+G2):

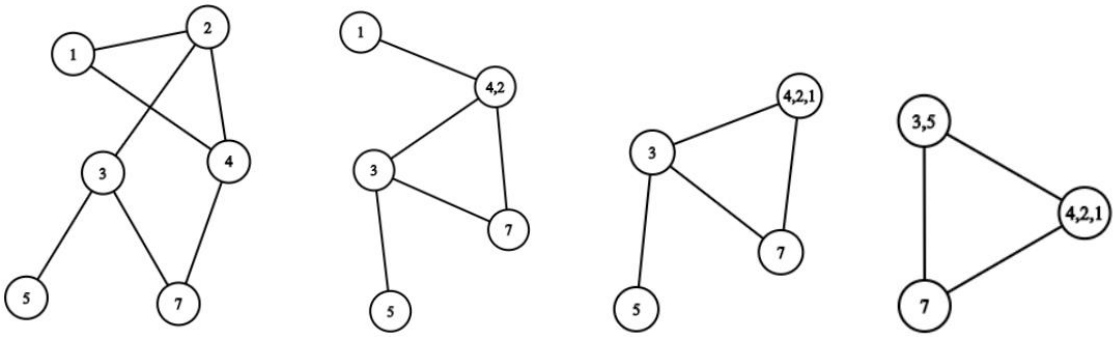


4) Розмноження вершини у другому графі:

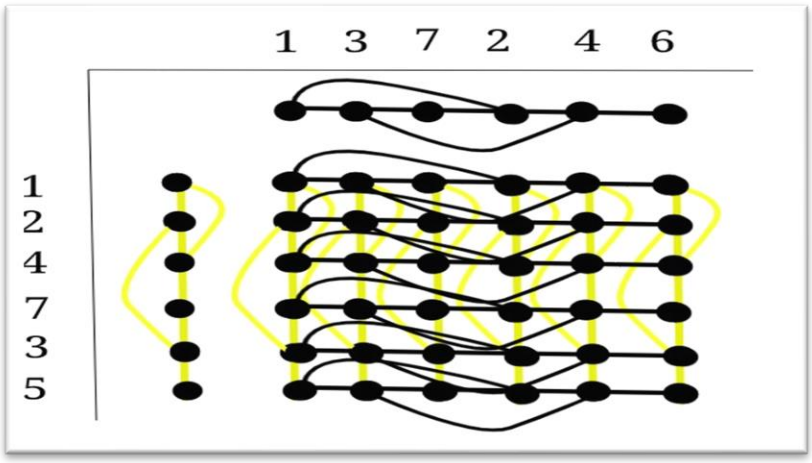


5) Виділення підграфа А - що скадається з 3-х вершин в G1:

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

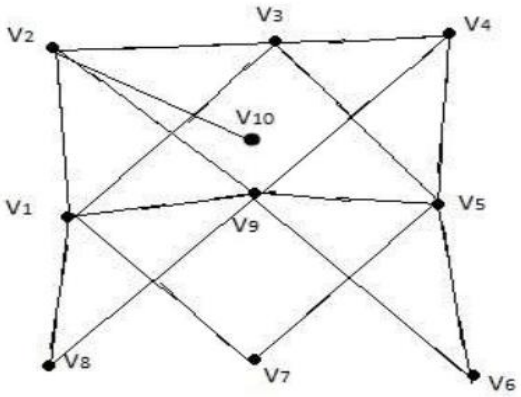


6) добуток графів.



Завдання № 2

Скласти таблицю суміжності для орграфа.



Таблиця Суміжності

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

Завдання № 3

Для графа з другого завдання знайти діаметр.

Таблиця Ваг

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

Діаметр = 3.

Завдання № 4

Для графа з другого завдання виконати обхід дерева вглиб (варіант закінчується на непарне число) або вшир (закінчується на парне число).

Вершина	BFS-номер	Вміст черги
V_{10}	1	V_{10}
V_2	2	V_{10} V_2
-	-	V_2
V_3	3	V_2 V_3
V_1	4	V_2 V_3 V_1
V_9	5	V_2 V_3 V_1 V_9
-	-	V_3 V_1 V_9
V_4	6	V_3 V_1 V_9 V_4
V_5	7	V_3 V_1 V_9 V_4 V_5
-	-	V_1 V_9 V_4 V_5
V_8	8	V_1 V_9 V_4 V_5 V_8
V_7	9	V_1 V_9 V_4 V_5 V_8 V_7
-	-	V_9 V_4 V_5 V_8 V_7
V_6	10	V_9 V_4 V_5 V_8 V_7 V_6
-	-	V_4 V_5 V_8 V_7 V_6
-	-	V_5 V_8 V_7 V_6
-	-	V_8 V_7 V_6
-	-	V_7 V_6
-	-	V_6
-	-	Ø

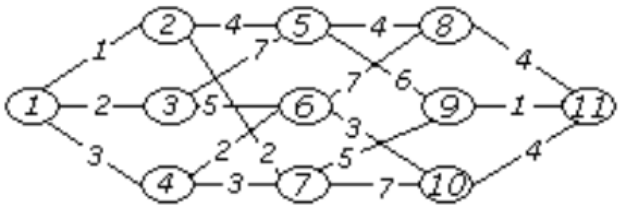
*main.cpp

```cpp
#include <iostream>
#include<queue>
#define v 10
using namespace std;

class n {
public:
    int val;
    int st;};

int matrix[v][v] = {
{0, 1,  1,  0,  0,  0,  1,  1,  1,  0},
{1, 0,  1,  0,  0,  0,  0,  0,  1,  1},
{1, 1,  0,  1,  1,  0,  0,  0,  0,  0},
{0, 0,  1,  0,  1,  0,  0,  0,  1,  0},
{0, 0,  1,  1,  0,  1,  1,  0,  1,  0},
{0, 0,  0,  0,  1,  0,  0,  0,  1,  0},
{1, 0,  0,  0,  1,  0,  0,  0,  0,  0},
{1, 0,  0,  0,  0,  0,  0,  0,  1,  0},
{1, 1,  0,  1,  1,  1,  0,  1,  0,  0},
{0, 1,  0,  0,  0,  0,  0,  0,  0,  0}
};

void bfs(n* verh, n s){
    n u;
    int i, j;
    queue<n> que;
    for(i=0; i<v; i++){
        verh[i].st=0;
    }
    verh[s.val].st=1;
    que.push(s);
    while(!que.empty()){
        u=que.front();
        que.pop();
        cout<<u.val+1<<"";
        for(i=0; i<v; i++){
            if(matrix[i][u.val]){
                if(verh[i].st==0){
                    verh[i].st=1;
                    que.push(verh[i]);
                }
            }
        }
        u.st=2;}
}
int main(){
    n verh[v];
    n start;
    int s;
    for (int i=0;i<v;i++){
        verh[i].val=i;
    }
    s=65;
    start.val=s-65;
    cout<<"BFS:";
    bfs(verh, start);
    cout<< endl;
}
```

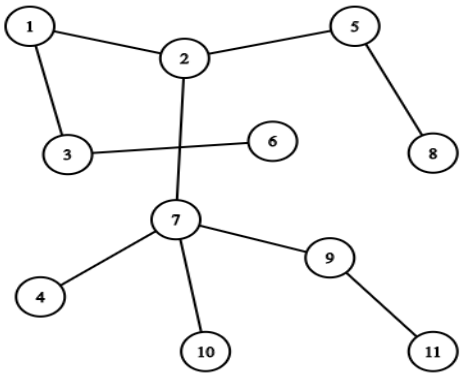
```
BFS:12378910456

Process returned 0 (0x0)   execution time : 0.108 s
Press any key to continue.
```

Завдання № 5

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



Краскал:



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

E={(1,2), (9,11),(1,3),(2,7),(2,5),(7,4),(3.6),(5,8),(7,9),(7, 10).

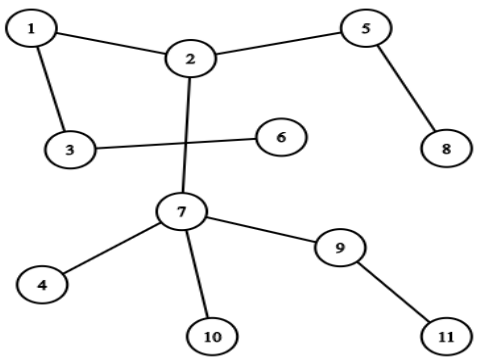
*main.cpp

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

class Edge
{
    public:
    int src, dest, weight;
};

class Graph
{
    public:
    int V, E;
    Edge* edge;
};

Graph* createGraph(int V, int E)
{
    Graph* graph = new Graph;
    graph->V = V;
    graph->E = E;
    graph->edge = new Edge[E];
    return graph;
}


class subset
{   public:
    int parent;
    int rank;
};


int find(subset subsets[], int i)
{
    if (subsets[i].parent != i)
    subsets[i].parent = find(subsets, subsets[i].parent);
    return subsets[i].parent;
}

void Union(subset subsets[], int x, int y)
{
    int xroot = find(subsets, x);
    int yroot = find(subsets, y);


    if (subsets[xroot].rank < subsets[yroot].rank)
        subsets[xroot].parent = yroot;
    else if (subsets[xroot].rank > subsets[yroot].rank)
        subsets[yroot].parent = xroot;
    else
    {
        subsets[yroot].parent = xroot;
        subsets[xroot].rank++;
    }
}
int myComp(const void* a, const void* b)
{
    Edge* a1 = (Edge*)a;
    Edge* b1 = (Edge*)b;
    return a1->weight > b1->weight;
}

void Kruskal(Graph* graph)
{
    int V = graph->V;
    Edge result[V];
    int e = 0;
    int i = 0;

    qsort(graph->edge, graph->E, sizeof(graph->edge[0]), myComp);
    subset *subsets = new subset[( V * sizeof(subset) )];

    for (int v = 0; v < V; ++v)
    {   subsets[v].parent = v;
        subsets[v].rank = 0;}

    while (e < V - 1 && i < graph->E)
    {
        Edge next_edge = graph->edge[i++];

        int x = find(subsets, next_edge.src);
```

```cpp
    for (int v = 0; v < V; ++v)
    {   subsets[v].parent = v;
        subsets[v].rank = 0;}

    while (e < V - 1 && i < graph->E)
    {
        Edge next_edge = graph->edge[i++];

        int x = find(subsets, next_edge.src);
        int y = find(subsets, next_edge.dest);

        if (x != y)
        {   result[e++] = next_edge;
            Union(subsets, x, y);}
    }
    cout<<"Following are the edges in the constructed Kruskal graph: \n";
    for (i = 0; i < e; ++i)
        cout<<result[i].src<<" -- "<<result[i].dest<<" == "<<result[i].weight<<endl;
    return;
}

int main()
{
    int V;
    int E;
    cout<<"Enter number of Vershun:";
    cin>>V;
    cout<<"Enter number of Rebra";
    cin>>E;
    Graph* graph = createGraph(V, E);

    for(int k=0; k<18; k ++){
    cout<<"Enter two Vershuns:";
    cin>>graph->edge[k].src ;
    cin>>graph->edge[k].dest ;
    cout<<"Enter weight of edge:";
    cin>>graph->edge[k].weight;}

    Kruskal(graph);
    return 0;
}
```

```
Enter weight of edge:4
Enter two Vershuna:10 7
Enter weight of edge:4
Enter two Vershuna:7 4
Enter weight of edge:3
Enter two Vershuna:3 1
Enter weight of edge:4
Enter two Vershuna:1 3
Enter weight of edge:2
Enter two Vershuna:3 6
Enter weight of edge:6
Enter two Vershuna:11 9
Enter weight of edge:1
Enter two Vershuna:2 7
Enter weight of edge:2
Enter two Vershuna:3 5
Enter weight of edge:7
Enter two Vershuna:6 4
Enter weight of edge:2
Enter two Vershuna:7 9
Enter weight of edge:5
Enter two Vershuna:6 10
Enter weight of edge:3
Enter two Vershuna:6 8
Enter weight of edge:7
Enter two Vershuna:9 5
Enter weight of edge:5
Following are the edges in the constructed MST
1 -- 2 == 1
2 -- 5 == 3
6 -- 10 == 3
8 -- 11 == 4
11 -- 10 == 4
10 -- 7 == 4
7 -- 4 == 3
3 -- 1 == 4
9 -- 5 == 5
11 -- 9 == 1
```

Прима:



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

E={(1,2),(1,3),(2,7),(2,5),(7,4),(3.6),(5,8),(7,9),(9,11),(7, 10).

*main.cpp

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

class Edge
{
    public:
    int src, dest, weight;
};

class Graph
{
    public:
    int V, E;
    Edge* edge;
};

Graph* createGraph(int V, int E)
{
    Graph* graph = new Graph;
    graph->V = V;
    graph->E = E;
    graph->edge = new Edge[E];
    return graph;
}


class subset
{   public:
    int parent;
    int rank;
};


int find(subset subsets[], int i)
{
    if (subsets[i].parent != i)
    subsets[i].parent = find(subsets, subsets[i].parent);
    return subsets[i].parent;
}

void Union(subset subsets[], int x, int y)
{
    int xroot = find(subsets, x);
    int yroot = find(subsets, y);


    if (subsets[xroot].rank < subsets[yroot].rank)
        subsets[xroot].parent = yroot;
    else if (subsets[xroot].rank > subsets[yroot].rank)
        subsets[yroot].parent = xroot;
    else
    {
        subsets[yroot].parent = xroot;
        subsets[xroot].rank++;
    }
}

int myComp(const void* a, const void* b)
{
    Edge* a1 = (Edge*)a;
    Edge* b1 = (Edge*)b;
    return a1->weight > b1->weight;
}

void Kruskal(Graph* graph)
{
    int V = graph->V;
    Edge result[V];
    int e = 0;
    int i = 0;

    qsort(graph->edge, graph->E, sizeof(graph->edge[0]), myComp);
    subset *subsets = new subset[( V * sizeof(subset) )];

    for (int v = 0; v < V; ++v)
    {   subsets[v].parent = v;
        subsets[v].rank = 0;}

    while (e < V - 1 && i < graph->E)
    {
        Edge next_edge = graph->edge[i++];

        int x = find(subsets, next_edge.src);
```

```
    for (int v = 0; v < V; ++v)
    {   subsets[v].parent = v;
        subsets[v].rank = 0;}

    while (e < V - 1 && i < graph->E)
    {
        Edge next_edge = graph->edge[i++];

        int x = find(subsets, next_edge.src);
        int y = find(subsets, next_edge.dest);

        if (x != y)
        {   result[e++] = next_edge;
            Union(subsets, x, y);}
    }
    cout<<"Following are the edges in the constructed Kruskal graph: \n";
    for (i = 0; i < e; ++i)
        cout<<result[i].src<<" -- "<<result[i].dest<<" == "<<result[i].weight<<endl;
    return;
}

int main()
{
    int V;
    int E;
    cout<<"Enter number of Vershun:";
    cin>>V;
    cout<<"Enter number of Reber";
    cin>>E;
    Graph* graph = createGraph(V, E);

    for(int k=0; k<10; k ++){
    cout<<"Enter two Vershuna:";
    cin>>graph->edge[k].src ;
    cin>>graph->edge[k].dest ;
    cout<<"Enter weight of edge:";
    cin>>graph->edge[k].weight;}

    Kruskal(graph);
    return 0;
}
```

```
Enter weight of edge:4
Enter two Vershuna:10 7
Enter weight of edge:4
Enter two Vershuna:7 4
Enter weight of edge:3
Enter two Vershuna:3 1
Enter weight of edge:4
Enter two Vershuna:1 3
Enter weight of edge:2
Enter two Vershuna:3 6
Enter weight of edge:6
Enter two Vershuna:11 9
Enter weight of edge:1
Enter two Vershuna:2 7
Enter weight of edge:2
Enter two Vershuna:3 5
Enter weight of edge:7
Enter two Vershuna:6 4
Enter weight of edge:2
Enter two Vershuna:7 9
Enter weight of edge:5
Enter two Vershuna:6 10
Enter weight of edge:3
Enter two Vershuna:6 8
Enter weight of edge:7
Enter two Vershuna:9 5
Enter weight of edge:5
Following are the edges in the constructed MST
1 -- 2 == 1
2 -- 5 == 3
6 -- 10 == 3
8 -- 11 == 4
11 -- 10 == 4
10 -- 7 == 4
7 -- 4 == 3
3 -- 1 == 4
9 -- 5 == 5
11 -- 9 == 1
```

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

Вершина	Найближча до неї	Довжина шляху
1(початкова)	2	1
1-2	4	1+1
1-2-4	8	1+1+1
1-2-4-8	7	1+1+1+1
1-2-4-8-7	3	1+1+1+1+1
1-2-4-8-7-3	6	1+1+1+1+1+4
1-2-4-8-7-3-6	5	1+1+1+1+1+4+1
1-2-4-8-7-3-6-5	1	1+1+1+1+1+4+1+1
1-2-4-8-7-3-6-5-1	=	11

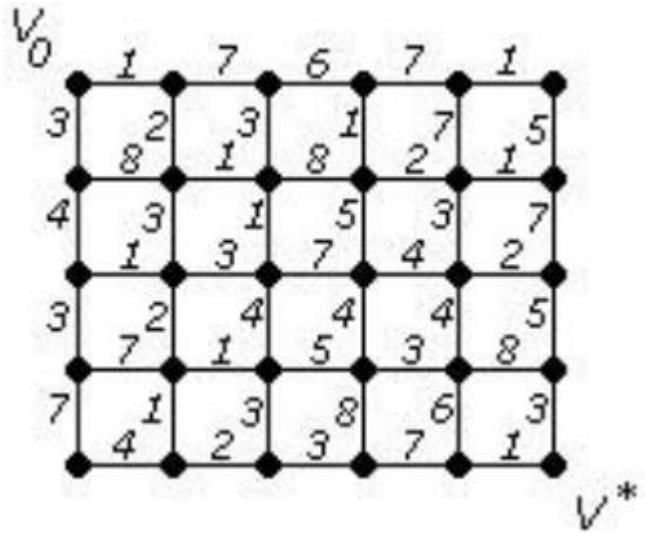
main.cpp

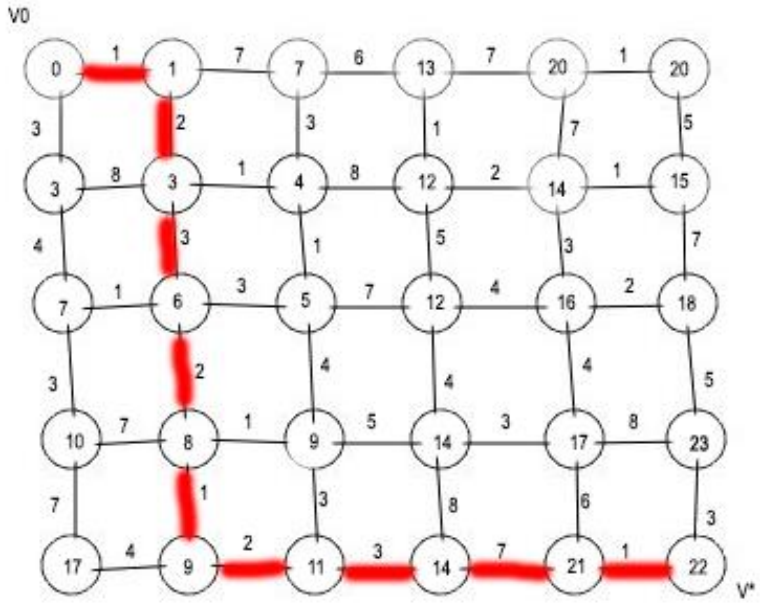
```cpp
#include <bits/stdc++.h>
using namespace std;
#define vr 8
int TSP(int grph[][vr], int p){
    vector<int> ver;
    for (int i = 0; i < vr; i++)
        if (i != p)
            ver.push_back(i);
            int m_p = INT_MAX;
    do {
        int cur_pth = 0;
        int k = p;
        for (int 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, 1, 1, 1, 1, 1, 3, 1},
        {1, 0, 5, 1, 2, 1, 3, 3},
        {1, 5, 0, 2, 5, 4, 1, 5},
        {1, 1, 2, 0, 5, 5, 6, 2},
        {1, 2, 5, 5, 0, 1, 5, 1},
        {1, 1, 4, 5, 1, 0, 5, 6},
        {3, 3, 1, 6, 5, 5, 0, 1},
        {1, 3, 5, 1, 1, 6, 1, 0}
    };
    int p = 0;
    cout<< "\n The result is: "<< TSP(grph, p) << endl;
    return 0;
}
```

```
The result is: 11

Process returned 0 (0x0)   execution time : 0.108 s
Press any key to continue.
```

Завдання № 7

За допомогою алгоритму Дейкстри знайти найкоротший шлях у графі між парою вершин V0 і V *.





Найкоротша відстань від Vo до V* = 22

```cpp
int main()
{
    int inf = 100000;
    int a, b, c;
    int v = 0;
    cout << "Number of Vershunas  : ";
    cin >> v;
    int** graph = new int* [v];
    for (int j = 0; j < v; j++)
    {
        graph[j] = new int[v];
    }
    for (int a = 0; a < v; a++)
    {
        for (int j = 0; j < v; j++)
        {
            graph[a][j] = 0;
        }
    }
    int r = 0;
    cout << "Enter number of Reber : ";
    cin >> r;
    cout << "Enter weight of rebex : " << endl;
    for (size_t i = 0; i < r; i++)
    {
        cin >> a;
        cin >> b;
        cin >> c;
        graph[a - 1][b - 1] = graph[b - 1][a - 1] = c;
    }
```

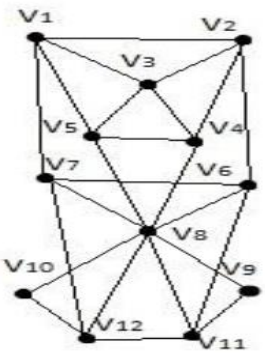
```cpp
    int p;
    int** tops = new int* [v];
    for (int j = 0; j < v; j++)
    {
        tops[j] = new int[2];
    }
    int* tops_path = new int[v];
    cout << "Begin at: ";
    cin >> p;
    for (int i = 0; i < v; i++)
    {
        if (i == p - 1) {
            tops[i][0] = 0;
            tops[i][1] = 1;
        }
        else {
            tops[i][0] = inf;
            tops[i][1] = 1;
        }
    }
```

```
    tops_path[p - 1] = 0;
    int m;
    for (int i = 0; i < v; i++)
    {
        m = min_top(tops, v);
        for (int j = 0; j < v; j++)
        {
            if (graph[m][j])
            {
                if (tops[j][0] > tops[m][0] + (graph[m][j]))
                {
                    tops[j][0] = tops[m][0] + (graph[m][j]);
                    tops_path[j] = m;
                }
            }
        }
        tops[m][1] = 0;
    }
    ////max
    cout << "End at: ";
    int k;
    cin >> k;
    cout << "Shortest route: ";
    cout << tops[k - 1][0];
    cout << endl << k << " <-- ";
    k--;
    for (int a = 0; tops_path[k] != p - 1; a++)
    {
        cout << tops_path[k] + 1 << " <-- ";
        k = tops_path[k];

    }
    cout << p << endl;
    return 0;
}
```

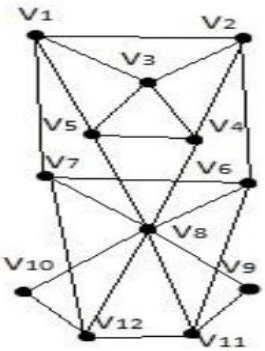
```
15 21 7
16 17 5
16 22 8
17 18 7
17 23 2
18 24 5
19 20 7
19 25 3
20 21 3
20 26 2
21 22 1
21 27 1
22 23 1
22 28 3
23 24 8
23 29 8
24 30 7
25 26 4
26 27 7
27 28 3
28 29 3
29 30 6
Begin at: 1
End at: 30
Shortest route: 24
30 <-- 29 <-- 28 <-- 27 <-- 21 <-- 15 <-- 9 <-- 3 <-- 2 <-- 1

Process returned 0 (0x0)   execution time : 15.453 s
Press any key to continue.
```

Завдання № 8

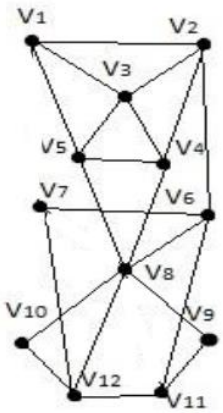
Знайти ейлеровий цикл в ейлеровому графі двома методами: а) Флері; б) елементарних циклів.



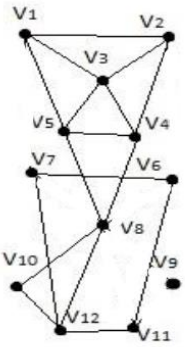
а)



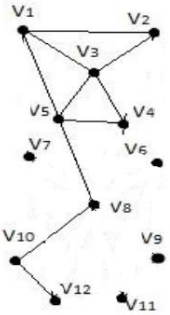
1) 1 – 7 – 8 – 11



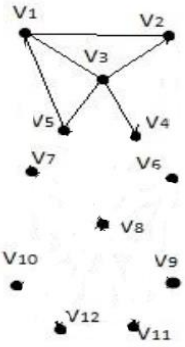
2) 11 – 9 – 8 – 6 – 2



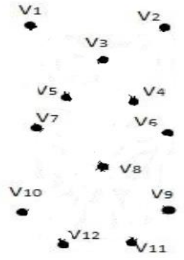
3) 2 – 4 – 8 – 12 – 11 – 6 – 7 – 12



4) 12 – 10 – 8 – 5 – 4



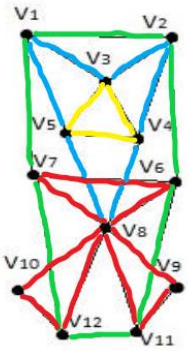
5) 1 – 7 – 8 – 11 – 9 – 8 – 6 – 2 – 4 – 8 – 12 – 11 – 6 – 7 – 12 – 10 – 8 – 5 – 4 – 3 – 2 – 1 – 3 – 5 – 1



```cpp
#include<iostream>
#include<vector>
#define NODE 12
using namespace std;
int graph[NODE][NODE] = {
   {0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0},
   {1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0},
   {1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0},
   {0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0},
   {1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0},
   {0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0},
   {1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1},
   {0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1},
   {0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0},
   {0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1},
   {0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1},
   {0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0}
};
int tempGraph[NODE][NODE];
int findStartVert(){
   for(int i = 0; i<NODE; i++){
      int deg = 0;
      for(int j = 0; j<NODE; j++){
         if(tempGraph[i][j])
         deg++;
      }
      if(deg % 2 != 0)
      return i;
   }
   return 0;
}
bool isBridge(int u, int v){
   int deg = 0;
   for(int i = 0; i<NODE; i++)
      if(tempGraph[v][i])
         deg++;
      if(deg>1){
         return false;
      }
   return true;
}
int edgeCount(){
   int count = 0;
   for(int i = 0; i<NODE; i++)
      for(int j = i; j<NODE; j++)
         if(tempGraph[i][j])
            count++;
   return count;
}
void fleuryAlgorithm(int start){
   static int edge = edgeCount();
   for(int v = 0; v<NODE; v++){
      if(tempGraph[start][v]){
         if(edge <= 1 || !isBridge(start, v)){
            cout << start << "--" << v << " ";
            tempGraph[start][v] = tempGraph[v][start] = 0;
            edge--;
            fleuryAlgorithm(v);
         }
      }
   }
}
int main(){
   for(int i = 0; i<NODE; i++)
   for(int j = 0; j<NODE; j++)
   tempGraph[i][j] = graph[i][j];
   cout << "Fleury Path Or Circuit: ";
   fleuryAlgorithm(findStartVert());
}
```

```
Fleury Path Or Circuit: 0 -- 6 -- 7 -- 10 -- 8 -- 7 -- 5 -- 1 -- 3 -- 7 -- 11 -- 10 -- 5 -- 6 -- 11 -- 9 -- 7 -- 4 -- 3
-- 2 -- 1 -- 0 -- 2 -- 4 -- 0
Process returned 0 (0x0)   execution time : 0.096 s
Press any key to continue.
```

б)



Цикли :

1)1-2-6-11-12-7-1

2) 7-6-8-10-12-8-11-9-8-7

3)8-5-1-3-2-4-8

4)3-5-4-3

Ейлерів цикл:

1-5-3-4-5-8-10-12-8-11-9-8-4-2-6-11-12-7-6-8-7-1-3-2-1

main.cpp

```cpp
#include<iostream>
#include<vector>
#define NODE 12
using namespace std;
int graph[NODE][NODE] = {
   {0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0},
   {1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0},
   {1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0},
   {0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0},
   {1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0},
   {0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0},
   {1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1},
   {0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1},
   {0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0},
   {0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1},
   {0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1},
   {0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0}
};
int tempGraph[NODE][NODE];
int findStartVert(){
   for(int i = 0; i<NODE; i++){
      int deg = 0;
      for(int j = 0; j<NODE; j++){
         if(tempGraph[i][j])
         deg++;
      }
      if(deg % 2 != 0)
      return i;
   }
   return 0;
```

```
}
bool isBridge(int u, int v){
   int deg = 0;
   for(int i = 0; i<NODE; i++)
      if(tempGraph[v][i])
         deg++;
      if(deg>1){
         return false;
      }
   return true;
}
int edgeCount(){
   int count = 0;
   for(int i = 0; i<NODE; i++)
      for(int j = i; j<NODE; j++)
         if(tempGraph[i][j])
            count++;
   return count;
}
void fleuryAlgorithm(int start){
   static int edge = edgeCount();
   for(int v = 0; v<NODE; v++){
      if(tempGraph[start][v]){
         if(edge <= 1 || !isBridge(start, v)){
            cout << start << "--" << v << " ";
            tempGraph[start][v] = tempGraph[v][start] = 0;
            edge--;
            fleuryAlgorithm(v);
         }
      }
   }
}
int main(){
   for(int i = 0; i<NODE; i++)
   for(int j = 0; j<NODE; j++)
   tempGraph[i][j] = graph[i][j];
   cout << "Euler Path Or Circuit: ";
   fleuryAlgorithm(findStartVert());
}
```

```
Euler Path Or Circuit: 0--1 1--3 3--2 2--0 0--4 4--1 1--5 5--6 6--7 7--2 2--4 4--3 3--7 7--5 5--10 10--7 7--8 8--10 10--
11 11--6 11--7 7--9
Process returned 0 (0x0)   execution time : 0.116 s
Press any key to continue.
```

Завдання №9

Спростити формули (привести їх до скороченої ДНФ)

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

Оскільки наша функція відповідає цим вимогам:

1) будь-які два доданки відрізняються як мінімум в двох позиціях,

2) жоден з Кон'юнктів не міститься в іншому.

можемо стверджувати що ця функція вже є скороченою ДНФ, і спростити її не можна.