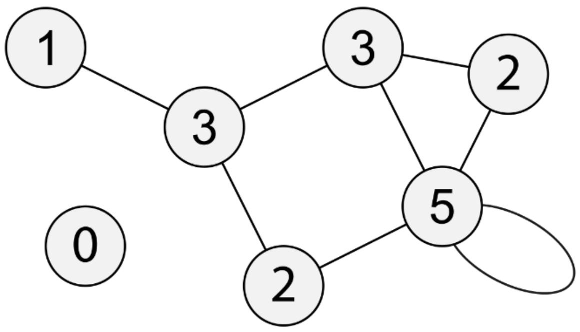
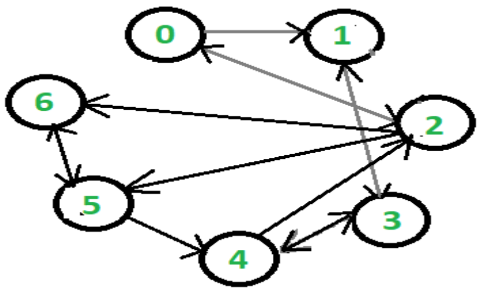
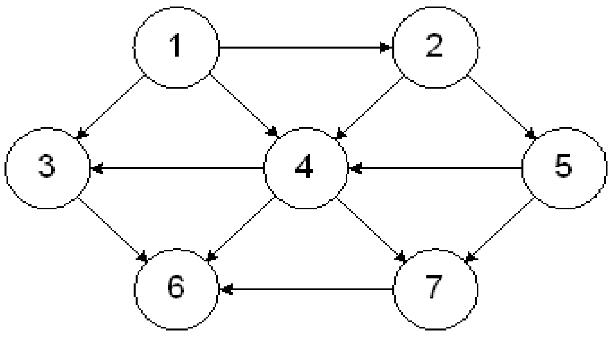
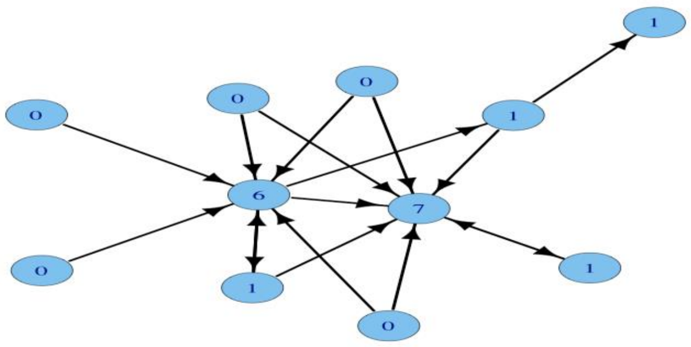
Mavzu: Grafni mashina xotirasida ifodalash usullari: tomonlar ketma-ketligi, uchlar qo`shniligi massivi orqali, uchlar qo`shniligi ro`yhat orqali, qo`shnilik matrisalar orqali. Grafda o`tish muammolari. BFS va DFS algoritmlari

1. Quydagi graflarni tahlil qiling. (Kalit so’zlar Graf turi, qirrasi, ildizi, darajasi, hajmi)

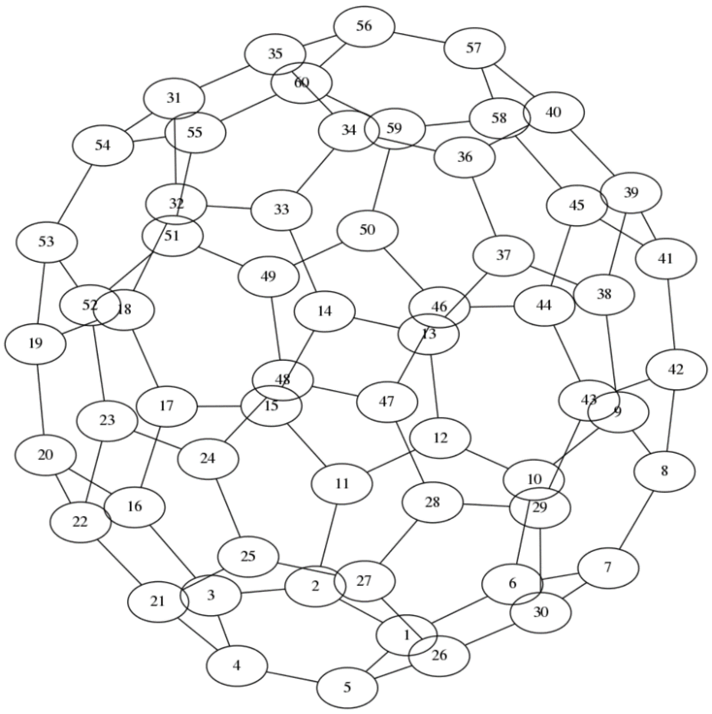


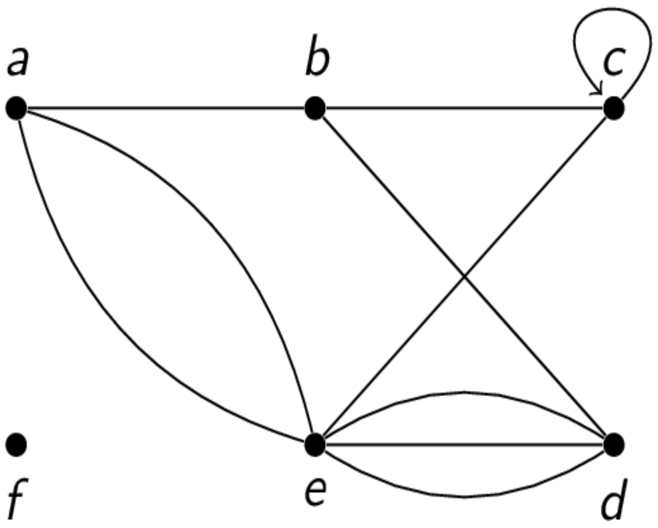






0
0
0
0
0
0
6
7
1
1
1
1
1

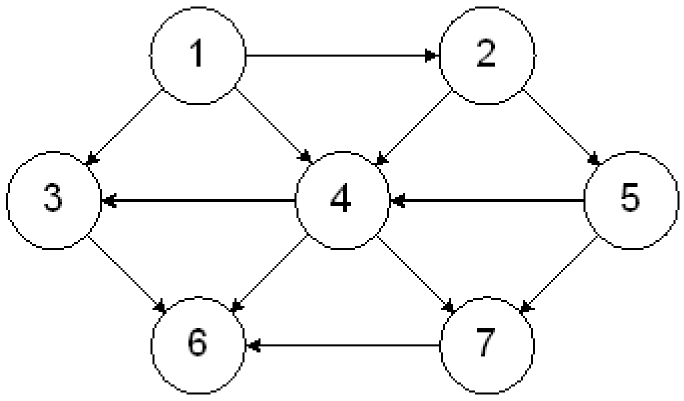


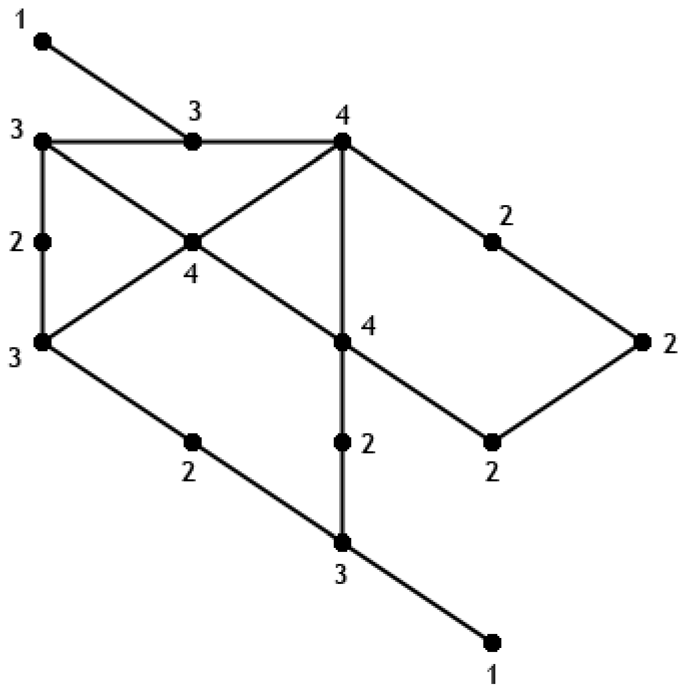


a
b
c
f
e
d

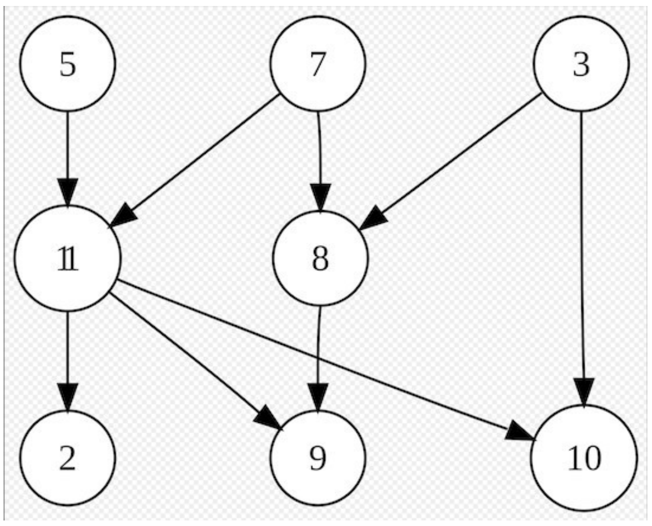
2. 6.ta uchdan iborat orentrlanmagan, to’la garf hosil qiling va uning hajmini toping?
3. Regulyar grafda darajasi 5 va n ta tugundan iborat grafni chizing va uning

   hajmini toping ?
4. Quyidagi grafda qo’shnilik matritsasi va insendentlik matritsasini hosil qiling .

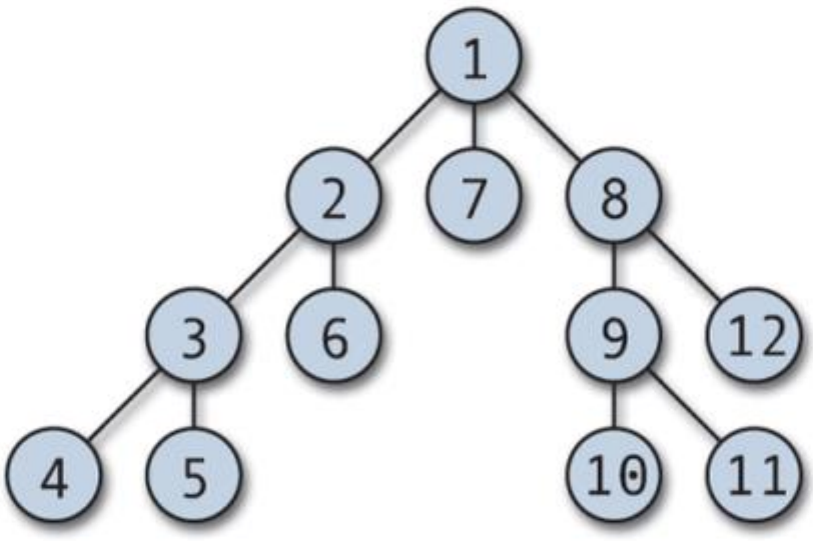




5. Quyidagi grafda qo’shnilik matritsasi va insendentlik matritsasini hosil qiling



6. Quyidagi daraxt elementlari tahlil qiling

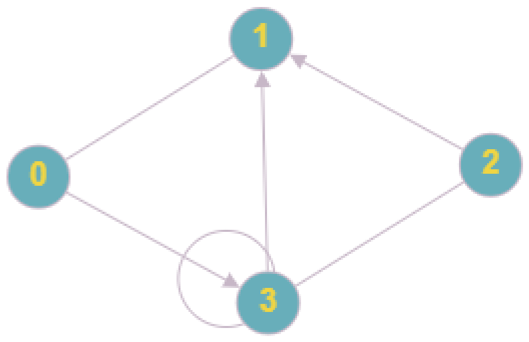


7.Shunday orgraf hosil qiling unda 5 ta tugundan iborat bo’lsin, maximum va minimum darajalar yig’indisini toping.
8. 7 bargdan iborat daraxt hosil qiling. Uning elementlarini tahlil qiling
9. Muvazanatlashgan daraxt hosil qiling va uning hajmini aniqlang.
10. Muvazanatlashgan daraxt hosil qiling va uning balandligini aniqlang.
11. Shundan graf hosil qilingki, unda barcha barglarning balandligi bir xil bo’lsin.
12. 12 ta qirradan iborat graf hosil qiling , uni tugunlari sonini va chiqivchi darajasi 0 bo’lgan tugunlar sonini aniqlang.
13. Bo’g’langan, bog’lanmagan graflarni hosil qiling va tahlil qiling
14. Asikl graf hosil qiling va kiriruvchisi 0 bo’lgan tugunni aniqlang.
15. Yo’naltirilmagan va orgraf hosil qiling.
16. Daraxtlarga oid dastur tuzing
17. Graflarga oid dastur tuzing.

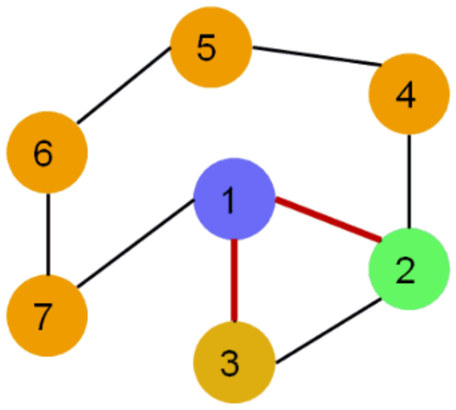
18. Quyida berilgan graflar bo’yicha berilgan vazifalarni hal qiling

a) Berilgan grafning barcha tushunchalarini keltiring (uchlar, qirralar, grafning yo’nalishga ega yoki yo’qligi bo’yicha aniqlanishi, yakkalangan uch mavjudligi, regular yoki regular emasligi va hokazo).
b) Grafni mashina xotirasida tasvirlash
c) 2.1) Grafni uchlar qo’shniligi matritsasi orqali tasvirlang
d) 2.2) Grafni qo’shnilik ro’yxati orqali tasvirlash
e) 2.3) Grafning insidentlik matritsasi orqali tasvirlash

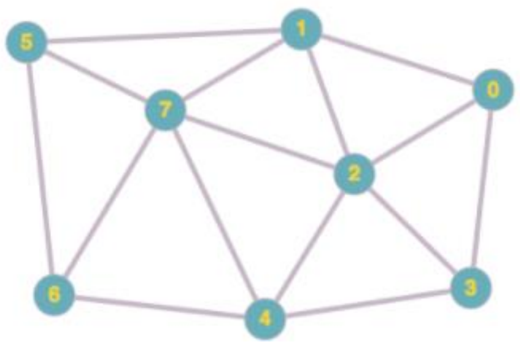
a)

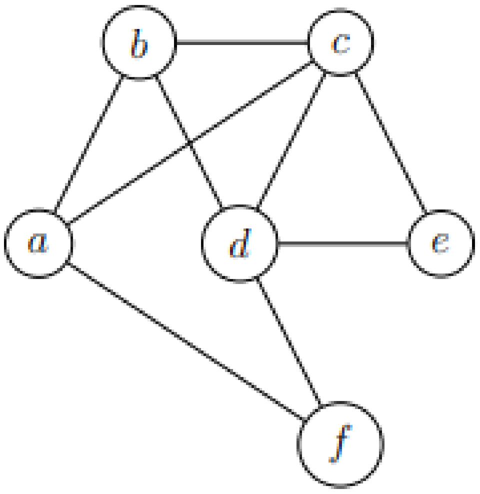


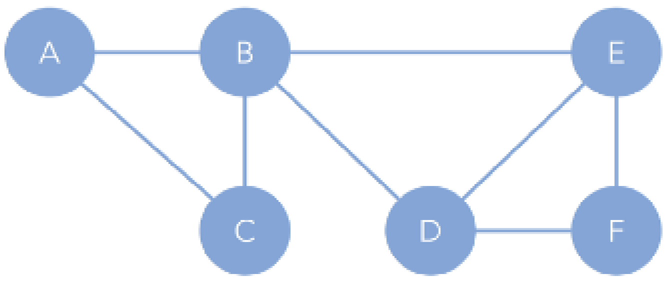
b)

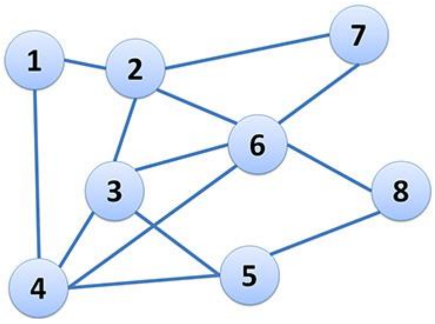


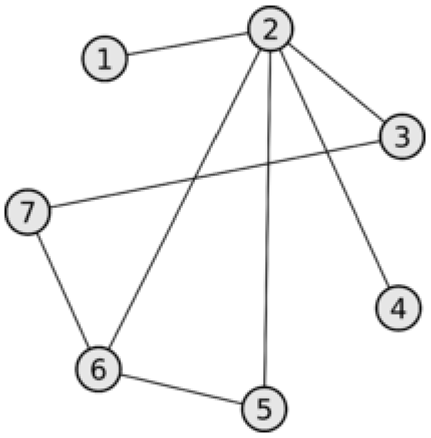
c)







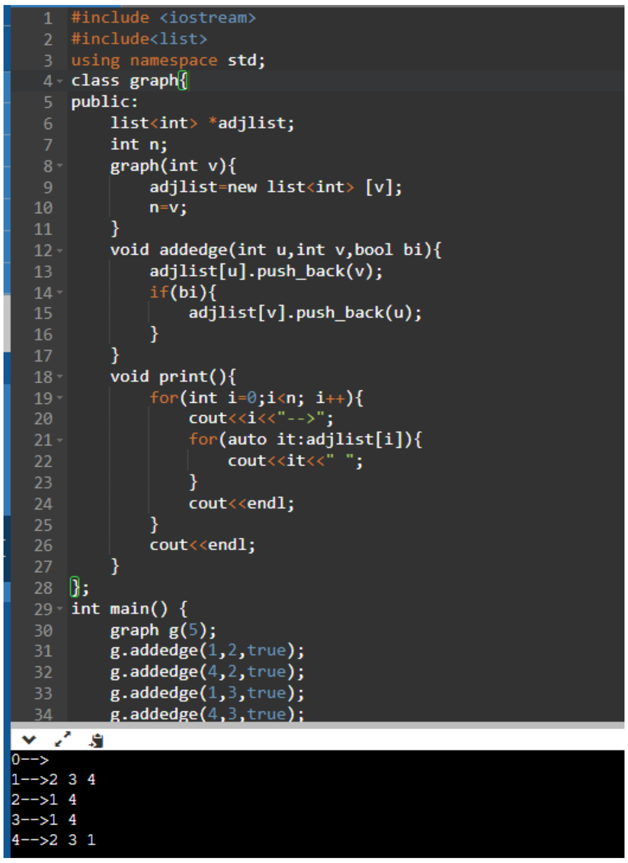




19.Quyidagi keltirilgan dasturni tahlil qiling.

```
 #include <iostream>
#include<list>
using namespace std;
class graph{
public:
    list<int> *adjlist;
    int n;
    graph(int v){
        adjlist=new list<int> [v];
        n=v;
    }
    void addedge(int u,int v,bool bi){
        adjlist[u].push_back(v);
        if(bi){
            adjlist[v].push_back(u);
        }
    }
    void print(){
        for(int i=0;i<n; i++){
            cout<<i<<"-->";
            for(auto it:adjlist[i]){
                cout<<it<<" ";
            }
            cout<<endl;
        }
        cout<<endl;
    }
};
int main() {
    graph g(5);
    g.addedge(1,2,true);
    g.addedge(4,2,true);
    g.addedge(1,3,true);
    g.addedge(4,3,true);
    g.addedge(1,4,true);
    g.print();
}
```

20.Quyidagi dastur kodini tahlil qiling.



```
#include <iostream>
#include<list>
using namespace std;
class graph{
public:
    list<int> *adjlist;
    int n;
    graph(int v){
        adjlist=new list<int> [v];
        n=v;
    }
    void addedge(int u,int v,bool bi){
        adjlist[u].push_back(v);
        if(bi){
            adjlist[v].push_back(u);
        }
    }
    void print(){
        for(int i=0;i<n; i++){
            cout<<i<<"-->";
            for(auto it:adjlist[i]){
                cout<<it<<" ";
            }
            cout<<endl;
        }
        cout<<endl;
    }
};
int main() {
    graph g(5);
    g.addedge(1,2,true);
    g.addedge(4,2,true);
    g.addedge(1,3,true);
    g.addedge(4,3,true);
```

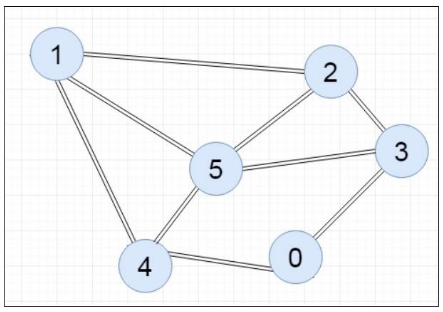
```
0-->
1-->2 3 4
2-->1 4
3-->1 4
4-->2 3 1
```

21. Quyidagi kodni tahlil qiling va natijasini yozing.

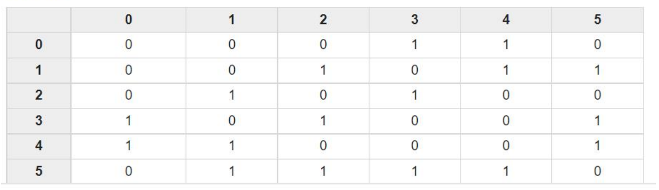
```
 #include<iostream>
using namespace std;
int inc_arr[20][20]; //initial array to hold incidence matrix
int ed_no = 0;
void displayMatrix(int v, int e) {
   int i, j;
   for(i = 0; i < v; i++) {
      for(j = 0; j < e; j++) {
         cout << inc_arr[i][j] << " ";  }
      cout << endl;
   }
}
void add_edge(int u, int v) { //function to add edge into the matrix with edge number
   inc_arr[u][ed_no] = 1;
   inc_arr[v][ed_no] = 1;
   ed_no++; //increase the edge number
}
main(int argc, char* argv[]) {
   int v = 6; //there are 6 vertices in the graph
   int e = 9; //there are 9 edges in the graph
   add_edge(0, 4);
   add_edge(0, 3);
   add_edge(1, 2);
   add_edge(1, 4);
   add_edge(1, 5);
   add_edge(2, 3);
   add_edge(2, 5);
   add_edge(5, 3);
   add_edge(5, 4);
   displayMatrix(v, e);
}
```

22.Quyidagi grafni va matritsani tahliling.

Input:

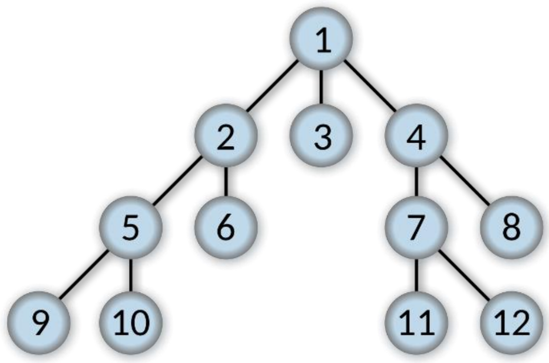


Output:

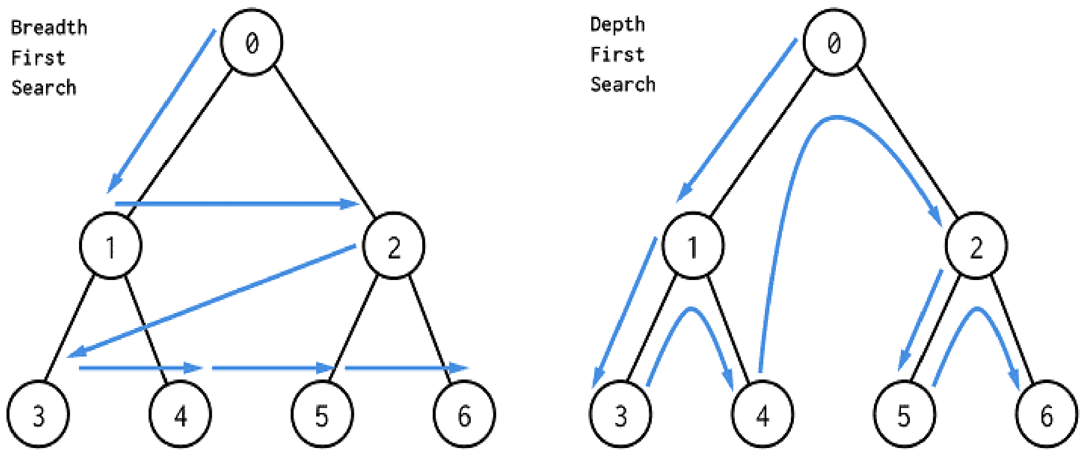


	0	1	2	3	4	5
0	0	0	0	1	1	0
1	0	0	1	0	1	1
2	0	1	0	1	0	0
3	1	0	1	0	0	1
4	1	1	0	0	0	1
5	0	1	1	1	1	0

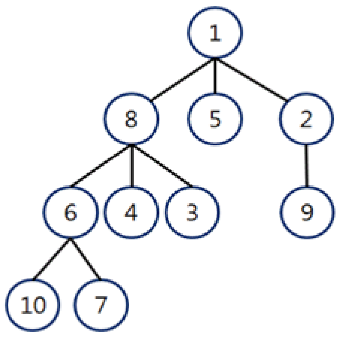
23. Quyidagi grafni DFS va BFS bo'yicha qidirish ketma ketligini yozing.



24. Quyidagi grafni DFS va BFS bo'yicha qidirish ketma ketligini yozing.

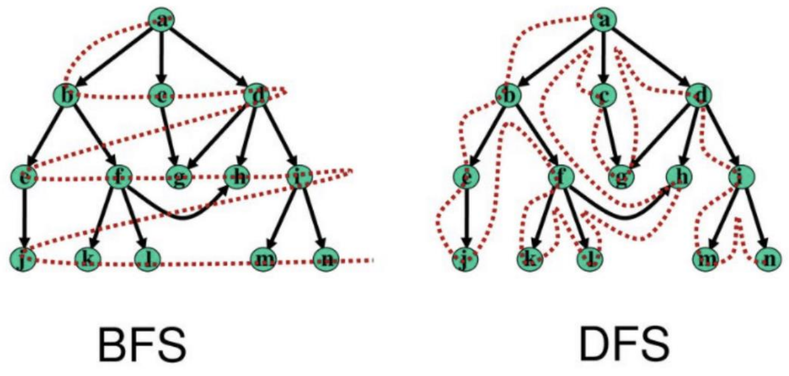


25. Quyidagi algoritm qaysi struktura bo'yicha qidirilgan ?

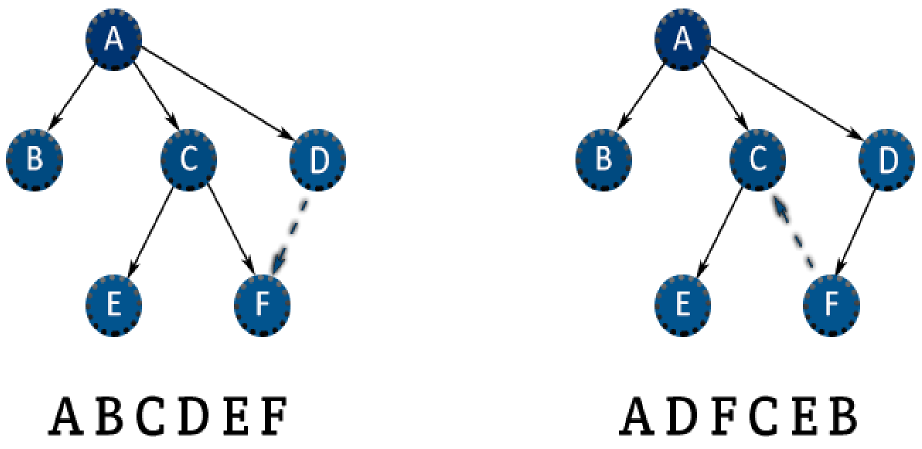


8 5 2 6 4 3 9 10 7

26. Quyidagi grafni DFS va BFS bo’yicha qidirish ketma ketligini yozing.



27. Quyidagi qidiruv algoritmni aniqlang. (BFS va DFS bo’yicha)



28.Quyidagi insendentlik matritsasi orqali yo’naltirilmagan grafni hosil qiling va maxsimum uch darajasini aniqlang.

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

29. Quyidagi insendentlik matritsadasi orqali grafni hosil qiling va uning hajmini aniqlang.

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \end{pmatrix}$$

30. Quyidagi qo’shnilik matritsasi yordamida grafni hosil qiling.

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

31. Quyidagi BFS bo’yicha qidiruv algoritmini tahlil qiling va natijani yozing.

```
#include<bits/stdc++.h>

using namespace std;

class Graph

{

   int V;

   vector<list<int>> adj;

public:

   Graph(int V);
```

```
    void addEdge(int v, int w);
    void BFS(int s);
};
Graph::Graph(int V)
{
    this->V = V;
    adj.resize(V);
}

void Graph::addEdge(int v, int w)
{
    adj[v].push_back(w);
}

void Graph::BFS(int s)
{
    vector<bool> visited;
    visited.resize(V,false);
    list<int> queue;
    visited[s] = true;
    queue.push_back(s);
    while(!queue.empty())
    {
        s = queue.front();
        cout << s << " ";
        queue.pop_front();

        for (auto adjecent: adj[s])
        {
            if (!visited[adjecent])
            {
                visited[adjecent] = true;
                queue.push_back(adjecent);
            }
        }
    }
}
int main()
{
    Graph g(4);
    g.addEdge(0, 1);
    g.addEdge(0, 2);
    g.addEdge(1, 2);
    g.addEdge(2, 0);
    g.addEdge(2, 3);
    g.addEdge(3, 3);

    cout << "Following is Breadth First Traversal "
         << "(starting from vertex 2) \n";
    g.BFS(2);

    return 0;
```

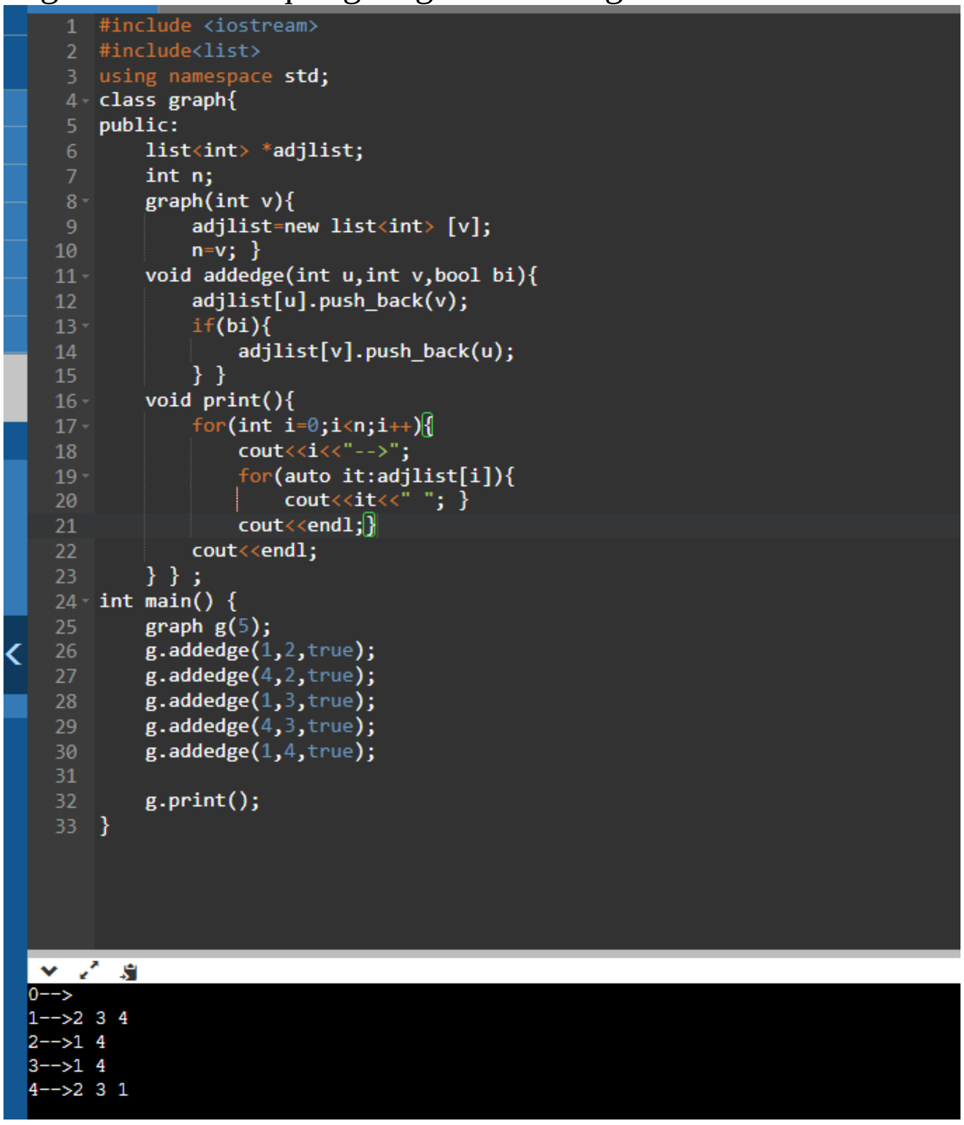
32. Quyidagi DFS bo'yicha qidiruv algoritmini tahlil qiling va natijani yozing..

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

// Graph class represents a directed graph
// using adjacency list representation
class Graph {
```

```
public:
    map<int, bool> visited;
    map<int, list<int> > adj;

    // function to add an edge to graph
    void addEdge(int v, int w);

    // DFS traversal of the vertices
    // reachable from v
    void DFS(int v);
};

void Graph::addEdge(int v, int w)
{
    adj[v].push_back(w); // Add w to v’s list.
}

void Graph::DFS(int v)
{
    // Mark the current node as visited and
    // print it
    visited[v] = true;
    cout << v << " ";

    // Recur for all the vertices adjacent
    // to this vertex
    list<int>::iterator i;
    for (i = adj[v].begin(); i != adj[v].end(); ++i)
        if (!visited[*i])
            DFS(*i);
}

// Driver's code
int main()
{

    Graph g;
    g.addEdge(0, 1);
    g.addEdge(0, 2);
    g.addEdge(1, 2);
    g.addEdge(2, 0);
    g.addEdge(2, 3);
    g.addEdge(3, 3);

    cout << "Following is Depth First Traversal"
```

```
            " (starting from vertex 2) \n";
    g.DFS(2);
    return 0;
}
```

19. C++ dasturlash tilida mashinada grafni realizatsiya qilinishi, dastur algoritmini tahlil qiling va grafni chizing.



```cpp
#include <iostream>
#include<list>
using namespace std;
class graph{
public:
    list<int> *adjlist;
    int n;
    graph(int v){
        adjlist=new list<int> [v];
        n=v; }
    void addedge(int u,int v,bool bi){
        adjlist[u].push_back(v);
        if(bi){
            adjlist[v].push_back(u);
        } }
    void print(){
        for(int i=0;i<n;i++){
            cout<<i<<"-->";
            for(auto it:adjlist[i]){
                cout<<it<<" "; }
            cout<<endl;}
        cout<<endl;
    } } ;
int main() {
    graph g(5);
    g.addedge(1,2,true);
    g.addedge(4,2,true);
    g.addedge(1,3,true);
    g.addedge(4,3,true);
    g.addedge(1,4,true);

    g.print();
}
```

```
0-->
1-->2 3 4
2-->1 4
3-->1 4
4-->2 3 1
```