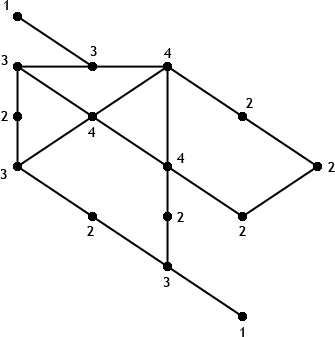
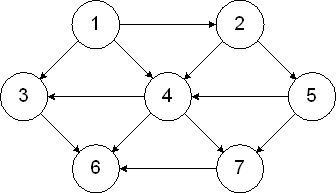
**7 – Topshiriq**

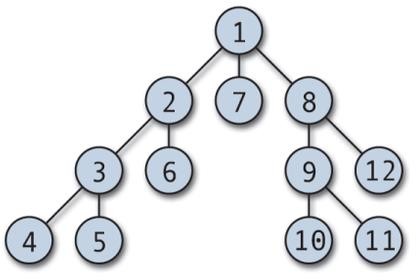
*Ilhomjonov Iqbolshoh*

2) 6.ta uchdan iborat orentrlanmagan, to’la garf hosil qiling va uning hajmini toping?

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



6) Quyidagi daraxt elementlari tahlil qiling



8) 7 bargdan iborat daraxt hosil qiling. Uning elementlarini tahlil qiling

10) Muvazanatlashgan daraxt hosil qiling va uning balandligini aniqlang.

12) 12 ta qirradan iborat graf hosil qiling , uni tugunlari sonini va chiqivchi darajasi 0 bo’lgan tugunlar sonini aniqlang.

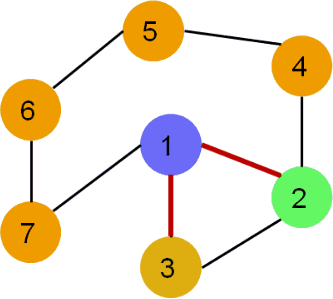
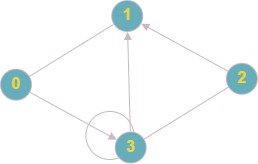
14) Asikl graf hosil qiling va kiriruvchisi 0 bo’lgan tugunni aniqlang.

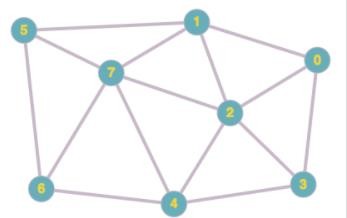
16) Daraxtlarga oid dastur tuzing

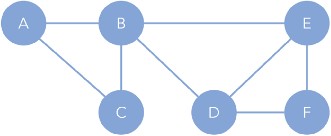
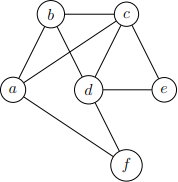
18) Quyida berilgan graflar bo’yicha berilgan vazifalarni hal qiling

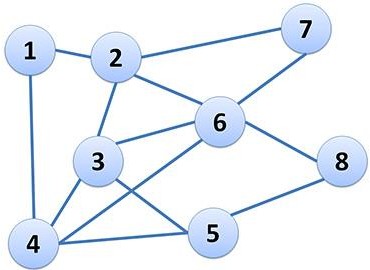
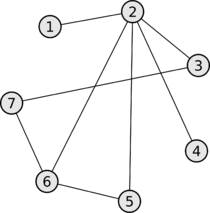
* 1. Berilgan grafning barcha tushunchalarini keltiring (uchlar, qirralar, grafning yo’nalishga ega yoki yo’qligi bo’yicha aniqlanishi, yakkalangan uch mavjudligi, regular yoki regular emasligi va hokazo).
  2. Grafni mashina xotirasida tasvirlash
  3. 2.1) Grafni uchlar qo’shniligi matritsasi orqali tasvirlang
  4. 2.2) Grafni qo’shnilik ro’yxati orqali tasvirlash
  5. 2.3) Grafning insidentlik matritsasi orqali tasvirlash

# a) b)

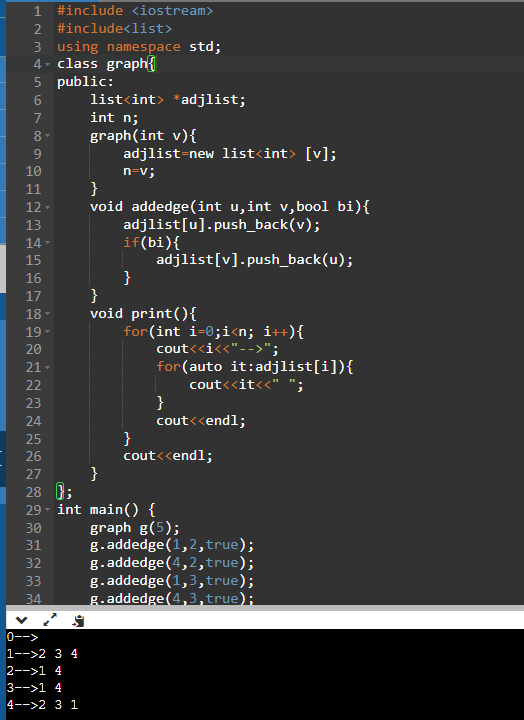


**c)**

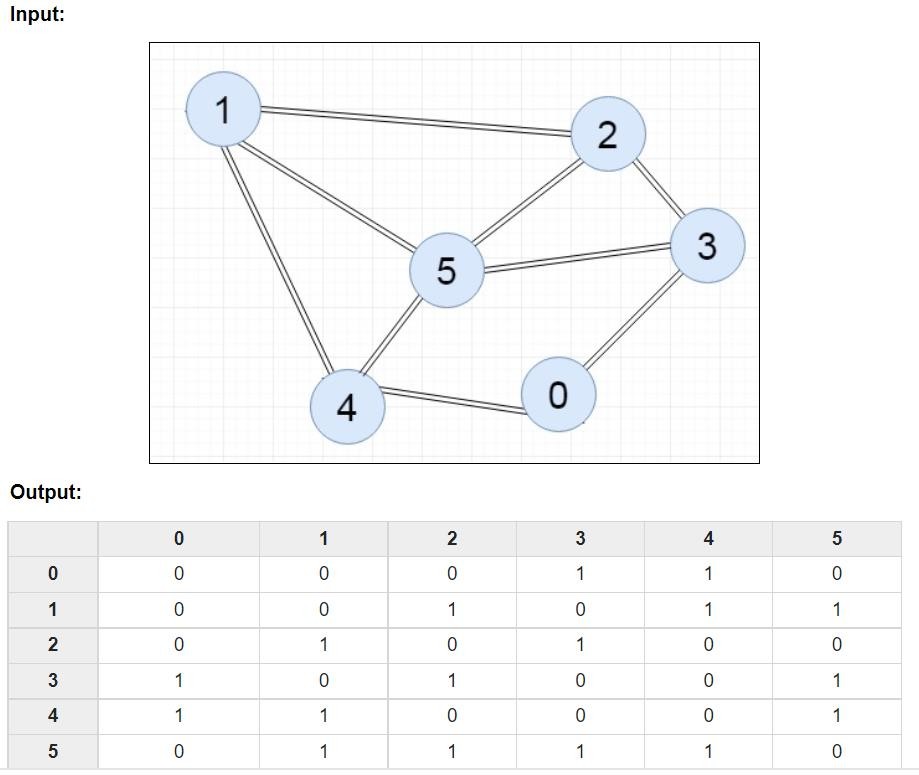


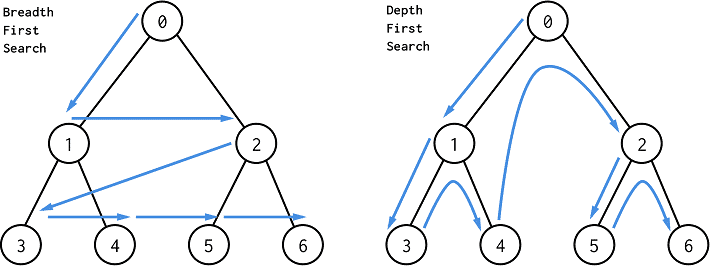
20) Quyidagi dastur kodini tahlil qiling.



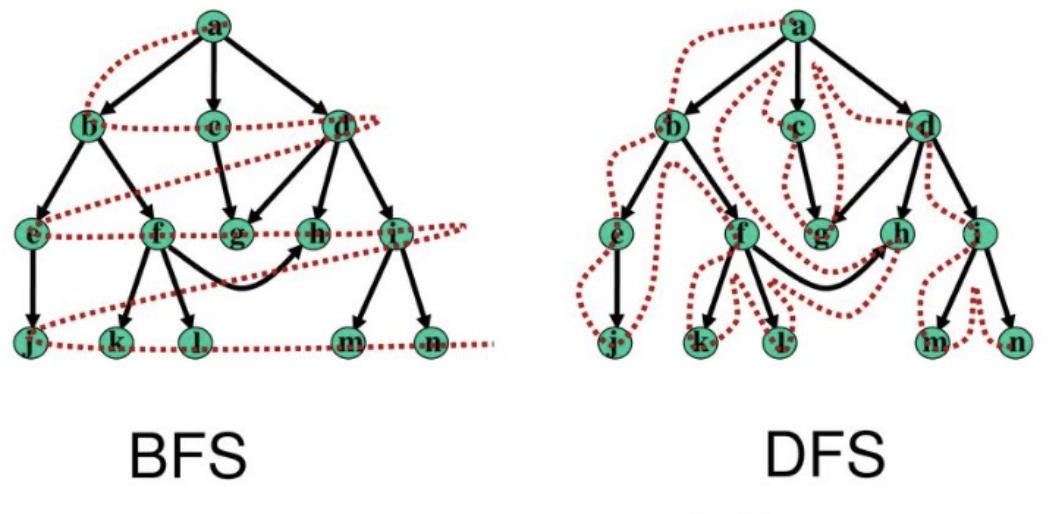
22) Quyidagi grafni va matritsani tahliling.



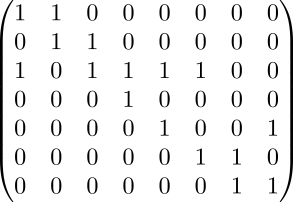
24) Quyidagi grafni DFS va BFS bo’yicha qidirish ketma ketligini yozing.



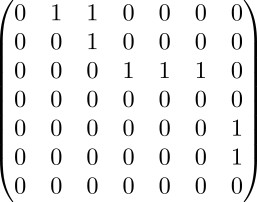
26) Quyidagi grafni DFS va BFS bo’yicha qidirish ketma ketligini yozing.



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



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



# 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;

}