**BFS and DFS**

#include<bits/stdc++.h>

#define N 5

using namespace std;

void DFS(int adjmatrix[N][N],int start){

int visited[N];

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

visited[i]=0;

}

stack<int> s;

s.push(start);

visited[start]=1;

while(!s.empty()){

int curr=s.top();

cout<<curr<<" ";

s.pop();

for(int v2=0;v2<N;v2++){

if(adjmatrix[curr][v2]==1 && visited[v2]==0){

visited[v2]=1;

s.push(v2);

break;

}

}

}

}

void BFS(int adjmatrix[N][N],int start){

int visited[N];

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

visited[i]=0;

}

queue<int> q;

q.push(start);

visited[start]=1;

while(!q.empty()){

start=q.front();

cout<<start<<" ";

q.pop();

for(int v2=0;v2<N;v2++){

if(adjmatrix[start][v2]==1 && visited[v2]==0){

visited[v2]=1;

q.push(v2);

}

}

}

}

int main(){

int adjmatrix[N][N]{

{0,1,0,0,1},

{1,0,1,0,0},

{0,1,0,1,0},

{0,0,1,0,1},

{1,0,0,1,0},

};

int start;

cout<<"Enter Start Node: ";

cin>>start;

cout<<"DFS Traversal: ";

DFS(adjmatrix,start);

cout<<endl;

cout<<"BFS Traversal: ";

BFS(adjmatrix,start);

}

**A\***

import copy

def printMatrix(mat):

    for i in range(3):

        print(mat[i])

def countMisplaced(currentState,finalState):

    count=0

    for i in range(3):

        for j in range(3):

            if(currentState[i][j] != -1 and currentState[i][j]!=finalState[i][j]):

                count+=1

    return count

def findPosOfMinusOne(mat):

    for i in range(3):

        for j in range(3):

            if(mat[i][j]== -1):

                return [i,j]

def moveLeft(mat,pos):

    if(pos[1]==0):

        return None

    m=copy.deepcopy(mat)

    m[pos[0]][pos[1]],m[pos[0]][pos[1]-1]=m[pos[0]][pos[1]-1],m[pos[0]][pos[1]]

    return m

def moveRight(mat,pos):

    if(pos[1]==2):

        return None

    m=copy.deepcopy(mat)

    m[pos[0]][pos[1]],m[pos[0]][pos[1]+1]=m[pos[0]][pos[1]+1],m[pos[0]][pos[1]]

    return m

def moveUp(mat,pos):

    if(pos[0]==0):

        return None

    m=copy.deepcopy(mat)

    m[pos[0]][pos[1]],m[pos[0]-1][pos[1]]=m[pos[0]-1][pos[1]],m[pos[0]][pos[1]]

    return m

def moveDown(mat,pos):

    if(pos[0]==2):

        return None

    m=copy.deepcopy(mat)

    m[pos[0]][pos[1]],m[pos[0]+1][pos[1]]=m[pos[0]+1][pos[1]],m[pos[0]][pos[1]]

    return m

def a\_star(initialState,finalState):

    currentState=copy.deepcopy(initialState)

    explored=[]

    gn=1;

    while(currentState!=finalState):

        left=moveLeft(currentState,findPosOfMinusOne(currentState))

        right=moveRight(currentState,findPosOfMinusOne(currentState))

        up=moveUp(currentState,findPosOfMinusOne(currentState))

        down=moveDown(currentState,findPosOfMinusOne(currentState))

        fnl=fnr=fnu=fnd=99999;

        if(left!=None):

            hnl=countMisplaced(left,finalState);

            fnl=hnl+gn;

        if(right!=None):

            hnr=countMisplaced(right,finalState);

            fnr=hnr+gn;

        if(up!=None):

            hnu=countMisplaced(up,finalState);

            fnu=hnu+gn;

        if(down!=None):

            hnd=countMisplaced(down,finalState);

            fnd=hnd+gn;

        minimum=min(fnl,fnr,fnu,fnd)

        if(minimum==fnl):

            currentState=copy.deepcopy(left)

            explored.append(currentState)

        if(minimum==fnr):

            currentState=copy.deepcopy(right)

            explored.append(currentState)

        if(minimum==fnu):

            currentState=copy.deepcopy(up)

            explored.append(currentState)

        if(minimum==fnd):

            currentState=copy.deepcopy(down)

            explored.append(currentState)

        gn+=1

    return explored

def printAllMatrix(mat):

    for i in range(len(mat)):

        print("\nState:",i+1,"\n")

        printMatrix(mat[i])

initialState=[[1,2,3],[-1,4,6],[7,5,8]]

finalState=[[1,2,3],[4,5,6],[7,8,-1]]

print(" \n\n------Initial Matrix-------\n")

printMatrix(initialState)

print(" \n\n------Final Matrix---------\n")

printMatrix(finalState)

print(" \n\n------States---------\n")

result= a\_star(initialState,finalState)

printAllMatrix(result)

**Job Scheduling**

#include<bits/stdc++.h>

using namespace std;

struct Job{

int id;

int deadline;

int profit;

};

bool compare(Job a,Job b){

if(a.profit==b.profit)

return a.deadline<b.deadline;

return a.profit>b.profit;

}

int main(){

int n;

cout<<"Enter Number of Jobs: ";

cin>>n;

vector<Job>v(n);

cout<<"Enter id, deadline, profit of each job:\n";

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

cin>>v[i].id;

cin>>v[i].deadline;

cin>>v[i].profit;

}

sort(v.begin(),v.end(),compare);

int res[n+1];

for(int i=0;i<n+1;i++){

res[i]=0;

}

int result=0;

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

for(int j=v[i].deadline;j>0;j--){

if(res[j]==0){

res[j]=v[i].id;

result+=v[i].profit;

break;

}

}

}

cout<<"Maximum Profit is: "<<result<<endl;

cout<<"Sequence of Job for Maximum Profit is: ";

for(int i=1;i<=n;i++){

if(res[i]!=0)

cout<<res[i]<<" ";

}

return 0;

}

**N-Queen**

#include<bits/stdc++.h>

using namespace std;

#define size 4

bool isSafe(int v[size][size],int r,int c){

for(int i=r-1,j=c;i>=0;i--){

if(v[i][j]==1)

return false;

}

for(int i=r-1,j=c-1;i>=0 && j>=0;i--,j--){

if(v[i][j]==1)

return false;

}

for(int i=r-1,j=c+1;i>=0 && j<size;i--,j++){

if(v[i][j]==1)

return false;

}

return true;

}

bool n\_queen(int v[size][size],int r){

if(r>=size){

return true;

}

for(int c=0;c < size;c++){

if(isSafe(v,r,c)){

v[r][c]=1;

if(n\_queen(v,r+1))

return true;

v[r][c]=0;

}

}

return false;

}

int main(){

int v[size][size];

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

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

v[i][j]=0;

}

}

n\_queen(v,0);

cout<<"Output for N queen problem:\n";

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

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

cout<<v[i][j]<<" ";

}

cout<<endl;

}

return 0;

}

**ChatBot**

#include<bits/stdc++.h>

using namespace std;

class Chatbot{

private:

vector<pair<string,string> >v;

public:

Chatbot(){

v =

{

{

"Hi",

"Hello"

},

{

"Who are you?",

"I am an program!"

},

{

"What is your name?",

"My name id ChatBot2.0"

},

{

"How are you?",

"I am Fine."

},

{

"Bye",

"It was nice talking to you user, Bye..."

}

};

}

void response(string input);

};

void Chatbot::response(string input){

if(input==""){

cout<<"Sorry....I'm not getting..";

}

for(int i=0;i<v.size();i++){

if(v[i].first==input){

cout<<"Chatbot: "<<v[i].second<<endl;

}

}

}

int main(){

Chatbot obj;

while(true){

string input;

cout<<"User: ";

getline(cin,input);

obj.response(input);

if(input=="Bye")

break;

cout<<"\n";

}

return 0;

}

**Expert system**

#include<bits/stdc++.h>

using namespace std;

int main(){

cout<<"\n------Covid 19 Expert System-----\n";

int countCovidSuspicious=0;

int severity=0;

int oxylevel=0;

int templevel=0;

vector<string> que1={

"\n What is the Patient's age: ",

"\n How many doses has patient taken: ",

"\n What is the patient's Oxygen level: ",

"\n What is the patients's temperature: "

};

vector<string>que2={

"\n Does the patient has cold or cough(yes/no): ",

"\n is the patient is able to recognize smell(yes/no): ",

"\n is the patient suffering from headache(yes/no): ",

"\n is the patient suffering from BP/Diabetes(yes/no): ",

"\n is the patient suffering from soar throat(yes/no): ",

"\n has the patient is contact with Covid patient(yes/no): "

};

for(int i=0;i<que2.size();i++){

string ans;

cout<<que2[i];

cin>>ans;

if(i==1 && ans=="no")

countCovidSuspicious++;

else if(i!=1 && ans=="yes")

countCovidSuspicious++;

}

for(int i=0;i<que1.size();i++){

if(i==0){

int age;

cout<<que1[i];

cin>>age;

if(age>12 && age<31)

severity+=0;

else if(age>=31 && age<51)

severity+=1;

else

severity+=2;

}

else if(i==1){

int does;

cout<<que1[i];

cin>>does;

if(does==2)

severity+=0;

else if(does==1)

severity+=1;

else if(does==0)

severity+=2;

}

else if(i==2){

int oxy;

cout<<que1[i];

cin>>oxy;

if(oxy>94)

severity+=0;

else if(oxy<=94 && oxy>=87)

severity+=1;

else if(oxy<87){

severity+=2;

countCovidSuspicious++;

oxylevel=1;

}

}

else if(i==3){

float temp;

cout<<que1[i];

cin>>temp;

if(temp<99.0)

severity+=0;

else if(temp<=101.20 && temp>99.0)

severity+=1;

else if(temp>101.20){

severity+=2;

countCovidSuspicious++;

templevel=1;

}

}

}

if(countCovidSuspicious>3){

if(severity<3){

cout<<"\n Patient has mild Symtoms of Covid positive.";

}

else if(severity>=3 && severity<=6){

cout<<"\n Patient can get admission in general ward.";

}

else{

cout<<"\n Patient looks critical.";

}

}

else{

cout<<"\n Patient is not Covid positive.";

}

if(oxylevel==1)

cout<<"\n Keep Monitoring Patient's Oxygen level.";

if(templevel==1)

cout<<"\n Keep Monitoring Patient's temperature level.";

return 0;

}