**Heuristic Algorithms: (A\* and AO\*)**

**A\* Algorithm**

***CODE:***

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

**using namespace std;**

**typedef pair<int, int> iPair;**

**struct node**

**{**

**node(int a, int b) {**

**dest=a;**

**weight= b;**

**}**

**int dest;**

**int weight;**

**};**

**void astar(int s,int d,vector<node>v[],int visited[],int parent[],int heu[],int dist[],int n)**

**{**

**priority\_queue< iPair, vector <iPair> , greater<iPair> > pq;**

**pq.push(make\_pair(heu[s],s));**

**dist[s]=0;**

**while(!pq.empty())**

**{**

**int p\_s = pq.top().second;**

**if(p\_s == d)**

**{**

**return;**

**}**

**pq.pop();**

**if(!visited[p\_s])**

**{**

**for(int i=0 ; i<v[p\_s].size() ; i++)**

**{**

**if(!visited[v[p\_s][i].dest])**

**{**

**int f = dist[p\_s] + v[p\_s][i].weight + heu[v[p\_s][i].dest];**

**pq.push(make\_pair(f,v[p\_s][i].dest));**

**dist[v[p\_s][i].dest] = dist[p\_s] + v[p\_s][i].weight;**

**parent[v[p\_s][i].dest] = p\_s;**

**}**

**}**

**visited[p\_s]=1;**

**}**

**}**

**}**

**int main()**

**{**

**cout<<"19MID0109 --- KOUSALYA D"<<endl;**

**cout<<"Enter number of vertices in the graph :"<<endl;**

**int n;**

**cin>>n;**

**vector<node>v[n];**

**cout<<"Enter the number of edges :"<<endl;**

**int e;**

**cin>>e;**

**for(int i=0;i<e;i++)**

**{**

**cout<<"Enter the from vertex: "<<endl;**

**int s;**

**cin>>s;**

**cout<<"Enter the to vertex: "<<endl;**

**int d;**

**cin>>d;**

**cout<<"Enter the weight "<<endl;**

**int w;**

**cin>>w;**

**cout<<endl;**

**v[s].push\_back(node(d, w));**

**}**

**int visited[n];**

**int parent[n];**

**int heu[n];**

**int dist[n];**

**cout<<"Enter the heuristic values"<<endl;**

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

**{**

**cout<<"For vertex "<<i<<" : ";**

**cin>>heu[i];**

**visited[i]=0;**

**parent[i]=i;**

**dist[i]=INT\_MAX;**

**}**

**cout<<"Enter the source : ";**

**int s;**

**cin>>s;**

**cout<<"Enter the destination : ";**

**int d;**

**cin>>d;**

**/\***

**v[0].push\_back(node(1, 2));**

**v[0].push\_back(node(2, 2));**

**v[0].push\_back(node(3, 2));**

**v[1].push\_back(node(4, 2));**

**v[1].push\_back(node(5, 2));**

**v[4].push\_back(node(6, 2));**

**v[4].push\_back(node(7, 2));**

**v[6].push\_back(node(8, 2));**

**v[7].push\_back(node(9, 2));**

**heu[0] = 4;**

**heu[1] = 3;**

**heu[2] = 5;**

**heu[3] = 5;**

**heu[4] = 2;**

**heu[5] = 4;**

**heu[6] = 3;**

**heu[7] = 1;**

**heu[8] = 4;**

**heu[9] = 0;**

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

**{**

**visited[i]=0;**

**parent[i]=i;**

**dist[i]=INT\_MAX;**

**}**

**\*/**

**astar(s,d,v,visited,parent,heu,dist,n);**

**int cur=d;**

**cout<<endl;**

**cout<<"Path from Vertex "<<s<<"to Vertex "<<d<<"is "<<endl;**

**stack<int> path;**

**do**

**{**

**path.push(cur);**

**cur=parent[cur];**

**}while(cur!=0);**

**path.push(s);**

**while(!path.empty())**

**{**

**cout<<path.top()<<"\t";**

**path.pop();**

**}**

**cout << endl;**

**return 0;**

**}**

**AO\* Algorithm**

**CODE:**

**#include <iostream>**

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

**using namespace std;**

**struct node**

**{**

**int data;**

**vector< vector<node\* >\* >v;**

**bool mark;**

**bool solved;**

**};**

**int edge\_cost=0;**

**void insert(node\* root)**

**{**

**cout<<"Enter data of node :"<<endl;**

**cin>>root->data;**

**//vector<vector<node\*> >vec=root->v;**

**cout<<"Enter number of OR nodes for value "<<root->data<<" :"<<endl;**

**int or\_no;**

**cin>>or\_no;**

**for(int i=0;i<or\_no;i++)**

**{**

**vector<node\*>\* ans=new vector<node\*>;**

**cout<<"Enter number of AND nodes for "<<i+1<<" or node for value "<<root->data<<" :"<<endl;**

**int and\_no;**

**cin>>and\_no;**

**for(int j=0;j<and\_no;j++)**

**{**

**node\* n=new node;**

**n->solved=false;**

**n->mark=false;**

**insert(n);**

**(\*ans).push\_back(n);**

**//cout<<"inserted node with value"<<n->data<<endl;**

**}**

**root->v.push\_back(ans);**

**}**

**}**

**void aostar(node\* root)**

**{**

**vector<node\*>\* min\_ans=new vector<node\*>;**

**(\*min\_ans).push\_back(root);**

**while(!root->solved)**

**{**

**node\* next\_node=root;**

**stack<node\*>st;**

**while(next\_node && next\_node->mark)**

**{**

**if((next\_node->v).size()==0)**

**{**

**root->solved=true;**

**return;**

**}**

**int cost=INT\_MAX;**

**st.push(next\_node);**

**for(unsigned int i=0;i<next\_node->v.size();i++)**

**{**

**vector<node\*>\*ans=(next\_node->v)[i];**

**vector<node\*> ans\_v=\*ans;**

**int temp\_cost=0;**

**for(unsigned int j=0;j<(ans\_v.size());j++)**

**{**

**node\* n=ans\_v[j];**

**temp\_cost+=n->data;**

**}**

**if(temp\_cost<cost)**

**{**

**min\_ans=ans;**

**cost=temp\_cost;**

**}**

**}**

**vector<node\*> min\_ans\_v=\*min\_ans;**

**next\_node=NULL;**

**for(unsigned int j=0;j<min\_ans\_v.size();j++)**

**{**

**if(min\_ans\_v[j]->mark)**

**{**

**next\_node=min\_ans\_v[j];**

**break;**

**}**

**}**

**}**

**vector<node\*> min\_ans\_v=\*min\_ans;**

**for(unsigned int j=0;j<min\_ans\_v.size();j++)**

**{**

**node\* n=min\_ans\_v[j];**

**cout<<"Exploring :"<<n->data<<endl;**

**int final\_cost=INT\_MAX;**

**if(n->v.size()==0)**

**{**

**n->mark=true;**

**}**

**else{**

**for(unsigned int i=0;i<n->v.size();i++)**

**{**

**vector<node\*>\*ans=(n->v)[i];**

**vector<node\*> ans\_v=\*ans;**

**int temp\_cost=0;**

**for(unsigned int j=0;j<(ans\_v.size());j++)**

**{**

**node\* n=ans\_v[j];**

**temp\_cost+=n->data;**

**temp\_cost+=edge\_cost;**

**}**

**if(temp\_cost<final\_cost)**

**{**

**final\_cost=temp\_cost;**

**}**

**}**

**n->data=final\_cost;**

**n->mark=true;**

**}**

**cout<<"Marked : "<<n->data<<endl;**

**}**

**for(int i=0;i<20;i++) cout<<"=";**

**cout<<endl;**

**while(!st.empty())**

**{**

**node\* n=st.top();**

**cout<<n->data<<" ";**

**st.pop();**

**int final\_cost=INT\_MAX;**

**for(unsigned int i=0;i<n->v.size();i++)**

**{**

**vector<node\*>\*ans=(n->v)[i];**

**vector<node\*> ans\_v=\*ans;**

**int temp\_cost=0;**

**for(unsigned int j=0;j<(ans\_v.size());j++)**

**{**

**node\* n=ans\_v[j];**

**temp\_cost+=n->data;**

**temp\_cost+=edge\_cost;**

**}**

**if(temp\_cost<final\_cost)**

**{**

**min\_ans=ans;**

**final\_cost=temp\_cost;**

**}**

**}**

**n->data=final\_cost;**

**}**

**cout<<endl;**

**next\_node=root;**

**}**

**}**

**void print(node\* root)**

**{**

**if(root)**

**{**

**cout<<root->data<<" ";**

**vector<vector<node\*>\* >vec=root->v;**

**for(unsigned int i=0;i<(root->v).size();i++)**

**{**

**vector<node\*>\* ans=(root->v)[i];**

**vector<node\*> ans\_v=\*ans;**

**for(unsigned int j=0;j<ans\_v.size();j++)**

**{**

**node\* n=ans\_v[j];**

**print(n);**

**}**

**}**

**}**

**return;**

**}**

**int main()**

**{**

**cout<<”19MID0109 --- KOUSALYA D”<<endl;**

**node\* root=new node;**

**root->solved=false;**

**root->mark=false;**

**insert(root);**

**cout<<endl;**

**cout<<"Enter the edge cost: "<<endl;**

**cin>>edge\_cost;cout<<endl;**

**cout<<"The tree is as follows :"<<endl;**

**print(root);**

**cout<<endl;**

**aostar(root);**

**cout<<"The minimum cost is : "<<root->data<<endl;**

**return 0;**

**}**