人工智能实验五

171491218刘晓萌

一．实验题目

A\*算法

二．实验分析

基本原理：

公式表示为： f(n)=g(n)+h(n),

其中 f(n) 是从初始点经由节点n到目标点的估价函数，

g(n) 是在状态空间中从初始节点到n节点的实际代价，

h(n) 是从n到目标节点最佳路径的估计代价。

保证找到最短路径（最优解的）条件，关键在于估价函数f(n)的选取：

首先将起始结点S放入OPEN表，CLOSE表置空，算法开始时：

1、如果OPEN表不为空，从表头取一个结点n，如果为空算法失败。

2、n是目标解吗？是，找到一个解（继续寻找，或终止算法）。

3、将n的所有后继结点展开，就是从n可以直接关联的结点（子结点），如果不在CLOSE表中，就将它们放入OPEN表，并把S放入CLOSE表，同时计算每一个后继结点的估价值f(n)，将OPEN表按f(x)排序，最小的放在表头，重复算法，回到1。

A\*搜索估价值h(n)<= n到目标节点的距离实际值，这种情况下，搜索的点数多，搜索范围大，效率低。但能得到最优解。并且如果h(n)=d(n)，即距离估计h(n)等于最短距离，那么搜索将严格沿着最短路径进行， 此时的搜索效率是最高的。如果 估价值>实际值,搜索的点数少，搜索范围小，效率高，但不能保证得到最优解。

三．实验内容

publicclassAXing{

intMaxWeight=10000;//表示无穷大

Stack stack=newStack<Integer>();

/\*\*A\*搜索

\* @param g：图

\* @param H：启发式函数值

\* @param v0：初始值

\* @param end：目标值

\*/

publicvoid A\_Search(Graph g,int H[],int v0,int end){

boolean flag=true;

int x;//表示栈顶元素

int vex;//寻找目标节点

intMinF,MinVex= v0;//记录最小的f(n)和对应的节点

int[][]GHF=newint[g.path.length][3];//分别用于存储g(n),h(n),f(n)

for(int i =0; i < g.path.length; i++){

GHF[i][0]=0;

GHF[i][2]=MaxWeight;//对f(n)初始化,1000表示无穷大

}

stack.push(v0);//v0入栈

GHF[v0][0]=0;//g(n)

GHF[v0][1]=H[v0];//h(n)

GHF[v0][2]=GHF[v0][0]+GHF[v0][1];//f(n)

g.mark[v0]=1;

while(flag){

MinF=MaxWeight;

x=(Integer) stack.peek();

//处理第一个子节点

vex=g.getFirstVex(x);

if(vex==end){//找到目标节点

stack.push(vex);

g.mark[vex]=1;

break;

}

if(vex!=-1){//子节点能找到，继续

if(g.mark[vex]==0){//没被访问

GHF[vex][0]=GHF[x][0]+g.path[x][vex];//节点vex的g(n)

GHF[vex][1]=H[vex];//节点vex的h(n)

GHF[vex][2]=GHF[vex][0]+GHF[vex][1];

if(GHF[vex][2]<MinF){

MinF=GHF[vex][2];

MinVex=vex;

}

}//处理剩下的邻接点（宽度遍历）

while(vex!=-1){

vex=g.getNextVex(x, vex);

if(vex!=-1&&g.mark[vex]==0){//有邻节点

GHF[vex][0]=GHF[x][0]+g.path[x][vex];//节点vex的g(n)

GHF[vex][1]=H[vex];//节点vex的h(n)

GHF[vex][2]=GHF[vex][0]+GHF[vex][1]；

if(GHF[vex][2]<MinF){

MinF=GHF[vex][2];

MinVex=vex;

}

}

if(vex==-1){//没有邻接点了，此时确定最小消耗节点，并压栈

stack.push(MinVex);

g.mark[MinVex]=1;

break;

}

if(vex==end){

stack.push(vex);//压栈目标节点

g.mark[vex]=1;

flag=false;

break;

}

}

}

else{//没有子节点或者子节点被访问了，循环出栈

while(vex==-1){

stack.pop();

}

}

}

newMain().show(g, stack);

}

}