**ai第三次实验(A\*算法)**

前两次实验我写了一个dfs和一个迭代加深搜索，这节课我决定将搜索进行到底，所以我选择了A\*算法。我依然是在bzoj上选择的一道题来作为这次实验课的素材。这道题让我明白了A\*算法的神奇之处，**其实它的核心就是f(x)=g(x)+h(x)**。

先来写一写我对A\*的理解。核心式子f(x)=g(x)+h(x),这里的g(x)表示的是从搜索起点到x的花费，h(x)表示的是从x到达终点预计的花费。每次我们都选f(x)最小的进行走，这样我们就能找到花费较少的路径了。

**h(x)分下面三种情况:**

**h(x)=0**，那么f(x)=g(x)这个直接变成了广度优先搜索。

**0<h(x)<实际花费**，这个我们就一定可以找到一条最短的路径(如果路径存在的话)，因为如果存在另外一条更短的评估路径，就会选择更小的那个，。

**h(x)>实际花费**，这个就可能很快就能找到一个路径，但不一定是最优解。

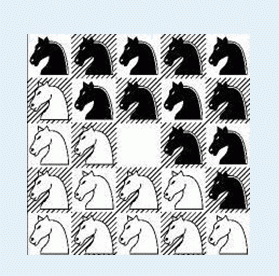
**因此，我们使用A\*算法的时候，我们需要权衡的就是时间和距离(也就是我们需要怎样的解)。**如果我们一定要求最短距离，那么我们的h(x)一定要小于实际花费；但是如果我们需要的是时间上快一点找到解，我们又不需要找到最优解，那么我们的预估函数h(x)可以选择大于实际距离。

上面就是我对A\*算法的一些理解。

下面来看bzoj 1085

**题目：**

在一个5×5的棋盘上有12个白色的骑士和12个黑色的骑士， 且有一个空位。在任何时候一个骑士都能按照骑士的走法（它可以走到和它横坐标相差为1，纵坐标相差为2或者横坐标相差为2，纵坐标相差为1的格子）移动到空位上。 给定一个初始的棋盘，怎样才能经过移动变成如下目标棋盘： 为了体现出骑士精神，他们必须以最少的步数完成任务。

(目标棋盘)

**想法：**

这道题给我感觉就是用搜索，但是如果不大量减枝那么时间复杂度太高，我觉得这里的A\*算法实际上起到的是一种减枝的作用，因为这个是状态的转移。当我做完这道题的时候我觉得这道题更像一道迭代加深搜索，只是减枝的时候用到了A\*算法的预估函数h(x)的思想，也让我想明白了我原来再做很多题的过程中无形地用到了A\*算法进行减枝。

cd：

#include<iostream>

#include<algorithm>

#include<cstdio>

#include<cstring>

using namespace std;

typedef long long ll;

int k;

int flag;

char s[10][10];

int a[10][10];

int dx[10]={1,2,1,2,-1,-2,-1,-2};

int dy[10]={2,1,-2,-1,-2,-1,2,1};

int ans[10][10]={{1,1,1,1,1},{0,1,1,1,1},{0,0,-1,1,1},{0,0,0,0,1},{0,0,0,0,0}};

//1表示黑，0表示白

bool judge()

{

for(int i=0;i<5;i++)

for(int j=0;j<5;j++){

if(a[i][j]!=ans[i][j])

return 0;

}

return 1;

}

bool check(int t)

{

int m=0;

for(int i=0;i<5;i++)

for(int j=0;j<5;j++){

if(a[i][j]!=ans[i][j]){

m++;

if(m+t>k) return 0;

}

}

return 1;

}

void Astar(int t,int x,int y)

{

if(t==k){

if(judge()) flag=1;

return ;

}

if(flag)

return;

int xx,yy;

for(int i=0;i<8;i++){

xx=x+dx[i];

yy=y+dy[i];

if(xx<0||xx>4||yy<0||yy>4)

continue;

swap(a[xx][yy],a[x][y]);

if(check(t)) Astar(t+1,xx,yy);

swap(a[xx][yy],a[x][y]);

}

}

int main()

{

int t;

scanf("%d",&t);

while(t--){

for(int i=0;i<5;i++)

scanf("%s",s[i]);

int x,y;

for(int i=0;i<5;i++)

for(int j=0;j<5;j++){

if(s[i][j]=='1')

a[i][j]=1;

else if(s[i][j]=='0')

a[i][j]=0;

else

a[i][j]=-1;//表示这个位置是空着的

}

for(x=0;x<5;x++){

for(y=0;y<5;y++)

if(s[x][y]=='\*')

break;//这里找到空格的初始位置

if(s[x][y]=='\*')

break;

}

flag=0;

for(k=0;k<=15;k++){

Astar(0,x,y);

if(flag){

printf("%d\n",k);

break;

}

}

if(!flag)

printf("-1\n");

}

return 0;

}