神犇(shenben)

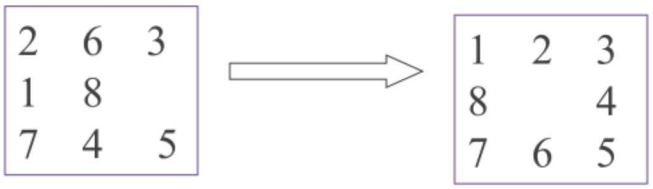
“经历只有自己去体会，别人的眼中只有结果”

luogu P1379 八数码难题（A*算法入门详细讲解）

启发式搜索算法(A算法)

- 在BFS算法中，若对每个状态n都设定估价函数f(n)=g(n)+h(n)，并且每次从Open表中选节点进行扩展时，都选取f值最小的节点，则该搜索算法为启发式搜索算法，又称A算法。
- g(n)：从起始状态到当前状态n的代价
- h(n)：从当前状态n到目标状态的估计代价

A算法的例子--八数码



定义估价函数：

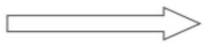
$f(n) = g(n) + h(n)$

g(n)为从初始节点到当前节点的步数

h(n)为当前节点“不在位”的方块数

h计算举例

2	6	3
1	8	
7	4	5



1	2	3
8		4
7	6	5

2,6,1,8,4,5都不在位，因此h(n) = 6

A*算法

- A算法中的估价函数若选取不当，则可能找不到解，或找到的解也不是最优解。因此，需要对估价函数做一些限制，使得算法确保找到最优解（步数，即状态转移次数最少的解）。A*算法即为对估价函数做了特定限制，且确保找到最优解的A算法。

A*算法

- f*(n)=g*(n) +h*(n)

  f*(n)：从初始节点S0出发，经过节点n到达目标节点的最小步数(真实值）。

  g*(n): 从S0出发，到达n的最少步数（真实值）

  h*(n): 从n出发，到达目标节点的最少步数（真实值）

  估价函数f(n)则是f*(n)的估计值。

5. 【NOIP2016】天天爱跑步 (2163)

A*算法

- f (n)=g(n) +h(n) ,且满足以下限制：

g(n)是从s0到n的真实步数（未必是最优的），因此：

g(n)>0 且g(n)>=g*(n)

h(n)是从n到目标的估计步数。估计总是过于乐观的，即

h(n)<= h*(n)

且h(n)相容，则A算法转变为A*算法。A*正确性证明略。

A*算法

h(n)的相容:

如果h函数对任意状态s1和s2还满足：

h(s1) <= h(s2) + c(s1,s2)

c(s1,s2）是s1转移到s2的步数，则称h是相容的。

h相容能确保随着一步步往前走，f递增，这样A*能更高效到最优解。

h相容 =>

g(s1) + h(s1) <= g(s1) + h(s2) +c(s1,s2) = g(s2)+h(S2)

=>

f(s1) <= f(s2) 即f是递增的。

A*算法

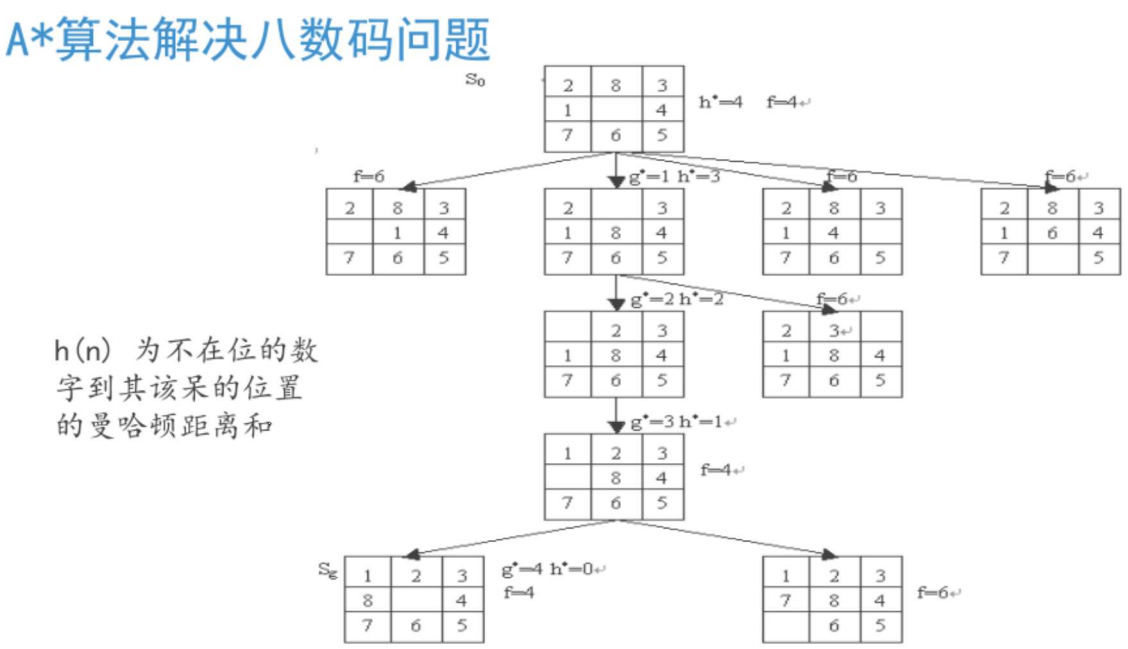
A*算法的搜索效率很大程度上取决于估价函数h(n)。一般说来，满足h(n)≤h*(n)的前提下，h(n)的值越大越好。

八数码问题：

方案一. h(n) 为不在位的数字个数

方案二. h(n) 为不在位的数字到其该呆的位置的曼哈顿距离

后者优于前者　　代码采用方案2



代码实现细节

首先定义一个dis[10][10]数组，记录偏移距离，我们把地图中的点按行标则dis[i][j]表示从第i个点到第j个点至少移动的步数。
举个例子：
点的编号：1 2 3
4 5 6
7 8 9
那么显然dis[1][2]=1,dis[2][9]=3,dis[3][4]=3
这个dis数组可通过预处理完成，应该是这个样子：

```
0 1 2 1 2 3 2 3 4
1 0 1 2 1 2 3 2 3
2 1 0 3 2 1 4 3 2
1 2 3 0 1 2 1 2 3
2 1 2 1 0 1 2 1 2
3 2 1 2 1 0 3 2 1
2 3 4 1 2 3 0 1 2
3 2 3 2 1 2 1 0 1
4 3 2 3 2 1 2 1 0
```

接着定义一个goal[10]数组，goal[i]表示数字i在目标状态中的位置也可以通过预处理完成，

```
int goal[10]={5,1,2,3,6,9,8,7,4};
```

然后定义一个now[10]数组，now[i]表示数字i在当前状态中的位置，这个需要动态维护。
那么估价函数的值当然就是数字0-8 的偏移量之和了。

```
#include<cstdio>
#include<cstring>
#include<iostream>
using namespace std;
const int N=10;
const int dx[]={0,0,1,-1};
const int dy[]={1,-1,0,0};
int flag,now[N],goal[N];
int dis[N][N],a[N][N],mp[N][N];
inline int calcx(int x){return (x-1)/3+1;}
inline int calcy(int x){return x%3?x%3:3;}
inline int abs(int x){return x>0?x:-x;}
inline int h(){int t=0;for(int i=1;i<=9;i++) t+=dis[now[i]][goal[i]];return t;}
inline int check(){for(int i=0;i<9;i++) if(now[i]!=goal[i]) return 0;return 1;}
```

```
void dfs(int depth,int x,int y,int lim){
    if(depth+h()>lim) return ;
    if(check()){flag=1;return ;}
    for(int i=0,nx,ny;i<4;i++){
        nx=x+dx[i];
        ny=y+dy[i];
        if(flag) return ;
        if(nx>0&&nx<=3&&ny>0&&ny<=3){
            swap(a[x][y],a[nx][ny]);swap(now[a[x][y]],now[a[nx][ny]]);
            dfs(depth+1,nx,ny,lim);
            swap(a[x][y],a[nx][ny]);swap(now[a[x][y]],now[a[nx][ny]]);
        }
    }
}
void pre(){
    for(int i=1;i<=9;i++)
        for(int j=i+1;j<=9;j++)
            dis[i][j]=dis[j][i]=calcx(j)-calcx(i)+abs(calcy(j)-calcy(i));
}
int main(){
    pre();
    goal[0]=5;goal[1]=1;goal[2]=2;goal[3]=3;goal[4]=6;goal[5]=9;goal[6]=8;goal[7]=7;goal[8]=4;
    int sx,sy;
    for(int i=1,x,y,z;i<=9;i++){
        scanf("%1d",&z);
        x=calcx(i);y=calcy(i);
        mp[x][y]=z;now[z]=i;
        if(!z) sx=x,sy=y;
    }
    for(int i=0;;i++){
        memcpy(a,mp,sizeof mp);
        dfs(0,sx,sy,i);
        if(flag){printf("%d\n",i);break;}
    }
    return 0;
}
```

内心独白：

——将来的我，一定会感谢现在的自己!

版权声明：

神犇(shenben)
关注 - 7
粉丝 - 146
+加关注

« 上一篇：codevs 5967 [SDOI2017]相关分析
» 下一篇：POJ2286 The Rotation Game[IDA*迭代加深搜索]