

A*

简介&思想

通过估价函数预计搜索到的结点与终点的距离，以估价距离从小到大为顺序搜索终点，得到最短路，减少搜索时间。与dijkstra相似。

估价函数越大搜索越少，但估价函数小于等于真实距离时才能得到正确的结果，因此要注意估价函数的选取，可以调整估价函数的倍数。

例题

八数码难题

来源：洛谷 p1379

概述：3×3的棋盘上有八个棋子和一个空白，空白前后左右的棋子可以填到空白处，给定初始布局和目标布局，求最少移动次数。

答案：统计当前布局每个数字距离目标布局中相同数字的距离作为估价，估计距离则为估价+当前移动次数，按照类似dij的做法，按估价距离的优先级更新结点，同时更新已知结点的最优移动次数并更新其估价距离，直到得到终点。

```
#include<cstdio>
#include<map>
#include<queue>
#include<string>
#include<cstring>
using namespace std;
map<string,int>vist,D,H;
struct unit
{
    string state;
    int d,h,f,p;
    bool operator < (const unit&x) const
    {
        return f>x.f;
    }
};
priority_queue<unit>q;
string End("123804765");
int
targ[9]={4,0,1,2,5,8,7,6,3},dpos[4][9]={{-1,-1,-1,0,1,2,3,4,5},{3,4,5,6,7,8,-1,-1,-1},{-1,0,1,-1,3,4,-1,6,7},{1,2,-1,4,5,-1,7,8,-1}};
int
dis[9][9]={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}};
void read()
{
    string s;
    s.resize(9);
    scanf("%s", &s[0]);
    unit zero;
    zero.state = s;
    zero.d = zero.h = 0;
    for(int i = 0; i < 9; i++)
    {
        zero.h += dis[i][targ[s[i] - 48]];
        if(s[i] == 48)
            zero.p = i;
    }
    zero.f = zero.h >> 1;
    q.push(zero);
    D[s] = 0;
    H[s] = zero.h;
}
int main()
{
    read();
    int m = 1;
    unit head;
    while(m--)
    {
        head = q.top();
        q.pop();
        if(head.state == End)
        {
            printf("%d", head.d);
            return 0;
        }
        if(vist[head.state] == -1)
            continue;
        vist[head.state] = -1;
        string state_;
        for(int i = 0, p_, h_, d_; i < 4; i++)
        {
            p_ = dpos[i][head.p];
            if(p_ >= 0)
            {
                state_ = head.state;
                state_[head.p] = state_[p_];
                state_[p_] = 48;
                if(vist[state_] == -1) continue;
                d_ = head.d + 1;
                if(!vist[state_])
                {
```

```
        h_=head.h-dis[head.p][targ[0]]-
dis[p_][targ[head.state[p_]-48]]+dis[head.p][targ[state_[head.p]-48]]+dis[p
_][targ[0]];
        q.push({state_,d_,h_,d_+(h_>>1),p_});
        m++;
        D[state_]=d_;
        H[state_]=h_;
        vist[state_]=1;
    }
    else if(d_<D[state_])
    {
        q.push({state_,d_,d_+(H[state_]>>1),p_});
        m++;
        D[state_]=d_;
    }
}
}
}
return 0;
}
```

总结

From:
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:no_morning_training:shaco:%E7%9F%A5%E8%AF%86%E7%82%B9:%E6%90%9C%E7%B4%A2:a

Last update: 2020/08/20 19:30