Problem D　　ZOJ 2404

一.题目大意

地图上有 m 个小人“M”，有 h 个小房子“H”，小人每移动一格需要费用 1￥。问若使小人全部进入房子，需要的最小费用。

二.题目分析

题目模型中有两个明显的特点，第一是匹配，使小人和房子的配对数最多（此题为一对一关系），第二是费用最少。所以推出最小费用流模型。

三.算法分析

最小费用流基本 bellman_ford 算法寻找剩余流量，然后在图中进行更新。详见代码。

四.算法改进

bellman_ford 算法可以替换为效率更高的 SPFA。同样这题还有一个 KM 算法的解法。

五.参考代码

```
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <algorithm>
#include <iostream>
#include <vector>
#include <bitset>
#include <stack>
#include <queue>

using namespace std;

#define print(x) cout<<x<<endl
#define input(x) cin>>x

#define SIZE 256
#define SOURCE 0
#define SINK m+n+1
#define pb push_back
#define INF 1<<25

struct point
{
        int x,y;

        point(){}
        point(int i_x,int i_y)
        {
                x=i_x;y=i_y;
        }

        int dis(point& p)
        {
                return abs(p.x-x)+abs(p.y-y);
        }
};

vector<point> man;
vector<point> house;
```

```
int flow[SIZE][SIZE];
int dis[SIZE];
int cost[SIZE][SIZE];
int pre[SIZE];
int ans=0;

int w,l;
int m,n;
//n->man  m->hause

bool bellman_ford()//Bellman_ford找最小费用增广路
{
        memset(pre,-1,sizeof(pre));
        for(int i=SOURCE;i<=SINK;i++) dis[i]=INF;
        dis[SOURCE]=0;
        bool flag=true;
        while(flag)
        {
                flag=false;
                for(int i=SOURCE;i<=SINK;i++)
                {
                        if(dis[i]>=INF) continue;
                        else
                        {
                                for(int j=SOURCE;j<=SINK;j++)
                                {
                                        if(flow[i][j]&&cost[i][j]<INF&&dis[i]+cost[i][j]<dis[j])
                                        {
                                                dis[j]=dis[i]+cost[i][j];
                                                pre[j]=i;
                                                flag=true;
                                        }
                                }
                        }
                }
        }
        return dis[SINK]!=INF;
}
void mcmf()//最小费用最大流
{
        int now=SINK,c=0,minflow=INF;
        while(bellman_ford())
        {
                while(now!=SOURCE)
                {
                        int p=pre[now];
                        minflow=min(minflow,flow[p][now]);
                        now=p;
                }

                now=SINK;
                while(now!=SOURCE)
                {
                        int p=pre[now];
                        flow[p][now]-=minflow;
                        flow[now][p]+=minflow;
                        cost[now][p]=-cost[p][now];
                        c+=minflow*cost[p][now];
                        now=p;
                }
        }
        ans=c;
}


void makeG()
{
        n=man.size()-1;
        m=house.size()-1;
```

```
        memset(flow,0,sizeof(flow));
        for(int i=SOURCE;i<=SINK;i++)
        {
                for(int j=SOURCE;j<=SINK;j++) cost[i][j]=INF;
        }
        for(int i=1;i<=n;i++)
        {
                flow[SOURCE][i]=1;
                cost[SOURCE][i]=0;
                for(int j=1;j<=m;j++)
                {
                        flow[i][j+n]=1;
                        cost[i][j+n]=man[i].dis(house[j]);
                }
        }
        for(int i=1;i<=m;i++)
        {
                flow[i+n][SINK]=1;
                cost[i+n][SINK]=0;
        }
}


int main()
{
        char tmpStr[SIZE];
        while(input(l>>w) && l+w)
        {
                ans=0;
                man.clear();
                house.clear();
                man.pb(point(-1,-1));
                house.pb(point(-1,-1));//占位
                for(int i=0;i<l;i++)
                {
                        input(tmpStr);
                        for(int j=0;tmpStr[j];j++)
                        {
                                if(tmpStr[j]=='m')
                                {
                                        man.pb(point(j,i));
                                }
                                else if(tmpStr[j]=='H')
                                {
                                        house.pb(point(j,i));
                                }
                        }
                }
                makeG();
                mcmf();
                print(ans);
        }
        return 0;
}
```