## 百练 4115:鸣人和佐助

### 1 原题目

**描述**

佐助被大蛇丸诱骗走了，鸣人在多少时间内能追上他呢？



已知一张地图（以二维矩阵的形式表示）以及佐助和鸣人的位置。地图上的每个位置都可以走到，只不过有些位置上有大蛇丸的手下，需要先打败大蛇丸的手下才能到这些位置。鸣人有一定数量的查克拉，每一个单位的查克拉可以打败一个大蛇丸的手下。假设鸣人可以往上下左右四个方向移动，每移动一个距离需要花费1个单位时间，打败大蛇丸的手下不需要时间。如果鸣人查克拉消耗完了，则只可以走到没有大蛇丸手下的位置，不可以再移动到有大蛇丸手下的位置。佐助在此期间不移动，大蛇丸的手下也不移动。请问，鸣人要追上佐助最少需要花费多少时间？

**输入**

输入的第一行包含三个整数：M，N，T。代表M行N列的地图和鸣人初始的查克拉数量T。0 < M,N < 200，0 ≤ T < 10  
后面是M行N列的地图，其中@代表鸣人，+代表佐助。\*代表通路，#代表大蛇丸的手下。

**输出**

输出包含一个整数R，代表鸣人追上佐助最少需要花费的时间。如果鸣人无法追上佐助，则输出-1。

**样例输入**

样例输入1

4 4 1

#@##

\*\*##

###+

\*\*\*\*

样例输入2

4 4 2

#@##

\*\*##

###+

\*\*\*\*

**样例输出**

样例输出1

6

样例输出2

4

### 2 基础知识

广度优先搜索（解法一用到）

### 3 解法

#### 3.1 解法1（C++已过）

这题和普通的广度优先搜索1有点差别，很有意思这道题。我最开始的做法就是和普通的广搜一样标记数组开成二维的，然后那么继续广搜，然后发现提交后WA。百思不得其解，百度了下，醒悟了，其实这个问题我想到了，我以为我结构体node中加了记录查克拉的变量t就会避免此问题，结果没避免，此问题就是此道题目每个点访问次数不止一次，因为同一个点不同访问的次数时鸣人手里的查克拉数目可能会不一样。所以我们的标记数组要开成三维的，我之前就是忽略了这点，三维标记数组代表在[x,y]这点有k个查克拉时是否走过。这样问题就解决了。

本人代码：

|  |
| --- |
| #include <iostream>  #include<algorithm>  #include <cstring>  #include <cstdio>  #include <cstdlib>  #include <cmath>  #include <vector>  #include <queue>  using namespace std;  struct node  {      int x,y;      int t;      int s;      node() {}      node(int xx,int yy,int tt,int ss):x(xx),y(yy),t(tt),s(ss) {}  };  //标记[x,y]点有k个查克拉的鸣人是否走过  int book[205][205][15];  int main()  {      int M,N,T;      while (scanf("%d %d %d",&M,&N,&T) != EOF)      {          getchar();          char mapp[205][205];          memset(book,0,sizeof(book));          int sx,sy;          for (int i = 0; i < M; i++)          {              for (int j = 0; j < N; j++)              {                  scanf("%c",&mapp[i][j]);                  if (mapp[i][j] == '@')                  {                      sx = i;                      sy = j;                  }              }              getchar();          }          int res = 0;          int coor[4][2] = {0,1,0,-1,-1,0,1,0};//上下左右          queue<node> q;          q.push(node(sx,sy,T,0));          book[sx][sy][T] = 1;          //广度优先搜索          while (!q.empty())          {              int xx = q.front().x;              int yy = q.front().y;              int tt = q.front().t;              int ss = q.front().s;              q.pop();              if (mapp[xx][yy] == '+')              {                  res = ss;                  break;              }              for (int i = 0; i < 4; i++)              {                  int x1 = xx + coor[i][0];                  int y1 = yy + coor[i][1];                  if (x1 < 0 || y1 < 0 || x1 >= M || y1 >= N)                      continue;                  if (book[x1][y1][tt])                      continue;                  if (tt == 0)                  {                      if (mapp[x1][y1] == '\*' || mapp[x1][y1] == '+')                      {                          q.push(node(x1,y1,tt,ss + 1));                          book[x1][y1][tt] = 1;                      }                  }                  else                  {                      if (mapp[x1][y1] == '\*' || mapp[x1][y1] == '+')                      {                          q.push(node(x1,y1,tt,ss + 1));                          book[x1][y1][tt] = 1;                      }                      if (mapp[x1][y1] == '#')                      {                          q.push(node(x1,y1,tt - 1,ss + 1));                          book[x1][y1][tt - 1] = 1;                      }                  }              }          }          if (res == 0)              printf("-1\n");          else              printf("%d\n",res);      }      return 0;  } |

## 百练

### 1 原题目

### 2 基础知识

### 3 解法

#### 3.1 解法1（C++已过）

本人代码：

|  |
| --- |
|  |

## 百练

### 1 原题目

### 2 基础知识

### 3 解法

#### 3.1 解法1（C++已过）

本人代码：

|  |
| --- |
|  |