## 百练 4115:鸣人和佐助

### 1 原题目

**描述** [**http://bailian.openjudge.cn/practice/4115**](http://bailian.openjudge.cn/practice/4115)

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 4116:拯救行动

### 1 原题目

**描述** [**http://bailian.openjudge.cn/practice/4116**](http://bailian.openjudge.cn/practice/4116)

公主被恶人抓走，被关押在牢房的某个地方。牢房用N\*M (N, M <= 200)的矩阵来表示。矩阵中的每项可以代表道路（@）、墙壁（#）、和守卫（x）。  
英勇的骑士（r）决定孤身一人去拯救公主（a）。我们假设拯救成功的表示是“骑士到达了公主所在的位置”。由于在通往公主所在位置的道路中可能遇到守卫，骑士一旦遇到守卫，必须杀死守卫才能继续前进。  
现假设骑士可以向上、下、左、右四个方向移动，每移动一个位置需要1个单位时间，杀死一个守卫需要花费额外的1个单位时间。同时假设骑士足够强壮，有能力杀死所有的守卫。

给定牢房矩阵，公主、骑士和守卫在矩阵中的位置，请你计算拯救行动成功需要花费最短时间。

**输入**

第一行为一个整数S，表示输入的数据的组数（多组输入）  
随后有S组数据，每组数据按如下格式输入  
1、两个整数代表N和M, (N, M <= 200).  
2、随后N行，每行有M个字符。"@"代表道路，"a"代表公主，"r"代表骑士，"x"代表守卫, "#"代表墙壁。

**输出**

如果拯救行动成功，输出一个整数，表示行动的最短时间。  
如果不可能成功，输出"Impossible"

**样例输入**

2

7 8

#@#####@

#@a#@@r@

#@@#x@@@

@@#@@#@#

#@@@##@@

@#@@@@@@

@@@@@@@@

13 40

@x@@##x@#x@x#xxxx##@#x@x@@#x#@#x#@@x@#@x

xx###x@x#@@##xx@@@#@x@@#x@xxx@@#x@#x@@x@

#@x#@x#x#@@##@@x#@xx#xxx@@x##@@@#@x@@x@x

@##x@@@x#xx#@@#xxxx#@@x@x@#@x@@@x@#@#x@#

@#xxxxx##@@x##x@xxx@@#x@x####@@@x#x##@#@

#xxx#@#x##xxxx@@#xx@@@x@xxx#@#xxx@x#####

#x@xxxx#@x@@@@##@x#xx#xxx@#xx#@#####x#@x

xx##@#@x##x##x#@x#@a#xx@##@#@##xx@#@@x@x

x#x#@x@#x#@##@xrx@x#xxxx@##x##xx#@#x@xx@

#x@@#@###x##x@x#@@#@@x@x@@xx@@@@##@@x@@x

x#xx@x###@xxx#@#x#@@###@#@##@x#@x@#@@#@@

#@#x@x#x#x###@x@@xxx####x@x##@x####xx#@x

#x#@x#x######@@#x@#xxxx#xx@@@#xx#x#####@

**样例输出**

13

7

### 2 基础知识

优先队列（解法一用到）

广度优先搜索（解法一用到）

### 3 解法

#### 3.1 解法1（C++已过）

此题因为骑士在拯救公主的过程中会遇到守卫，就会多花一个时间，而我们为了保证在广搜的时候可以找到花费时间最少的路径，所以我们要利用优先队列来维护一个最短的时间，优先队列保证每次取出来的都是花费时间最少的到达的点。

注意的是优先队列默认是大顶堆，要自己写比较函数使之成为小顶堆。

本人代码：

|  |
| --- |
| #include <iostream>  #include<algorithm>  #include <cstring>  #include <cstdio>  #include <cstdlib>  #include <cmath>  #include <vector>  #include <queue>  #include <map>  #include <set>  using namespace std;  struct node  {      int x,y,s;      bool operator<(const node& p) const      {          return s > p.s;//小顶堆      }      node() {}      node(int xx,int yy,int ss):x(xx),y(yy),s(ss) {}  };  int main()  {      int S;      char mapp[205][205];      while (scanf("%d",&S) != EOF)      {          while (S--)          {              int N,M;              scanf("%d %d",&N,&M);              getchar();              int sx,sy;              for (int i = 0; i < N; i++)              {                  for (int j = 0; j < M; j++)                  {                      scanf("%c",&mapp[i][j]);                      if (mapp[i][j] == 'r')                      {                          sx = i;                          sy = j;                      }                  }                  getchar();              }              int res = 0;              int coor[4][2] = {0,1,0,-1,-1,0,1,0};//上下左右              int book[205][205] = {0};              priority\_queue<node>q;//优先队列              q.push(node(sx,sy,0));              book[sx][sy] = 1;              //广度优先搜索              while (!q.empty())              {                  int xx = q.top().x;                  int yy = q.top().y;                  int ss = q.top().s;                  q.pop();                  if (mapp[xx][yy] == 'a')                  {                      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 >= N || y1 >= M)                          continue;                      if (book[x1][y1] || mapp[x1][y1] == '#')                          continue;                      if (mapp[x1][y1] == '@' || mapp[x1][y1] == 'a')                      {                          q.push(node(x1,y1,ss + 1));                          book[x1][y1] = 1;                      }                      if (mapp[x1][y1] == 'x')                      {                          q.push(node(x1,y1,ss + 2));                          book[x1][y1] = 1;                      }                  }              }              if (res == 0)                  printf("Impossible\n");              else                  printf("%d\n",res);          }      }      return 0;  } |

## 百练 4130:Saving Tang Monk

### 1 原题目

**描述** [**http://bailian.openjudge.cn/practice/4130**](http://bailian.openjudge.cn/practice/4130)

《西游记》（又称《猴子》）是中国文学四大名著之一。它是由吴承恩在明代撰写的。在这部小说中，孙悟空孙悟空，猪猪八戒和沙五经护送唐僧到印度去，得到神圣的佛教文字。

在旅途中，唐僧经常被恶魔俘虏。大多数恶魔都想吃唐僧来达到永生，但是一些女恶魔只是想嫁给他，因为他很帅。因此，与恶魔战斗和拯救唐僧是孙悟空要做的主要工作。

有一次，唐僧被恶魔白骨俘获。白骨头住在宫殿里，她在房间里给唐僧打了个袖口。孙悟空设法进入了宫殿。但是，要营救唐僧，孙悟空可能需要拿一些钥匙，并杀死一些蛇。

宫殿可以描述为字符矩阵。每个字符代表一个房间。在矩阵中，“ K”代表孙悟空的原始位置，“ T”代表唐僧的位置，“ S”代表其中有蛇的房间。请注意，宫殿中只有一个“ K”和一个“ T”，最多五个蛇。还有，“。” 表示房间也很干净。“＃”表示孙悟空无法进入的致命房间。

房间中可能散布着一些不同种类的钥匙，但一间房间最多只能有一把钥匙。最多有9种键。带有钥匙的房间用数字表示（从“ 1”到“ 9”）。例如，“ 1”表示带有第一类密钥的房间，“ 2”表示带有第二类密钥的房间，“ 3”表示带有第三类密钥的房间...等等。为了保存唐僧，孙悟空必须获得所有种类的钥匙（换句话说，每种钥匙至少要有一个钥匙）。

对于每一步，孙悟空都可以沿四个方向（北，西，南和东）移动到相邻的房间（致命房间除外），每一步花了他一分钟。如果他进入一个居住有活蛇的房间，则必须杀死该蛇。杀死一条蛇也花了一分钟。如果孙悟空进入一个房间，房间里有N种钥匙，那么Sun只有在已经拥有1类，2种...和N-1种钥匙的情况下，才能获得该钥匙。换句话说，孙悟空必须先获得N类密钥，然后才能获得N + 1类密钥（N> = 1）。如果孙悟空拿到了他需要的所有钥匙，然后进入唐僧被戴上手铐的房间，救援任务就完成了。如果孙悟空没有得到足够的钥匙，他仍然可以通过唐僧的房间。由于孙悟空是个不耐烦的猴子，他想尽快拯救唐僧。

**输入**

有几个测试用例。  
  
对于每种情况，第一行都包含两个整数N和M（0 <N <= 100，0 <= M <= 9），这意味着宫殿是N \* N矩阵，并且孙悟空需要M种键（种类1，种类2，...种类M）。  
  
然后是N \* N矩阵。  
  
输入以N = 0和M = 0结尾。

**输出**

对于每个测试用例，请打印保存孙唐僧所需的最短时间（以分钟为单位）。如果孙和空无法完成任务，请打印“不可能”。

**样例输入**

3 1

K.S

##1

1#T

3 1

K#T

.S#

1#.

3 2

K#T

.S.

21.

0 0

**样例输出**

5

impossible

8

### 2 基础知识

优先队列（解法一用到）

广度优先搜索（解法一用到）

### 3 解法

#### 3.1 解法1（C++已过）

这道题很有意思嘛，可真是让我蒙了又蒙，此题建议做完百练4115，4116再来做。此题简直就像是二者的结合题，可以说结合的效果非常nice，达到了1 + 1 > 2的效果。4115题就是普通广搜只不过是标记数组多开一维代表重复走每个点不同查克拉数目（此题目点可以重复走，因为走到同一个点时候查克拉数目可能不同）。4116题目则是需要用到优先队列，因为遇到守卫需要多花时间（此题没必要重复走点，遇到守卫直接过去，因为我们借助了优先队列，优先队列会每次将步数最小的先弹出来）。

本题目的钥匙数类似于查克拉数目，蛇又类似于守卫，所以需要三维标记数组记录钥匙的状态，也需要优先队列弹出步数最小的。同时，此题还多了个需要考虑的地方，那就是每条蛇的状态，蛇只需要杀死一次，但是蛇这个点可以重复走，所以还需一个数组记录蛇的状态（此题目限制了蛇最多五条），注意这些就可以做题目了。还是一样套广搜板子，细节注意下就可以了。

本人代码：

|  |
| --- |
| #include <iostream>  #include<algorithm>  #include <cstring>  #include <cstdio>  #include <cstdlib>  #include <cmath>  #include <vector>  #include <queue>  #include <stack>  #include <map>  #include <set>  using namespace std;  struct node  {      int x,y,k,s;      //记录在book[x][y][k]这个状态时哪几个蛇走过      bool snack[6];      bool operator<(const node& p) const//小顶堆      {          //返回步数最小的，步数相同则返回钥匙树多的          if (s != p.s)              return s > p.s;          return k < p.k;      }  };  //标记数组  int book[105][105][10];  int main()  {      int N,M;      char mapp[105][105];      while (scanf("%d %d",&N,&M) != EOF)      {          if (N == 0 && M == 0)              break;          getchar();          int sx,sy;          //存储蛇的编号          map<pair<int,int>,int> snakeBit;          int num = 1;          for (int i = 0; i < N; i++)          {              for (int j = 0; j < N; j++)              {                  scanf("%c",&mapp[i][j]);                  if (mapp[i][j] == 'K')                  {                      sx = i;                      sy = j;                  }                  if (mapp[i][j] == 'S')                  {                      snakeBit[ {i,j}] = num++;                  }              }              getchar();          }          int res = 0;          int coor[4][2] = {0,1,0,-1,-1,0,1,0};//上下左右          memset(book,0,sizeof(book));          priority\_queue<node>q;          node start;          //初始化设置          start.x = sx;          start.y = sy;          start.k = start.s = 0;          memset(start.snack,0,sizeof(start.snack));          q.push(start);          book[sx][sy][0] = 1;          //广度优先搜索          while (!q.empty())          {              node a = q.top();              q.pop();              //救出唐僧的条件              if (mapp[a.x][a.y] == 'T' && a.k == M)              {                  res = a.s;                  break;              }              for (int i = 0; i < 4; i++)              {                  node b;                  b.x = a.x + coor[i][0];                  b.y = a.y + coor[i][1];                  b.k = a.k;                  //越界                  if (b.x < 0 || b.y < 0 || b.x >= N || b.y >= N)                      continue;                  //已遍历和走不过的地方                  if (book[b.x][b.y][a.k] || mapp[b.x][b.y] == '#')                      continue;                  for (int j = 0; j < 6; j++)                      b.snack[j] = a.snack[j];                  b.s = a.s + 1;                  //蛇没走过的需要额外耗费时间杀死，已经杀死过的就不要杀死第二次（蛇只有一条命）                  if (mapp[b.x][b.y] == 'S' && b.snack[snakeBit[ {b.x,b.y}]] == 0)                  {                      b.s++;                      b.snack[snakeBit[ {b.x,b.y}]] = 1;                  }                  //找钥匙                  else if (mapp[b.x][b.y] >= '1' && mapp[b.x][b.y] <= M + '0')                  {                      int temp = mapp[b.x][b.y] - '0';                      if (a.k != M && temp == a.k + 1)                      {                          b.k++;                      }                  }                  book[b.x][b.y][b.k] = 1;                  q.push(b);              }          }          if (res == 0)              printf("impossible\n");          else              printf("%d\n",res);      }      return 0;  } |

## 百练 4124:海贼王之伟大航路

### 1 原题目

**描述** [**http://bailian.openjudge.cn/practice/4124/**](http://bailian.openjudge.cn/practice/4124/)

“我是要成为海贼王的男人！”，路飞一边喊着这样的口号，一边和他的伙伴们一起踏上了伟大航路的艰险历程。



路飞他们伟大航路行程的起点是罗格镇，终点是拉夫德鲁（那里藏匿着“唯一的大秘宝”——ONE PIECE）。而航程中间，则是各式各样的岛屿。

因为伟大航路上的气候十分异常，所以来往任意两个岛屿之间的时间差别很大，从A岛到B岛可能需要1天，而从B岛到A岛则可能需要1年。当然，任意两个岛之间的航行时间虽然差别很大，但都是已知的。

现在假设路飞一行从罗格镇（起点）出发，遍历伟大航路中间所有的岛屿（但是已经经过的岛屿不能再次经过），最后到达拉夫德鲁（终点）。假设他们在岛上不作任何的停留，请问，他们最少需要花费多少时间才能到达终点？

**输入**

输入数据包含多行。  
第一行包含一个整数N(2 < N ≤ 16)，代表伟大航路上一共有N个岛屿（包含起点的罗格镇和终点的拉夫德鲁）。其中，起点的编号为1，终点的编号为N。  
之后的N行每一行包含N个整数，其中，第i(1 ≤ i ≤ N)行的第j(1 ≤ j ≤ N)个整数代表从第i个岛屿出发到第j个岛屿需要的时间t(0 < t < 10000)。第i行第i个整数为0。

**输出**

输出为一个整数，代表路飞一行从起点遍历所有中间岛屿（不重复）之后到达终点所需要的最少的时间。

**样例输入**

样例输入1：

4

0 10 20 999

5 0 90 30

99 50 0 10

999 1 2 0

样例输入2：

5

0 18 13 98 8

89 0 45 78 43

22 38 0 96 12

68 19 29 0 52

95 83 21 24 0

**样例输出**

样例输出1：

100

样例输出2：

137

**提示**

提示：  
对于样例输入1：路飞选择从起点岛屿1出发，依次经过岛屿3，岛屿2，最后到达终点岛屿4。花费时间为20+50+30=100。  
对于样例输入2：可能的路径及总时间为：  
1,2,3,4,5: 18+45+96+52=211  
1,2,4,3,5: 18+78+29+12=137  
1,3,2,4,5: 13+38+78+52=181  
1,3,4,2,5: 13+96+19+43=171  
1,4,2,3,5: 98+19+45+12=174  
1,4,3,2,5: 98+29+38+43=208  
所以最短的时间花费为137  
单纯的枚举在N=16时需要14!次运算，一定会超时。

### 2 基础知识

记忆化深度优先搜索（解法一用到）

二进制（解法一用到）

### 3 解法

#### 3.1 解法1（C++已过）

此题目其实深搜部分没有多难，难就难在看你是否能利用二进制的知识来解题，岛的数量恒定的，所以我们可以利用二进制来表示某个岛是否被访问过，以及为了方便剪枝或者说更快的找到最优解我们可以利用二进制来保存访问过路径的状态，这样就可以快速解题。

岛的二进制表示如下：第一个岛的二进制就是10，十进制值就是2，第一个岛的二进制就是100，十进制值就是4

本人代码：

|  |
| --- |
| #include <iostream>  #include<algorithm>  #include <cstring>  #include <cstdio>  #include <cstdlib>  #include <cmath>  #include <vector>  #include <queue>  #include <stack>  #include <map>  #include <set>  using namespace std;  #define INF 0X3f3f3f3f  int mapp[20][20];//存储岛与岛之间的时间  int book[20];//标记岛是否走过  int N;  int ans;//结果  //下面这三个变量其实都是利用的二进制  int cur;//当前的状态（二进制表示第i个岛走过第i位就是1否则是0）  int record[20][1<<15];//record[i][j]记录的是从j状态（i状态就是指哪些岛遍历过，二进制表示的）到i岛的最短时间  //每个岛的状态标记（第一个岛的二进制就是10，十进制值就是2，第一个岛的二进制就是100，十进制值就是4）  //如果2 + 4 = 6，二进制表示就是110，代表的就是第一个岛和第二个岛走过了（从后数，从0开始数）  int weight[20];  //pre是之前遍历过岛的数量，now是当前遍历岛的坐标，time就是遍历的时间  void dfs(int pre,int now,int time)  {      //now是当前点坐标，pre代表之前遍历过岛的数量      //岛的数量N包含了起点和终点，当前点now到终点就是两个点      //所以判定条件是pre + 2 == N      if (pre + 2 == N)      {          if (time + mapp[now][N] < ans)              ans = time + mapp[now][N];          return ;      }      //剪枝      if (time > ans)          return ;      for (int i = 2; i < N; i++)      {          //剪枝          if (book[i])              continue;          //判断状态是否之前出现过更小的，也是剪枝          if (record[i][cur + weight[i]] <= time + mapp[now][i])              continue;          cur += weight[i];//状态更新          book[i] = 1;//标记          record[i][cur] = time + mapp[now][i];//更新record          dfs(pre + 1,i,time + mapp[now][i]);          cur -= weight[i];          book[i] = 0;      }      return ;  }  int main()  {      //初始化      for (int i = 1; i <= 16; i++)      {          weight[i] = 1 << i;      }      while(scanf("%d",&N) != EOF)      {          for (int i = 1; i <= N; i++)          {              for (int j = 1; j <= N; j++)              {                  scanf("%d",&mapp[i][j]);              }          }          //初始化          ans = INF;          cur = 0;          for (int i = 0; i < 20; i++)              for (int j = 0; j < 1<<15; j++)                  record[i][j] = INF;          memset(book,0,sizeof(book));          dfs(0,1,0);          printf("%d\n",ans);      }      return 0;  } |

## 百练 1321:棋盘问题

### 1 原题目

**描述** [**http://bailian.openjudge.cn/practice/1321**](http://bailian.openjudge.cn/practice/1321)

在一个给定形状的棋盘（形状可能是不规则的）上面摆放棋子，棋子没有区别。要求摆放时任意的两个棋子不能放在棋盘中的同一行或者同一列，请编程求解对于给定形状和大小的棋盘，摆放k个棋子的所有可行的摆放方案C。

**输入**

输入含有多组测试数据。  
每组数据的第一行是两个正整数，n k，用一个空格隔开，表示了将在一个n\*n的矩阵内描述棋盘，以及摆放棋子的数目。 n <= 8 , k <= n  
当为-1 -1时表示输入结束。  
随后的n行描述了棋盘的形状：每行有n个字符，其中 # 表示棋盘区域， . 表示空白区域（数据保证不出现多余的空白行或者空白列）。

**输出**

对于每一组数据，给出一行输出，输出摆放的方案数目C （数据保证C<2^31）。

**样例输入**

2 1

#.

.#

4 4

...#

..#.

.#..

#...

-1 -1

**样例输出**

2

1

### 2 基础知识

深度优先搜索（解法一用到）

### 3 解法

#### 3.1 解法1（C++已过）

此题目吧多少有点像N皇后问题，N皇后是对角线也不允许出现重复棋子，本题目对角线可以出现，只限制了同一行行列不能出现两个棋子。此题目的解法就是标记数组标记列是否走过，对于行，我们只需此行走过后下一步的深搜从此行下一行开始（避免当前行继续下棋），就可以解决问题了。

本人代码：

|  |
| --- |
| #include <iostream>  #include<algorithm>  #include <cstring>  #include <queue>  #include <stack>  #include <cstdio>  #include <cstdlib>  #include <cmath>  #include <vector>  using namespace std;  char mapp[10][10];  int col[10];//标记此列是否有棋子  int n,k,res;  //深度优先搜索  //x代表从哪行开始找，num代表棋子个数  void dfs(int x,int num)  {      if (num == k + 1)      {          res++;          return ;      }      for (int i = x; i < n; i++)      {          for (int j = 0; j < n; j++)          {              if (!col[j])              {                  if (mapp[i][j] == '.')                      continue;                  col[j] = 1;                  dfs(i + 1,num + 1);                  col[j] = 0;              }          }      }      return ;  }  int main()  {      while (scanf("%d %d",&n,&k) != EOF)      {          if (n == -1 && k == -1)              break;          memset(col,0,sizeof(col));          res = 0;          getchar();          for (int i = 0; i < n; i++)          {              for (int j = 0; j < n; j++)              {                  scanf("%c",&mapp[i][j]);              }              getchar();          }          dfs(0,1);          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：

|  |
| --- |
|  |