滚动广搜

我们知道朴素广搜的空间浪费很严重，原因很简单，朴素广搜将从初始节点开始可以达到的所有状态全都记录下来了，而这样需要消耗的空间代价是指数级的。所有使用广搜时，空间就显得很宝贵。于是我们开始寻找对状态储存的优化，以减小对空间的消耗。一类典型的题目（求迷宫中到达某一个位置的最少步数），可以用到一种思想———滚动广搜。

滚动广搜的思想就是用到两个表来储存前一个阶段可以达到的所有状态和当前阶段可以达到的所有状态。因为这一步能到达的所有位置只由上一步的位置决定，而所有的位置的集合规模是 n*m（分别表示迷宫的长和宽）。我们只需要开两个大小为 n*m 的表就可以保证能记录所有可能达到的位置了。每次取出前一阶段所有状态节点，并按照产生式规则对其扩展并存入当前阶段的表内。然后通过 now:=1-now 不断切换前一阶段与当前阶段，（注意：每次要对当前阶段的表和标记数组进行清空）。

1. 下面给出滚动广搜的数据结构:

```
type rec=record{节点的描述，包括横纵、坐标，根据题目不同可适当添加}
              x,y:integer;
            end;
var q:array[0..1,1..maxn*maxm] of rec;  {0、1 分别表示当前阶段和前一个阶段}
    r:array[0..1] of longint; {两个表对应的尾指针}
    used:array[1..maxn,1..maxm] of Boolean;{标记(u,v)是否以可到达}
```

2. 以下是滚动广搜的核心算法部分。

```
procedure rollbfs;
  var i,f,u,v:integer;
  begin
    now:=0;r[now]:=1;q[now,1].x:=fx;q[now,1].y:=fy;{初始化，将起点位置入队}
    repeat
      now:=1-now; r[now]:=0;
      fillchar(used,sizeof(used),0);
      {切换到当前阶段，置队列为空，将 used 标记数组清空}
      for f:=1 to r[1-now] do{取出前一阶段的所有状态节点并对其进行扩展}
        begin
          for i:=1 to maxway do{枚举对于该位置所有可能的扩展方向}
            begin
              u:=q[1-now,f].x+a[i];{a,b 分别为横、纵方向上的增量}
              v:=q[1-now,f].y+b[i];
              if (map[u,v]可达到) and(not(used[u,v]){(u,v)未被标记过} then
                begin
                  inc(r[now]); if (r[now]>n*m) then exit;
                  q[now,r[now]].x:=u;,
                  q[now,r[now]].y:=v;
                  used[u,v]:=true;
                  if (u=ex)and(v=ey) then{如果达到终点则打应并结束算法}
                    begin writeln(time);close(output);halt; end;
```

```
                end;
            end;
        end;
    until 0>1;
end;
```

参考题目：《穿越封锁线》《最后的战犯》《SEARCH》《迷宫》

附：

最后的战犯

题目描述：话说 Lucky 和 Feli 以 3721 部队为诱饵，歼灭了大批日军。但顽固的日军军官小犬蠢一狼（以下简称小犬）在全军覆灭之后仍然不肯认输。他躲在山区的一个岩洞中，等待日军的救援部队。他妄图凭借复杂的地形躲避我军的追踪。于是，总部派出足智多谋的 Feli 前往岩洞搜寻小犬。Feli 来到岩洞入口，发现岩洞其实是一个巨大的迷宫。迷宫地形极为复杂，为一个正方形，其中布满了障碍物。迷宫可以分为 N*N（2≤N≤100）个区域，每个区域或者是空地，或者是不可逾越的障碍物。小犬就躲藏在其中某一个区域内。由于小犬已经忍受了几天的饥饿，Feli 进入迷宫时他已经失去思维处于迷乱状态。小犬每秒钟只会沿着他的方向直线前进，如果遇到障碍物或者迷宫边界，他会立刻向右转 90 度（不会花去时间），继续沿直线前进（初始方向向北）。Feli 每秒钟可以自主决定往哪个方向走。如果同一时刻 Feli 与小犬位于同一个区域，或者相邻的区域（非对角线相邻），Feli 可以立刻将小犬抓住。Feli 本来打算先确定小犬的位置，然后沿最短路线抓住他，但是 Feli 前进时小犬同时也在移动，就不能采取这种方法了。请你帮助 Feli 确定一种方案，使 Feli 抓获小犬所用的时间最短。

数据说明：

输入数据第一行是一个整数 N。以下 N 行每行 N 个字符，“*”表示岩洞中的障碍物，“.”表示空地，“J”表示小犬（一开始他会向北走），“F”表示 Feli。上北下南左西右东。

输出数据仅一行，如果 Feli 能抓到小犬，那么输出所需的最短时间，如果 Feli 抓不到小犬，那么这个最后的日本战犯将在岩洞中饿死（因为 Feli 将在离开的时候封闭岩洞的所有出口），此时输出“No solution.”，不要输出引号。

样例输入(maze.in)

```
3
F*J
.*.
...
```

样例输出(maze.out)

```
3
```

源程序：

```
program maze;
type rec=record x,y:integer; end;
const a:array[1..4] of integer=(0,1,0,-1);
```

```
b:array[1..4] of integer=(-1,0,1,0);
var q:array[0..1,1..10000] of rec;
    r:array[0..1] of longint;
    map:array[0..101,0..101] of integer;
    used:array[0..101,0..101] of boolean;
    i,j,k,time,p,m,fx,fy,ex,ey,now:longint;
    d:char;

procedure try;
  var j:integer;
  begin
    j:=0;
    ex:=ex+a[p];
    ey:=ey+b[p];
    while map[ex,ey]=0 do
      begin
        inc(j);if j=5 then begin writeln('No solution.');close(output);halt; end;
        ex:=ex-a[p];
        ey:=ey-b[p];
        inc(p);if p=5 then p:=1;
        ex:=ex+a[p];
        ey:=ey+b[p];
      end;
  end;

procedure rollbfs;
  var i,f,u,v:integer;
  begin
    now:=0;r[now]:=1;q[now,1].x:=fx;q[now,1].y:=fy;p:=1;
    repeat
      inc(time);try;
      now:=1-now; r[now]:=0; fillchar(used,sizeof(used),0);
      for f:=1 to r[1-now] do
        begin
          for i:=1 to 4 do
            begin
              u:=q[1-now,f].x+a[i];
              v:=q[1-now,f].y+b[i];
              if (map[u,v]=1) and (not(used[u,v])) then
                begin
                  inc(r[now]); if (r[now]>=10000) then exit;
                  q[now,r[now]].x:=u;
                  q[now,r[now]].y:=v;
                  used[u,v]:=true;
```

```
                if ((u=ex)and(v=ey)) or ((u+1=ex)and(v=ey))
                    or ((u-1=ex)and(v=ey)) or ((u=ex)and(v+1=ey))
                    or ((u=ex)and(v-1=ey)) then
                  begin writeln(time);close(output);halt; end;
              end;
          end;
      end;
  until 0>1;
end;

begin
  assign(input,'maze.in');reset(input);
  assign(output,'maze.out');rewrite(output);
  readln(m);
  fillchar(map,sizeof(map),0);
  for i:=1 to m do
    begin
      for j:=1 to m do
        begin
          read(d);
          case d of
              '.':map[j,i]:=1;
              '*':map[j,i]:=0;
              'F':begin map[j,i]:=1;fx:=j;fy:=i; end;
              'J':begin map[j,i]:=1;ex:=j;ey:=i; end;
            end;
        end;readln;
    end;
  close(input);

  if (map[ex+1,ey]=0) and (map[ex-1,ey]=0) and
    (map[ex,ey+1]=0) and (map[ex,ey-1]=0) then
    begin writeln('No solution.');close(output);halt;end;
  rollbfs;
  writeln('No solution.');
  close(output);
end.
```