基于迷宫问题的回溯法求解及算法实现

毕智超
（陕西职业技术学院，陕西西安，710100）

摘要：本文将利用迷宫问题作为实例，首先给出了走迷宫的问题描述；其次对网格状迷宫利用二维数组进行存储，模拟出走迷宫时行进方向的回溯法探测过程；最后利用 C++ 编程语言给出了解决迷宫问题的递归算法和非递归算法。最终在迷宫中找到一条最佳路径。

关键词：迷宫问题；二维数组；回溯法；最佳路径

Solution for backtracking based on maze problem and algorithm realization

Bi Zhichao
(Shaanxi Vocational &Technical College,xi'an,710100,China)

Abstract: This paper will use the maze problem as an example, the first description of maze problem is given; secondly to store grid maze using two-dimensional array, analog backtracking maze detection process from the direction of travel;finally,the C++ programming language provides a recursive algorithm to solve the maze problem and non-recursive algorithm using. Finally found an optimal path in a maze.

Keywords: Maze problem;Two-dimensional array;Backtracking;An optimal path

0　引言

回溯法也称为试探法。这种方法是按照某种次序将问题的候选解逐一进行列举和检验。本文将利用迷宫问题作为实例，讨论回溯法的求解过程。迷宫问题的提法如下：把一只小白鼠从一个无顶盖的大盒子（迷宫）的入口处赶进迷宫。迷宫中设置了很多墙壁，对前进方向形成了多处障碍。在迷宫的唯一出口处放置了鼠粮，吸引老鼠在迷宫中寻找通路以到达出口。如果从迷宫的入口到达出口，途中不出现行进方向错误，则得到一条最佳路线。我们利用回溯法可获得迷宫从入口到出口的最佳路线。

1　迷宫数据结构的设计

针对上述问题的提出，我们可以给出相应设计思想。首先对问题进行抽象给出数学模型。为此，用一个二维数组 maze[m+2][p+2] 来表示迷宫，当数组元素 maze[i][j]=1 时，表示该位置是墙壁，不能通行；当 maze[i][j]=0 时，表示该位置是通路。1 ≤ i ≤ m，1 ≤ j ≤ p。数组的第 0 行，第 m+1 行，第 0 列和第 p+1 列是迷宫的围墙。如图 1 所示。

图 1　用二维数组表示的迷宫

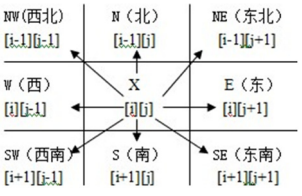

图 2 可能的前进方向

2　算法设计思想

在求解迷宫问题的过程中，当沿着某一条路径一步步走向出口但发现进入死胡同走不通时，就回溯一步或多步，寻找其他可走的路径。这就需要回溯。老鼠在迷宫中任一时刻的位置可用数组行下标 i 和列下标 j 表示。从 maze[i][j] 出发，可能的前进方向有 8 个，按顺时针方向为 N[i-1][j]，NE[i-1][j+1]，E[i][j+1]，SE[i+1][j+1]，S[i+1][j]，SW[i+1][j-1]，W[i][j-1],NW[i-1][j-1]。如图 2 所示。

设位置 [i][j] 标记为 X，它实际是一系列交通路口。X 周围有 8 个前进方向，分别代表 8 个前进位置。如果某一方向是 0 值，表示该方向有路可通，否则表示该方向已堵死。为了有效地选择下一位置，可以将从位置 [i][j] 出发可能的前进方向预先定义在一个表内。参看表 1，我们称该表为前进方向表（附上程序清单 1- 前进方向表的结构化定义），它给出向各个方向的偏移量。

程序清单 1 前进方向表的结构化定义

```
struct offsets
{// 位置在直角坐标下的偏移
int a,b;    //a，b 是 x，y 方向的偏移
char *dir;    // 指针 dir 指示方向
};offsets move[8];  // 所有方向的偏移表
```

当在迷宫中向前试探时，可根据表 1 所示的前进方向表，选择某一个前进方向向前试探。如果该前进方向走不通，则在前进路径上回退一步，再尝试其他的允许方向。

为了防止重走原路，另外设置一个标志矩阵 mark[m+2][p+2]，它的所有元素都初始化为 0。一旦行进到迷宫的某个位置 [i][j]，则将 mark[i][j] 置为 1。下次这个位置就不能再走了。

3　算法实现

回溯法解决迷宫问题的递归算法参见程序清单 2。

程序清单 2 解决迷宫问题的递归算法实现

```
int Seekpath(int x,int y)
{/* 从迷宫某一位置 [i][j] 开始,寻找通向出口 [m][p] 的一条路径。如果找到,则函数返回 1。否则函数返回 0。试探的出发点为 [1][1]。 */
int i,g,h;char *d;
if(x==m&&y==p)return 1;// 已到达出口,函数返回 1
for(i=0;i<8;i++)
{// 依次按每一个方向寻找通向出口的路径
g=x+move[i].a; h=y+move[i].b; d=move[i].dir;
if(maze[g][h]==0&&mark[g][h]==0)
{// 下一个位置可通,试探该方向
mark[g][h]=1;
if(Seekpath(g,h))
{// 从此位置递归试探
cout<<"("<<g<<","<<h<<"),"<<"Direction"<<dir<<",";
return 1;
}
}
// 回溯,换一个方向再试探通向出口的路径
}
if(x==1&&y==1)
cout<<"no path in maze"<<endl;
return 0;
}
```

为了将递归算法改为非递归算法,需要使用一个堆栈来存储在试探的过程中所走过的路径。一旦需要回退,可以从栈中取得刚才走过位置的坐标和前进方向。栈中需保存一系列三元组以记录这些信息,这些三元组的结构定义如下 :

程序清单 3 栈中的三元组结构

```
struct items
{
int x,y,dir;// 位置和前进方向序号
};
```

当在迷宫中向前试探时,可能同时存在几个允许的前进方向。我们利用三元组记下当前位置和上一步前进的方向,然后根据表 1 所示的前进方向表,选择某一个允许的前进方向前进一步,并将活动记录进栈,以记下前进路径。如果该前进方向走不通,则将位于栈顶的活动记录退栈,以在前进路径上回退一步,再尝试其他的允许方向。如果栈空则表示已经回退到开始位置。

因为数组 maze 中的每个位置最多只访问一次,故用来记录搜索路径的栈的大小一般不超过 m*p。若设栈的大小为 m*p,一般不存在栈满问题。迷宫问题的非递归算法如程序清单 4 所示。

程序清单 4 解决迷宫问题的非递归算法实现

```
void path(int m,int p)
{/* 算法输出迷宫 maze 中的一条路径。作为围墙, maze[0][i]=maze[m+1][i]=maze[j][0]
=maze[j][p+1]=1, 0 ≤ i ≤ p+1, 0 ≤ j ≤ m+1。在算法中用到一个栈 st 的重载函数 <<,功能是顺序输出栈中的各个元素。*/
int i,j,d,g,h; mark[1][0]=1;
Stack<items>st(m*p);items tmp;
tmp.x=1;tmp.y=0;tmp.dir=2;
st.Push(tmp);
while(st.IsEmpty()==false)
{
st.Pop(tmp);
i=tmp.x;j=tmp.y;d=tmp.dir;
while(d<8)
{
g=i+move[d].a;h=j+move[d].b;
if(g==m&&h==p)
{
cout<<st;
cout<<m<<" "<<p<<endl;
return;
}
if(maze[g][h]==0&&mark[g][h]==0)
{
mark[g][h]=1;
tmp.x=i;tmp.y=j;temp.dir=d;
st.Push(tmp);
i=g;j=h;d=0;
}
else d++;
}
}
cout<<"no path in maze"<<endl;
}
```

此程序的内部循环的循环次数是固定的,时间复杂度为 O(1);假设 maze 数组中零元素的个数为 n,那么最多有 n 个位置可以标志,而 n ≤ mp,因此在外层循环的控制下,内部循环的计算时间为 O(m*p)。

4 结束语

综上所述,用回溯法求解迷宫问题时常常使用递归方法进行试探,或使用栈帮助向前试探和回溯。本文中用递归和非递归两种方法诠释了回溯法解迷宫问题的算法设计思想。不仅迷宫问题,许多复杂的问题,规模较大的问题都可以使用回溯法求解。这对于今后其他问题的研究会有很大帮助。

参考文献

[1] 遇娜 . 基于迷宫问题的算法新解 [J]. 渭南师范学院学报 , 2011, 26(2):66-68.

[2] 崔兆顺 . 基于遗传规划的迷宫问题高效求解 [J]. 制造业自动化 , 2011, 33(1), :194-196.

[3] 邓延安 . 机器鼠走迷宫的优化路径算法及实践 [J]. 芜湖职业技术学院学报 , 2007, 9(2):37-39.

[4] 周蕾 . 改良填充法实现和解决迷宫问题 [J]. 电脑知识与技术 , 2007, 13:186-188.

[5] 胡小兵,黄席樾 . 蚁群算法在迷宫最优路径问题中的应用 [J] . 计算机仿真 .2005, 22(4): 115- 116.

作者简介

毕智超(1983—),男,陕西三原人,讲师,主要研究方向 :软件工程、算法分析。

表 1 前进方向表 move

move[q].dir	move[q].a	move[q].b
“N”	-1	0
“NE”	-1	1
“E”	0	1
“SE”	1	1
“S”	1	0
“SW”	1	-1
“W”	0	-1
“NW”	-1	-1