



中南大學
CENTRAL SOUTH UNIVERSITY

算法设计与分析

实验报告(搜索)

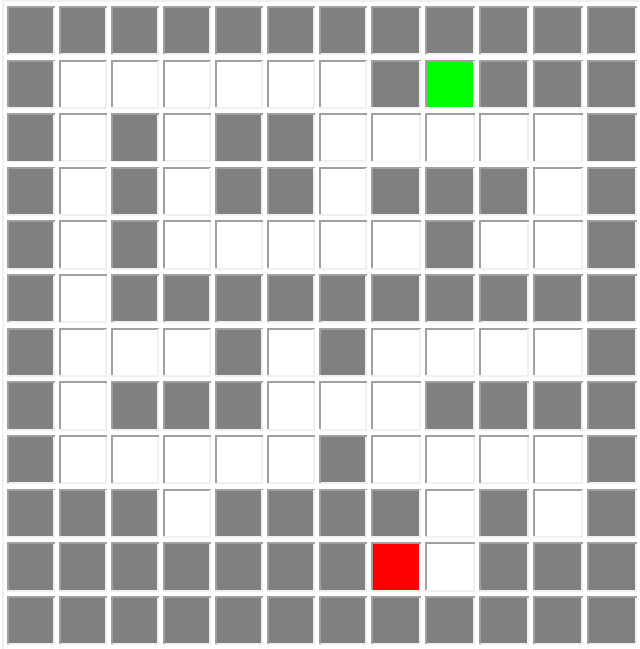
学生姓名	杨凯楠
学 号	8208201004
专业班级	信息安全 2002 班
指导教师	石峰
学 院	计算机学院
完成时间	2021 年 11 月 18 日

一、 实验目的：熟练掌握搜索算法

二、 实验内容

1.电子老鼠闯迷宫

如下图 12×12 方格图，找出一条自入口（2，9）到出口（11，8）的最短路径。



测试数据如下：

12 10 7 1 8

2. 分酒问题

有一酒瓶装有 8 斤酒，没有量器，只有分别装 5 斤和 3 斤的空酒瓶。设计一程序将 8 斤酒分成两个 4 斤，并以最少的步骤给出答案。

三、 具体设计

1. 电子老鼠闯迷宫

分析：本题的解题思路是将其等效为广度优先遍历，a[][]表示方格，save[][]表示【当前已经遍历过的点的序号】【横纵坐标】，path[]【内为当前遍历序号】，存储的内容为上一序号。如果当前的位置可以走的话就尾加 1，把当前位置的父结点加入队列；当现在的父结点已经没有子结点的话，就把头加 1，把上一个已经没有子结点的父结点退出队列。

只要是走过的路线都要把它赋值为 1（障碍），防止有重复的路线，而且先到某一个路线的一定是最快最优的。

详细设计：

```
1. #include<iostream>
2. #include<cstdio>
```

```

3. using namespace std;
4. const int mov[4][2] = { {0,1 }, {0, -1},{ -1, 0},{ 1, 0} };
5. int size = 12;
6. int a[12][12]; //方格状况
7. int save[12 * 12][2]; // [当前已经遍历过的点的序号][横纵坐标]
8. int path[12 * 12]; // 【内为当前遍历序号】存储的内容为上一序号
9. int a1, b1; //出发点坐标
10. int a2, b2; //终点坐标
11. int countlen = 0;
12. bool judge(int i, int j) {
13.     if (a[i][j] == 1)
14.         return false;
15.     return true;
16. }
17. void coutmy(int a) {
18.     if (a == 1)
19.         return;
20.     countlen++;
21.     coutmy(path[a]);
22.     cout << "-->" << "(" << save[a][0]+1 << "," << save[a][1]+1 << ")";
23. }
24. void solution() {
25.     int sign1=0, sign2=1; //sign1 为空处遍历过的序号, sign2 为正确路径序号
        path[]
26.     path[sign2] = sign1; //第一个遍历的前级肯定为出发点
27.     save[1][0] = a1; save[1][1] = b1;
28.     a[a1][b1] = 1; //出发点肯定走完了设为 1
29.     do {
30.         sign1++;
31.         for (int i = 0; i < 4; i++) {
32.             if (judge(save[sign1][0] + mov[i][0], save[sign1][1] + mo
                v[i][1])) {
33.                 sign2++;
34.                 path[sign2] = sign1;
35.                 save[sign2][0] = save[sign1][0] + mov[i][0];
36.                 save[sign2][1] = save[sign1][1] + mov[i][1];
37.                 a[save[sign2][0]][save[sign2][1]] = 1;
38.             }
39.             if (save[sign2][0] == a2-1 && save[sign2][1] == b2-1) {
40.                 int count = 0;
41.                 cout << "(" << a1 << "," << b1 << ")";
42.                 coutmy(sign2);
43.                 cout << "步长为" << countlen;
44.                 return;

```

```

45.         }
46.     }
47. } while (sign1 < sign2);
48.
49. }
50. int main(void) {
51.     int i, j;
52.
53.     cin >> a1 >> b1 >> a2 >> b2;
54.
55.     for (i = 0; i < 12; i++)
56.         for (j = 0; j < 12; j++)
57.             cin >> a[i][j];
58.     solution();
59. }

```

运行结果:

```

Microsoft Visual Studio 调试控制台
1 0 1 0 1 1 0 0 0 0 0 1
1 0 1 0 1 1 0 1 1 1 0 1
1 0 1 0 0 0 0 0 1 0 0 1
1 0 1 1 1 1 1 1 1 1 1 1
1 0 0 0 1 0 1 0 0 0 0 1
1 0 1 1 1 0 0 0 1 1 1 1
1 0 0 0 0 0 1 0 0 0 0 1
1 1 1 0 1 1 1 1 0 1 0 1
1 1 1 1 1 1 1 0 0 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
(2, 9) --> (3, 9) --> (3, 8) --> (3, 7) --> (2, 7) --> (2, 6) --> (2, 5) --> (2, 4) -->
(2, 3) --> (2, 2) --> (3, 2) --> (4, 2) --> (5, 2) --> (6, 2) --> (7, 2) --> (8, 2) -->
(9, 2) --> (9, 3) --> (9, 4) --> (9, 5) --> (9, 6) --> (8, 6) --> (8, 7) --> (8, 8) -->
(9, 8) --> (9, 9) --> (10, 9) --> (11, 9) --> (11, 8) 步长为28
D:\杨凯楠\ykn\算法\实验报告\algorithm\Debug\algorithm.exe (进程
63908) 已退出, 代码为 0。
要在调试停止时自动关闭控制台, 请启用“工具”->“选项”->“调试”
->“调试停止时自动关闭控制台”。
按任意键关闭此窗口. . .

```

调试分析: 本题的一个难点是路径的保存, 这也耗费了我不少时间, 路径的保存大致思路是

建立一个数组，让数组记录前一个位置的坐标。在到达终点输出路径时用递归的方法从头到尾输出。

2. 分酒问题

有一酒瓶装有 8 斤酒，没有量器，只有分别装 5 斤和 3 斤的空酒瓶。设计一程序将 8 斤酒分成两个 4 斤，并以最少的步骤给出答案。

分析：分酒问题采用广度优先遍历，将每一种可能的倒酒方式都全部考虑他下一步倒酒的所有情形，直到最先达到目标情况。

概要设计：

```
1. int captain[] = { 8,5,3 };//容器标准容量
2. int des[3];
3. int cou = 0;
4. int step = 0;
5. struct state {
6.     int now[3];//现在各杯中酒量;
7.     int xulie;//在数组中位置
8.     int qiandao;//实现这一步骤的前一步骤，前导
9. };
10. queue<state> q;//广度优先搜索队列
11. vector<state> p;//顺序数组
12. //判断是否可以进一步进行倒酒动作,i 向 j 里面倒酒。
13. bool check(int now[], int i, int j) ;
14.
15. //输出函数
16. void out(int a);
17.
18. //广度优先遍历，可以求得最优解
19. void bfs(state s);
```

详细设计：

```
1. bool check(int now[], int i, int j) {
2.     if (now[i] == 0 || now[j] == captain[j])
3.         return false;
4.     return true;
5. }
6. //输出函数
7. void out(int a) {
8.     step += 1;
9.     if (p[a].xulie == 0) {
10.         cout << p[a].now[0] << " " << p[a].now[1] << " " << p[a].now[
11.             2] << endl;
12.         return;
13.     }
```

```

13.     out(p[a].qiandao);
14.     cout << p[a].now[0] << " " << p[a].now[1] << " " << p[a].now[2] <
        < endl;
15. }
16. //广度优先遍历，可以求得最优解
17. void bfs(state s) {
18.     state tmp, tempr;
19.     s.xulie = 0;
20.     s.qiandao = 0;
21.     p.push_back(s);
22.     q.push(s);
23.     while (!q.empty()) {
24.         bool flag = true;
25.         tmp = q.front();
26.         for (int i = 0; i < 3; i++) {
27.             if (tmp.now[i] != des[i])
28.                 flag = false;
29.         }
30.         if (flag){
31.             cout << "倒酒步骤为: " << endl;
32.             out(tmp.xulie);
33.             cout << "共用" << step-1 << "步" << endl;
34.             return;
35.         }
36.         for (int i = 0; i < 3; i++) {
37.             if (tmp.now[i] > 0) {
38.                 for (int j = 0; j < 3; j++) {
39.                     if (i != j && check(tmp.now, i, j)) {
40.                         int tmpi = tmp.now[i], tmpj = tmp.now[j];
41.                         tempr = tmp;
42.                         //daojiu
43.                         tmpj += tmp.now[i];
44.                         tmpi -= tmp.now[i];
45.                         //panduan
46.                         if (tmpi == 0 && tmpj == captain[j]) {//ikong
                            ,
47.                             tempr.now[i] = tmpi;
48.                             tempr.now[j] = tmpj;
49.                             cou++;
50.                             tempr.xulie = cou;
51.                             tempr.qiandao = tmp.xulie;
52.                             p.push_back(tempr);
53.                             q.push(tempr);
54.                         }
                    }
                }
            }
        }
    }
}

```

```

55.         else if (tmpi == 0 && tmpj < captain[j]) {
56.             tempr.now[i] = tmpi;
57.             tempr.now[j] = tmpj;
58.             cou++;
59.             tempr.xulie = cou;
60.             tempr.qiandao = tmp.xulie;
61.             p.push_back(tempr);
62.             q.push(tempr);
63.         }
64.         //ibukong
65.         tmpi = tmp.now[i], tmpj = tmp.now[j];
66.         tmpi -= (captain[j] - tmpj);
67.         tmpj = captain[j];
68.         if (tmpi > 0 && tmpi < captain[i]) {
69.             tempr.now[i] = tmpi;
70.             tempr.now[j] = tmpj;
71.             cou++;
72.             tempr.xulie = cou;
73.             tempr.qiandao = tmp.xulie;
74.             p.push_back(tempr);
75.             q.push(tempr);
76.         }
77.     }
78. }
79. }
80. }
81.     q.pop();
82. }
83. }
84.
85.
86.
87.
88. int main() {
89.     state s;
90.     for (int i = 0; i < 3; i++) {
91.         cin >> des[i];
92.     }
93.     s.now[0] = 8;
94.     s.now[1] = 0;
95.     s.now[2] = 0;
96.     s.xulie = 0;
97.     s.qiandao = 0;
98.     bfs(s);

```

```
99.     return 0;  
100. }
```

运行结果:

。



```
Microsoft Visual Studio 调试控制台  
4 4 0  
倒酒步骤为:  
8 0 0  
3 5 0  
3 2 3  
6 2 0  
6 0 2  
1 5 2  
1 4 3  
4 4 0  
共用7步  
D:\杨凯楠\ykn\算法\实验报告\algorithm\Debug
```

调试分析:

同样，本道题在记录倒酒路径的问题上耗费了我不少功夫，但是总的而言还是上一道题所说过得算法思维。在倒酒的时候需要考虑 i 空和 i 不为空的情况。

四、 心得体会

经过本次实验，我学习掌握了 BFS 以及 DFS 算法，也通过不断摸索，学习到了利用数组存储遍历路径的方法。感悟颇多，最大的感受是数学知识与数学分析方法在算法中的重要性，当一道题没有头绪时，用笔认真在纸上分析一下往往能使自己的逻辑变得清晰明了，然后纸上的来说终觉浅，绝知此事要躬行，我将它用代码的形式写出来，获益颇多。