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

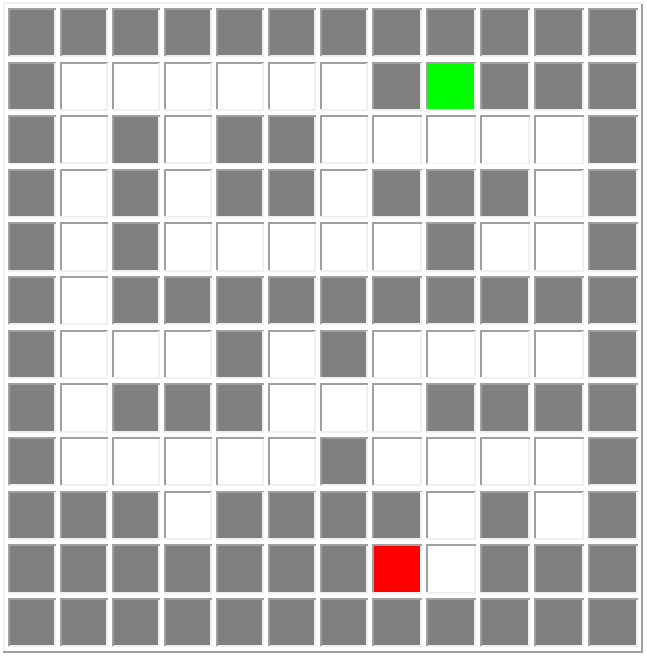
完成时间 2021 年 11 月 18 日

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。

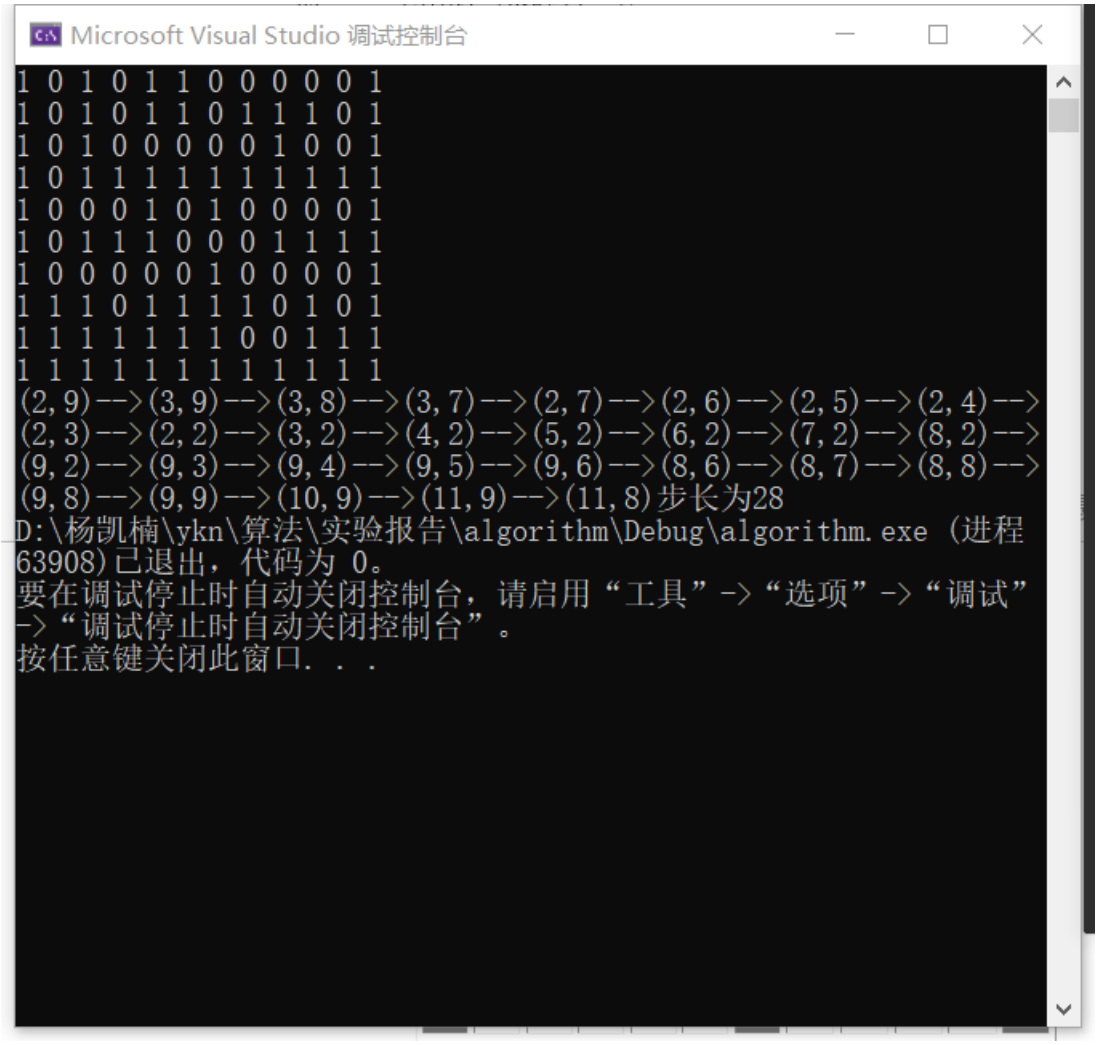
详细设计：

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

```cpp
using namespace std;
const int mov[4][2] = { {0,1 }, {0, -1},{ -1, 0},{ 1, 0} };
int size = 12;
int a[12][12];//方格状况
int save[12 * 12][2];//[当前已经遍历过的点的序号][横纵坐标】
int path[12 * 12];//【内为当前遍历序号】存储的内容为上一序号
int a1, b1;//出发点坐标
int a2, b2;//终点坐标
int countlen = 0;
bool judge(int i, int j) {
    if (a[i][j] == 1)
        return false;
    return true;
}
void coutmy(int a) {
    if (a == 1)
        return;
    countlen++;
    coutmy(path[a]);
    cout << "-->" <<"("<< save[a][0]+1 << "," << save[a][1]+1<<")";
}
void solution() {
    int sign1=0, sign2=1;//sign1 为空处遍历过的序号，sign2 为正确路径序号 path[]
    path[sign2] = sign1;//第一个遍历的前级肯定为出发点
    save[1][0] = a1; save[1][1] = b1;
    a[a1][b1] = 1;//出发点肯定走完了设为 1
    do {
        sign1++;
        for (int i = 0; i < 4; i++) {
            if (judge(save[sign1][0] + mov[i][0], save[sign1][1] + mov[i][1])) {
                sign2++;
                path[sign2] = sign1;
                save[sign2][0] = save[sign1][0] + mov[i][0];
                save[sign2][1] = save[sign1][1] + mov[i][1];
                a[save[sign2][0]][save[sign2][1]] = 1;
            }
            if (save[sign2][0] == a2-1 && save[sign2][1] == b2-1) {
                int count = 0;
                cout << "(" << a1 << "," << b1 << ")";
                coutmy(sign2);
                cout << "步长为" << countlen;
                return;
```

```
            }
        }
    } while (sign1 < sign2);

}
int main(void) {
    int i, j;

    cin >> a1 >> b1 >> a2 >> b2;

    for (i = 0; i < 12; i++)
        for (j = 0; j < 12; j++)
            cin >> a[i][j];
    solution();
}
```

运行结果：



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：

```
int captain[] = { 8,5,3 };//容器标准容量
int des[3];
int cou = 0;
int step = 0;
struct state {
    int now[3];//现在各杯中酒量;
    int xulie;//在数组中位置
    int qiandao;//实现这一步骤的前一步骤，前导
};
queue<state> q;//广度优先搜索队列
vector<state> p;//顺序数组
//判断是否可以进一步进行倒酒动作,i 向 j 里面倒酒。
bool check(int now[], int i, int j) ;

//输出函数
void out(int a);

//广度优先遍历，可以求得最优解
void bfs(state s);
```

详细设计：

```
bool check(int now[], int i, int j) {
    if (now[i] == 0 || now[j] == captain[j])
        return false;
    return true;
}
//输出函数
void out(int a) {
    step += 1;
    if (p[a].xulie == 0) {
        cout << p[a].now[0] << " " << p[a].now[1] << " " << p[a].now[2] << endl;
        return;
    }
```

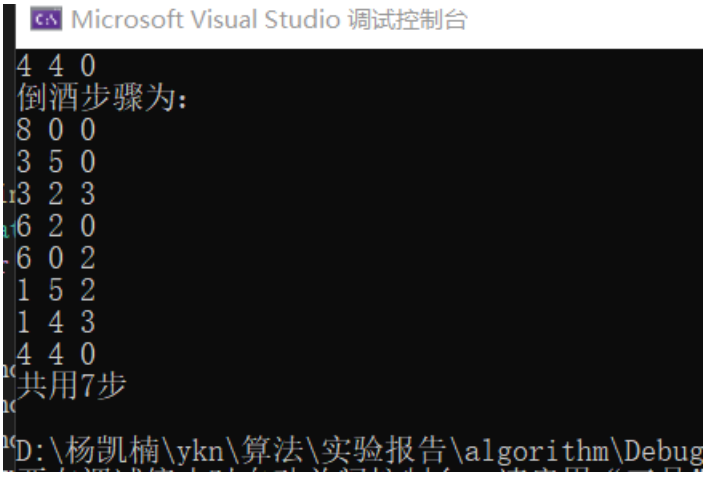
```
    out(p[a].qiandao);
    cout << p[a].now[0] << " " << p[a].now[1] << " " << p[a].now[2] << endl;
}
//广度优先遍历，可以求得最优解
void bfs(state s) {
    state tmp, tempr;
    s.xulie = 0;
    s.qiandao = 0;
    p.push_back(s);
    q.push(s);
    while (!q.empty()) {
        bool flag = true;
        tmp = q.front();
        for (int i = 0; i < 3; i++) {
            if (tmp.now[i] != des[i])
                flag = false;
        }
        if (flag){
            cout << "倒酒步骤为: " << endl;
            out(tmp.xulie);
            cout << "共用" << step-1 << "步" << endl;
            return;
        }
        for (int i = 0; i < 3; i++) {
            if (tmp.now[i] > 0) {
                for (int j = 0; j < 3; j++) {
                    if (i != j && check(tmp.now, i, j)) {
                        int tmpi = tmp.now[i], tmpj = tmp.now[j];
                        tempr = tmp;
                        //daojiu
                        tmpj += tmp.now[i];
                        tmpi -= tmp.now[i];
                        //panduan
                        if (tmpi == 0 && tmpj == captain[j]) {//ikong,
                            tempr.now[i] = tmpi;
                            tempr.now[j] = tmpj;
                            cou++;
                            tempr.xulie = cou;
                            tempr.qiandao =  tmp.xulie;
                            p.push_back(tempr);
                            q.push(tempr);
                        }
```

```
                        else if (tmpi == 0 && tmpj < captain[j]) {
                            tempr.now[i] = tmpi;
                            tempr.now[j] = tmpj;
                            cou++;
                            tempr.xulie = cou;
                            tempr.qiandao = tmp.xulie;
                            p.push_back(tempr);
                            q.push(tempr);
                        }
                        //ibukong
                        tmpi = tmp.now[i], tmpj = tmp.now[j];
                        tmpi -= (captain[j] - tmpj);
                        tmpj = captain[j];
                        if (tmpi > 0 && tmpi < captain[i]) {
                            tempr.now[i] = tmpi;
                            tempr.now[j] = tmpj;
                            cou++;
                            tempr.xulie = cou;
                            tempr.qiandao = tmp.xulie;
                            p.push_back(tempr);
                            q.push(tempr);
                        }
                    }
                }
            }
        }
        q.pop();
    }
}



int main() {
    state s;
    for (int i = 0; i < 3; i++) {
        cin >> des[i];
    }
    s.now[0] = 8;
    s.now[1] = 0;
    s.now[2] = 0;
    s.xulie = 0;
    s.qiandao = 0;
    bfs(s);
```

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

运行结果：

。



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。