

# 算法分析与设计

## 实验 05

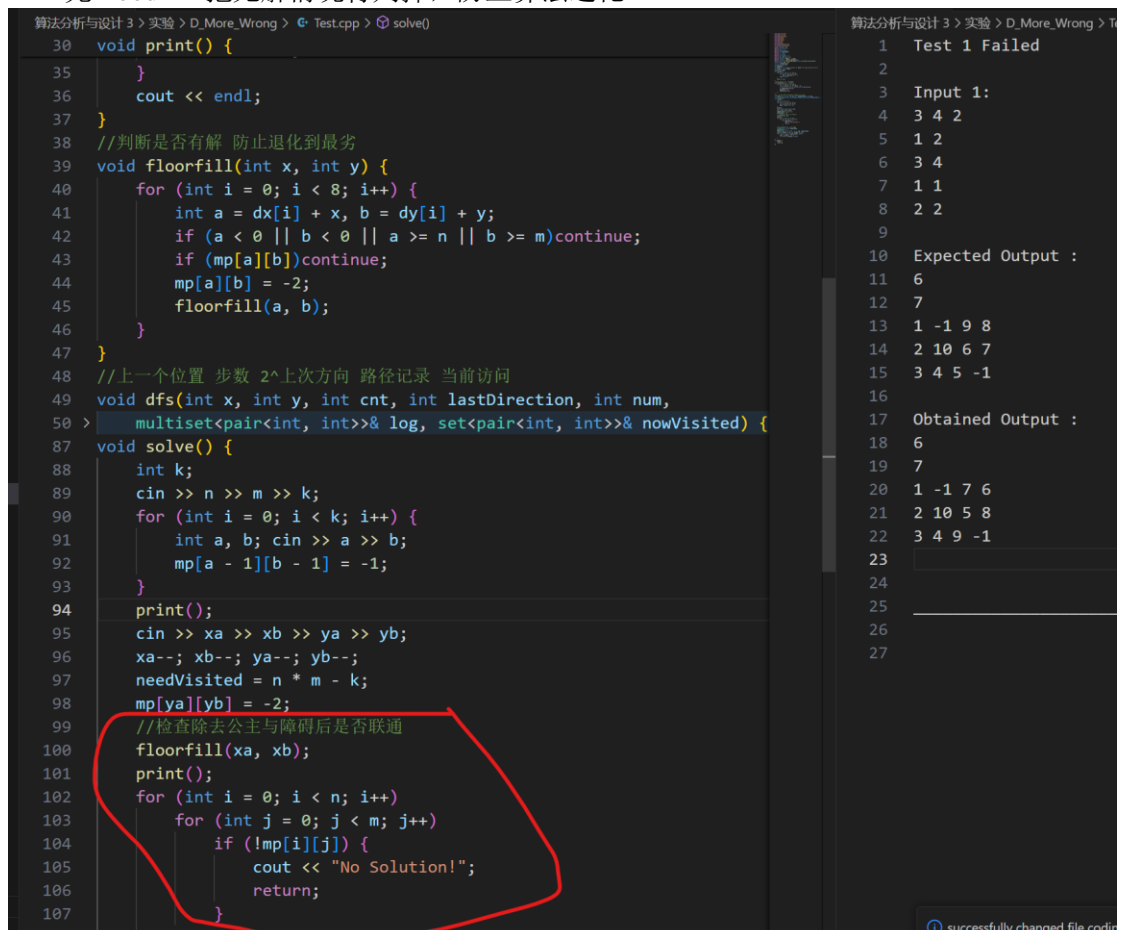
### 实验目的

- 掌握回溯法的深度优先搜索策略、剪枝策略及基本概念
- 掌握两种基本的解空间树：子集树和排列树
- 掌握装载问题、0-1 背包问题、n 后问题。

### 实训内容

1、算法设计题：罗密欧与朱丽叶的迷宫问题（P160，5-12）。

- 先 floodFill 把无解情况特判掉，防止算法退化



```
30 void print() {
31 }
32 cout << endl;
33 }
34 //判断是否有解 防止退化到最劣
35 void floodfill(int x, int y) {
36     for (int i = 0; i < 8; i++) {
37         int a = dx[i] + x, b = dy[i] + y;
38         if (a < 0 || b < 0 || a >= n || b >= m) continue;
39         if (mp[a][b]) continue;
40         mp[a][b] = -2;
41         floodfill(a, b);
42     }
43 }
44 //上一个位置 步数 2^上次方向 路径记录 当前访问
45 void dfs(int x, int y, int cnt, int lastDirection, int num,
46 > multiset<pair<int, int>>& log, set<pair<int, int>>& nowVisited) {
47 void solve() {
48     int k;
49     cin >> n >> m >> k;
50     for (int i = 0; i < k; i++) {
51         int a, b; cin >> a >> b;
52         mp[a - 1][b - 1] = -1;
53     }
54     print();
55     cin >> xa >> xb >> ya >> yb;
56     xa--; xb--; ya--; yb--;
57     needVisited = n * m - k;
58     mp[ya][yb] = -2;
59     //检查除去公主与障碍后是否联通
60     floodfill(xa, xb);
61     print();
62     for (int i = 0; i < n; i++)
63         for (int j = 0; j < m; j++)
64             if (!mp[i][j]) {
65                 cout << "No Solution!";
66                 return;
67             }
68 }
```

```
1 Test 1 Failed
2
3 Input 1:
4 3 4 2
5 1 2
6 3 4
7 1 1
8 2 2
9
10 Expected Output :
11 6
12 7
13 1 -1 9 8
14 2 10 6 7
15 3 4 5 -1
16
17 Obtained Output :
18 6
19 7
20 1 -1 7 6
21 2 10 5 8
22 3 4 9 -1
23
24
25
26
27
```

successfully changed file codin

- DFS 入口，初始化状态，利用&操作实现代价计算

```

109     multiset<pair<int, int>> log;
110     set<pair<int, int>> nowVisited;
111     mp[xa][xb] = 1;
112     dfs(xa, xb, 0, (1 << 9) - 1, 2, log, nowVisited);
113     cout << minAns << endl << cntAns << endl;
114     for (int i = 1; i <= path.size(); i++) {
115         cout << path[i - 1] << " ";
116         if (i % n == 0) cout << endl;
117     }
118 }

```

- DFS 过程，利用 set 与 multiset 判断是否满足条件

```

47 }
48 //上一个位置 步数 2^上次方向 路径记录 当前访问
49 void dfs(int x, int y, int cnt, int lastDirection, int num,
50         multiset<pair<int, int>>& log, set<pair<int, int>>& nowVisited) {
51     if (x == ya && y == yb) {
52         if (nowVisited.size() != needVisited) return;
53         //可能等于或小于
54         //由于走一步代价不一定增加，所以存在更新最优的情况
55         if (cnt < minAns) {
56             path.clear();
57             for (int i = 0; i < n; i++)
58                 for (int j = 0; j < m; j++)
59                     path.pb(mp[i][j]);
60             minAns = cnt;
61             cntAns = 1;
62         }
63         else cntAns++;
64     }
65     for (int i = 0; i < 8; i++) {
66         int a = x + dx[i], b = y + dy[i];
67         if (a < 0 || b < 0 || a >= n || b >= m) continue;
68         //前面floorfill时把所有位置涂成-2了
69         if (mp[a][b] != -2) continue;
70         //如果没转向&后为真
71         int nowCnt = cnt + !(lastDirection & (1 << i));
72         //最优性剪枝
73         if (nowCnt > minAns) continue;
74
75         log.insert({ a,b });
76         nowVisited.insert({ a,b });
77         mp[a][b] = num;
78         print();
79         dfs(a, b, nowCnt, 1 << i, num + 1, log, nowVisited);
80         mp[a][b] = -2;
81         //恢复现场
82         log.erase(log.find({ a,b }));
83         if (log.find({ a,b }) == log.end())
84             nowVisited.erase({ a,b });
85     }
86 }

```

2、算法设计题：使用回溯法实现最佳调度问题(P162, 5-15) (需要文字描述解空间树的组织和限界函数的设计)。

● 空间树与有界函数

对于样例: 7 3  
2 14 4 16 6 5 3

第1次分配

第2次分配

$t = \{t_0, t_1, \dots, t_n\}$   $k \geq 3$  则共有3次种情况

有限函数: 当前搜到第  $k$  个人, 第  $i$  个物品, 则

$q[k] + w[i] \leq \min$

若大于等于则不可能产生更优解, 故剪枝

对于 3 2  
样例: 4 4 有:

第 页

## ● 代码

```
Test.cpp x output_1.txt input_1.txt  Test.cpp > dfs(int)
1  #include<iostream>
2  #include<cstring>
3  #include<queue>
4  using namespace std;
5  #define ll long long
6  #define endl '\n'
7  #define inf 0x3f3f3f3f
8  ll n,m,ans=inf;
9  vector<ll> q,w;
10 void dfs(int x){
11     if(x==n){
12         ll now=-inf;
13         for(auto t:q)now=max(now,t);
14         //因为有剪枝所以这里一定更优秀
15         ans=now;
16         return;
17     }
18     for(int i=0;i<m;i++){
19         //最优性剪枝
20         if(q[i]+w[x]>=ans)continue;
21         q[i]+=w[x];
22         dfs(x+1);
23         q[i]-=w[x];
24     }
25 }
26 void solve(){
27     //n件事 m人
28     cin>>n>>m;
29     q.resize(m,0);
30     w.resize(n);
31     for(auto &t:w)cin>>t;
32     dfs(0);
33     cout<<"最短时间消耗: "<<ans;
34 }
35 int main(){
36     solve();
37     return 0;
38 }
```

```
result.txt x
算法分析与设计 3 > 实验 > D_More_Wrong > Tests >
1  Test 1 Passed
2
3  Input 1:
4  7 3
5  2 14 4 16 6 5 3
6
7  Expected Output :
8  最短时间消耗: 17
9
10 Obtained Output :
11 最短时间消耗: 17
12
13
14
15
```