

## 算法分析题 5-3 背包问题（回溯）

### 题目

重写 0-1 背包问题的回溯法，使算法能输出最优解。

### 源代码

```
1. #include <iostream>
2. #include <algorithm>
3. using namespace std;
4.
5. int n;          // 物品数
6. int c;          // 背包容量
7. int w[100] = {0}; // 物品重量
8. int v[100] = {0}; // 物品价值
9. int weight;     // 总重量
10. int value;      // 总价值
11. int vt;         // 最优价值
12.
13. // 回溯递归
14. void backtrack(int m)
15. {
16.     if (m == n) // 递归出口，当前方案与最优方案比较
17.     {
18.         vt = max(vt, value);
19.     }
20.     else
21.     {
22.         for (int i = 0; i < 2; i++) // 0 表示不加, 1 表示加入
23.         {
24.             if (i == 1 && weight + w[m] <= c)
25.             {
26.                 weight += w[m];
27.                 value += v[m];
28.             }
29.             backtrack(m + 1);
30.             if (i == 1)
31.             {
32.                 weight -= w[m];
33.                 value -= v[m];
```

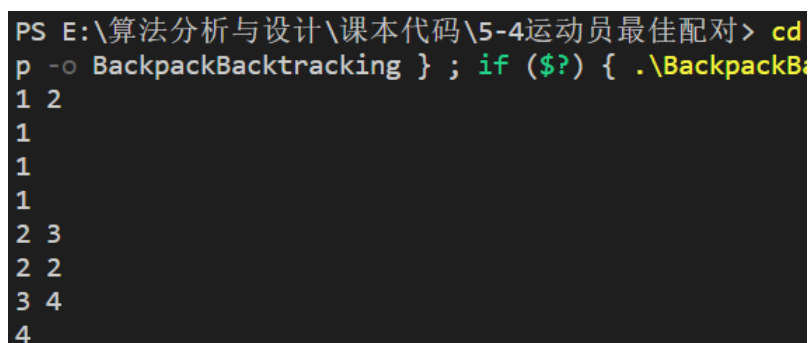
```

34.         }
35.     }
36. }
37. }
38.
39. int main()
40. {
41.     while (cin >> n >> c)
42.     {
43.         for (int i = 0; i < n; i++)
44.             cin >> w[i];
45.         for (int i = 0; i < n; i++)
46.             cin >> v[i];
47.         value = vt = weight = 0;
48.         backtrack(0);
49.         cout << vt << endl;
50.     }
51.     return 0;
52. }

```

## 程序运行截图

在截图中使用了两组测试数据，第一组有 1 个物品，背包容量为 2，这个物品的重量为 1，价值为 1，经计算，其最优价值为 1。第二组有 2 个物品，背包容量为 3，这 2 个物品的重量分别为 2, 2；价值为 3, 4；经计算，其最优价值为 4。



```

PS E:\算法分析与设计\课本代码\5-4运动员最佳配对> cd p -o BackpackBacktracking } ; if ($?) { .\BackpackB
1 2
1
1
1
2 3
2 2
3 4
4

```

图 10-1 背包问题运行结果截图

## 实验心得

解决本问题在于回溯的核心思想，通过遍历解空间树，来更新最优解。背包问题的解空间树为一棵子集树。需要注意的就是在进入下一层时要加上所选物品，在返回上一层时要减去所选物品。

# 算法实现题 5-4 运动员最佳配对问题

## 题目

### 5-4 运动员最佳配对问题。

**问题描述：**羽毛球队有男女运动员各  $n$  人。给定 2 个  $n \times n$  矩阵  $P$  和  $Q$ 。 $P[i][j]$  是男运动员  $i$  和女运动员  $j$  配对组成混合双打的男运动员竞赛优势； $Q[i][j]$  是女运动员  $i$  和男运动员  $j$  配合的女运动员竞赛优势。由于技术配合和心理状态等各种因素影响， $P[i][j]$  不一定等于  $Q[j][i]$ 。男运动员  $i$  和女运动员  $j$  配对组成混合双打的男女双方竞赛优势为  $P[i][j] \times Q[j][i]$ 。设计一个算法，计算男女运动员最佳配对法，使各组男女双方竞赛优势的总和达到最大。

**算法设计：**设计一个算法，对于给定的男女运动员竞赛优势，计算男女运动员最佳配对法，使各组男女双方竞赛优势的总和达到最大。

**数据输入：**由文件 input.txt 给出输入数据。第一行有 1 个正整数  $n$  ( $1 \leq n \leq 20$ )。接下来的  $2n$  行，每行  $n$  个数。前  $n$  行是  $p$ ，后  $n$  行是  $q$ 。

**结果输出：**将计算的男女双方竞赛优势的总和的最大值输出到文件 output.txt。

输入文件示例

input.txt

3

10 2 3

2 3 4

3 4 5

2 2 2

3 5 3

4 5 1

输出文件示例

output.txt

52

## 源代码

```
1. #include <iostream>
2. #include <algorithm>
3. #include <fstream>
4. #define N 100
5. using namespace std;
6.
7. int n;           // 男女运动员各 n 人
8. int P[N][N], Q[N][N]; // 竞赛优势
9. int x[N];        // 储存 j 的排列组合
10. int result[N];   // 储存最优接
11. int tempValue = 0, maxValue = 0;
12.
13. ifstream input("input.txt");
14. ofstream output("output.txt");
15.
16. /*****
17. * 函数描述： 从文件中读取测试数据
18. *****/
```

```

19. void getData()
20. {
21.     input >> n;
22.
23.     for (int i = 1; i <= n; i++)
24.     {
25.         x[i] = i;
26.     }
27.
28.     for (int i = 1; i <= n; i++)
29.     {
30.         for (int j = 1; j <= n; j++)
31.         {
32.             input >> P[i][j];
33.         }
34.     }
35.     for (int i = 1; i <= n; i++)
36.     {
37.         for (int j = 1; j <= n; j++)
38.         {
39.             input >> Q[i][j];
40.         }
41.     }
42. }
43.
44. /*****
45. * 函数描述: 格式化输出结果
46. *****/
47. void outputResult()
48. {
49.     output << maxValue << endl;
50.     cout << maxValue << endl;
51.
52.     for (int i = 1; i <= n; i++)
53.     {
54.         cout << i << "----" << result[i] << endl;
55.     }
56. }
57.
58.
59. /*****
60. * 函数描述: 计算竞赛优势并更新
61. *****/
62. void compute()
63. {
64.     tempValue = 0;
65.     for (int i = 1; i <= n; i++)
66.     {
67.         tempValue += P[i][x[i]] * Q[x[i]][i];

```

```

68.     }
69.     if (tempValue > maxValue)
70.     {
71.         maxValue = tempValue;
72.         for (int i = 1; i <= n; i++)
73.         {
74.             result[i] = x[i];
75.         }
76.     }
77. }
78.
79.
80. /*****
81. * 函数描述: 回溯递归函数, 遍历所有的组合方案
82. *****/
83. void backtrack(int t)
84. {
85.     int i, temp;
86.     if (t > n) // 递归出口
87.     {
88.         compute();
89.     }
90.     for (i = t; i <= n; i++)
91.     {
92.         // 排列树, 交换(j)
93.         temp = x[i];
94.         x[i] = x[t];
95.         x[t] = temp;
96.         backtrack(t + 1);
97.         temp = x[i];
98.         x[i] = x[t];
99.         x[t] = temp;
100.    }
101. }
102.
103. int main()
104. {
105.     getData(); // 读取测试数据
106.     backtrack(1);
107.     outputResult(); // 输出结果
108.     input.close();
109.     output.close();
110. }

```

## 程序运行截图

```
cd "e:\算法分析与设计\课本代码\5-4运动
PS E:\算法分析与设计\课本代码\5-3背包\
hPlayer } ; if ($?) { .\MatchPlayer }
52
1----1
2----3
3----2
```

图 2 运动员配对问题运行结果截图

input.txt - 记事本

文件(E) 编辑(E) 格式(O)

```
3
10 2 3
2 3 4
3 4 5
2 2 2
3 5 3
4 5 1
```

图 3 input.txt 输入文件截图

output.txt - 记事本

文件(F) 编辑(E) 格式(O)

```
52
```

图 4 output.txt 输出文件截图

## 实验心得

这个问题也是通过回溯法遍历解空间树，并对最优解进行更新。此问题的解空间树是排列树，是在确定  $i$  的情况下对  $j$  进行排列，在进入下一层时对  $j$  的顺序进行交换，返回上一层时还原交换。在递归中，对  $i$  进行循环遍历，形成了一个二维的组合遍历。其中 `compute()` 函数专门对竞争优势的数值进行计算，并对最优解进行更新。