222018321062006 宋行健

# 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;
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
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. }
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。

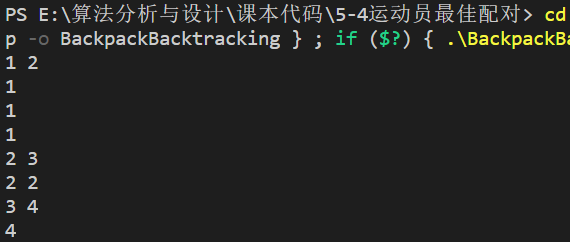


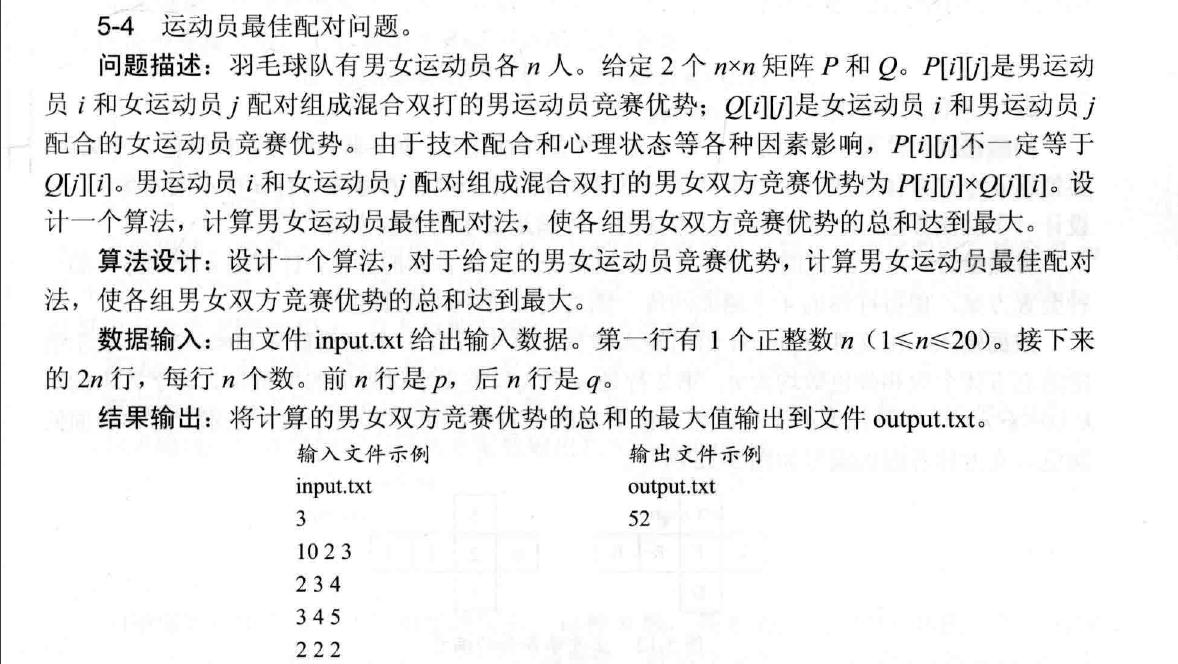
图 1 0-1背包问题运行结果截图

## 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





## 源代码

1. #include <iostream>
2. #include <algorithm>
3. #include <fstream>
4. #define N 100
5. **using** **namespace** std;
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;
13. ifstream input("input.txt");
14. ofstream output("output.txt");
16. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
17. \* 函数描述： 从文件中读取测试数据
18. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
19. **void** getData()
20. {
21. input >> n;
23. **for** (**int** i = 1; i <= n; i++)
24. {
25. x[i] = i;
26. }
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. }
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;
52. **for** (**int** i = 1; i <= n; i++)
53. {
54. cout << i << "----" << result[i] << endl;
55. }
56. }

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. }

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. }
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

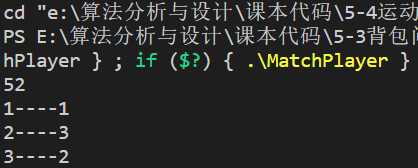


图 2运动员配对问题运行结果截图

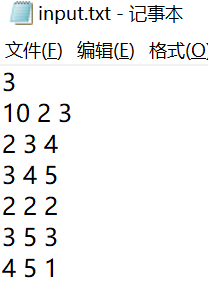


图 3 input.txt输入文件截图

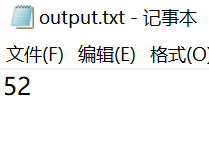


图 4 output.txt输出文件截图

## 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。