浙江工艺大学

算法分析与设计实验报告

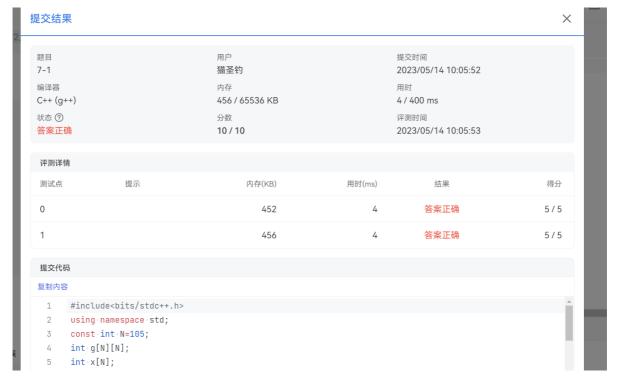
(2021级)



实验题目	实验 1
学生姓名	温家伟
学生学号	202103151422
专业班级	大数据分析 2101
所在学院	理学院
提交日期	2023-3-8
第5章 回溯法	
实验结果及相应代码	

1. 旅行售货员

1.1 PTA提交代码截图



1.2 PTA提交代码结果截图

C'' (g'')

```
#include<bits/stdc++.h>
 1
 2
      using namespace std;
 3
      const int N=105;
      int g[N][N];
 4
      int x[N];
 5
      int n;
 7
      int ans=0x3f3f3f3f;
 8
      int now=0;
      void TSP(int u)
 9
10
          if(u>n)
11
          {
12
              if(now+g[x[n]][x[1]] < ans)
13
              {
14
                   ans=now+g[x[n]][x[1]];
15
              }
16
          }
17
18
          else
          {
19
              for(int i=u;i<=n;i++)</pre>
20
              {
21
22
                   if(q[x[v-1]][x[i]]!=0x3f3f3f3f&&now+q[x[v-1]])
                   {
23
                       swap(x[u],x[i]);
24
                       now+=g[x[u-1]][x[u]];
25
                       TSP(u+1);
26
                       now-=g[x[u-1]][x[u]];
27
                       swap(x[u],x[i]);
28
29
                   }
              }
30
          }
31
      }
32
33
      int main()
34
      {
35
          cin>>n;
          memset(g, 0x3f, sizeof(g));
36
          for(int i=1;i<=n;i++)</pre>
37
38
          {
```

```
{
40
                cin>>g[i][j];
41
            }
42
         }
43
         for(int i=1;i<=n;i++) x[i]=i;</pre>
44
45
        TSP(2);
46
47
         cout<<ans;
48
        return 0;
49
50
```

1.3 算法分析

使用回溯法可以枚举所有情况(全排列),从而求出最优解。

- 将起始点作为当前点。
- 从当前点开始,前往没有遍历过的点。
- 在第一次回到起始点时,判断是否遍历了所有点。
- 如果是,则更新最优解,然后返回上一个节点(回溯到上一个分支)。
- 如果不是,则继续前往未访问过的节点。
- 直到所有情况都枚举完毕。

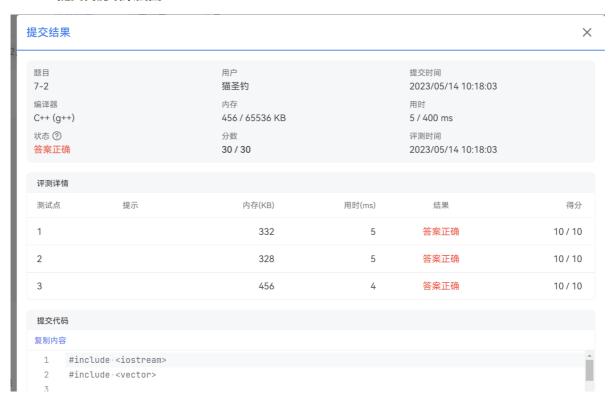
2 装载问题

2.1 PTA提交代码截图

```
C++(q++)
     #include <iostream>
 1
     #include < vector >
 2
 3
 4
     using namespace std;
 5
 6
     const int MAX_N = 100;
 7
     int c1, c2;
 8
 9
     int n;
     vector<int> w(MAX_N);
10
11
     vector<int> flag(MAX_N);
12
     int rest_capacity = 0;
13
     int current_weight = 0;
14
     int optimal_weight = 0;
15
     vector<int> choose(MAX_N);
16
     void backtrack(int i) {
17
     · · · · if · (i · > · n) · {
18
19
     ....if (current_weight > optimal_weight) {
     ·····optimal_weight = current_weight;
20
21
      ....choose.assign(flag.begin(), flag.end());
22
     . . . . . . . . }
23
      ····return;
24
     . . . . }
25
     rest_capacity -= w[i];
26
27
     if (current_weight + w[i] <= c1) {</pre>
28
29
     ....current_weight += w[i];
30
      \cdots flag[i] = 1;
     backtrack(i + 1);
31
     current_weight -= w[i];
32
     ·····flag[i]·=·0;
33
     . . . . }
34
35
     if (current_weight + rest_capacity > optimal_weight)
36
37
      backtrack(i + 1);
     . . . . }
38
39
```

```
40
     rest_capacity += w[i];
41
     }
42
     int main() {
43
     · · · · int · sum · = · 0;
44
     ····cin·>>·n;
45
46
     · · · · for · (int · i · = · 1; · i · <= · n; · i++) · {
47
48
     ·····cin·>>·w[i];
49
     rest_capacity += w[i];
     ····sum·+=·w[i];
50
     ...}
51
52
     ····cin·>> c1·>> c2:
53
54
55
     · · · backtrack(1);
56
     ····int·loop_count·=·0;
57
58
     ···if·(sum·-·optimal_weight·<=·c2) {
     ••••••for (int · i · = · 1; · i · <= · n; · i++) · {
59
     ·····if·(choose[i]·==·1)·{
60
     ·····if·(!loop_count)·{
61
62
     ····cout·<<·i;
63
     ·····loop_count++;
64
     ·····}·else·{
     ·····cout·<<·i;
65
     66
     67
     . . . . . . . . }
68
     ·····loop_count·=·0;
69
     ····cout·<< endl;
70
71
72
     \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot for \cdot (int \cdot i \cdot = \cdot 1; \cdot i \cdot \langle = \cdot n; \cdot i + +) \cdot \{
     ·····if·(choose[i]·==·0)·{
73
     ·····if·(!loop_count)·{
74
75
     ----cout << i;
     ·····loop_count++;
76
     ·····}·else·{
77
     ·····cout·<<·"·"·<<-i:
78
     79
     \Omega
```

2.2 PTA提交代码结果截图



2.3 算法分析

- 将所有货物从大到小排序。
- 初始化两个卡车的当前重量为0,当前位置为1。
- 从第一个货物开始,依次尝试将其放到两个卡车中的任意一个上,如果可以,则更新对应卡车的当前重量,并进入下一个货物的选择;如果不行,则回溯到上一个状态。
- 当所有货物都尝试过后,比较两辆卡车的重量差,更新最优解。
- 如果当前的状态是合法状态(即两辆卡车的重量都没有超过总载重量的一半),则继续扩展状态;否则,回溯到上一个状态。
- 当所有状态都扩展完毕后,回溯到最初状态,输出最优解。