

实验7-装载问题

问题分析

- 定义bestw为第一艘船能装的最大装载重量，定义字符串best为最佳的分配方案，cur为当前的分配方案，curw为当前方案第一艘船装载重量
- n的数据范围只有20，采用DFS暴力搜索答案即可，在递归的每一层，有两种状态可选，一是当前集装箱放在了第一艘船上（在能放下的前提下），二是当前集装箱选择放在第二艘船上。选上该集装箱时，对cur字符串相应的位置赋值为1，在退出递归时再赋值为0，即回溯的时候恢复现场

```
1  if (curw+w[u]<=c1) {
2      cur[u]='1';//能选则选上该货物
3      dfs(u+1, curw+w[u]);
4      cur[u]='0';//回溯时恢复现场
5  }
6  dfs(u+1, curw);//不选该货物
```

- 当u==n+1时代表到达搜索终点，若剩余货物第二艘船装不下则代表当前方案为不合法的装配方案，若当前第一艘船能装的重量大于第一艘船能装的最大装载重量，则更新最大装载重量，如果存在多种最大的方案 选择字典序更大的（根据字符串比较即可）

```
1  if (u==n+1) { //已经分配完
2      if (sum-curw>c2) {
3          return; //剩余货物第二艘船装不下 不合法的装配方案
4      }
5      if (curw>bestw) {
6          bestw=curw;
7          best=cur;
8      }
9      if (curw==bestw&&cur>best) {
10         //如果存在多种最大的方案 选择字典序更大的
11         best=cur;
12     }
13     return;
14 }
```

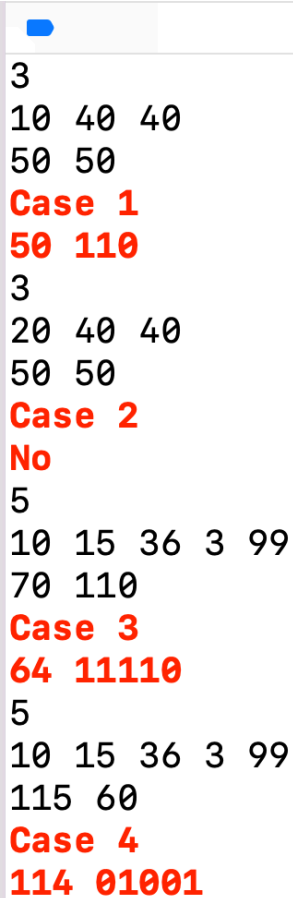
- 当两艘船能装的最大重量仍装不下全部的集装箱时，则不能装载

```
1    if (bestw+c2<sum) { //两艘船能装的最大重量仍装不下 则不能装载
2        cout<<"No\n";
3        continue;
4    }
```

- 对于多组测试数据，注意数据的初始化

```
1    best.resize(n+1, '0');
2    cur.resize(n+1, '0');
3    sum=0;
4    bestw=0;
```

运行结果截图



```
3
10 40 40
50 50
Case 1
50 110
3
20 40 40
50 50
Case 2
No
5
10 15 36 3 99
70 110
Case 3
64 11110
5
10 15 36 3 99
115 60
Case 4
114 01001
```

调试中的问题

回溯时忘记恢复现场导致答案错误，在退出下一层的搜索时，将字符串`cur`对应位置恢复为0即可

```
1    cur[u]='1';//能选则选上该货物
2    dfs(u+1, curw+w[u]);
3    cur[u]='0';//回溯时恢复现场
```

实验体会

本次实验是用DFS求解最佳方案，整体比较简单，需要注意一些搜索的细节，比如对于一些数据的进入下一层搜索时的改变与退出时的恢复。在本次实验的代码编写中，让我感到了回溯法思想很是巧妙，我们对每一层的状态进行改变和回溯便能不重不漏地搜索出所有的状态，那么在那些状态里去找最佳的状态就迎刃而解啦，只需要在递归的终点对答案更新即可。这是本课程的最后一次实验，在此希望自己以后依旧能热爱算法，能用巧妙的算法去解决更多问题。

程序源码

```
1    #include <bits/stdc++.h>
2    using namespace std;
3    const int N=30;
4    int n,w[N],c1,c2,sum;
5    int bestw;//第一艘船能装的最大装载重量
6    string best,cur;//最佳分配方案 当前分配方案
7    void dfs(int u,int curw){
8        if (u==n+1) {//已经分配完
9            if (sum-curw>c2) {
10                return;//剩余货物第二艘船装不下 不合法的装配方案
11            }
12            if (curw>bestw) {
13                bestw=curw;
14                best=cur;
15            }
16            if (curw==bestw&&cur>best) {
17                //如果存在多种最大的方案 选择字典序更大的
18                best=cur;
19            }
20            return;
```

```

21     }
22     if (curw+w[u]<=c1) {
23         cur[u]='1';//能选则选上该货物
24         dfs(u+1, curw+w[u]);
25         cur[u]='0';//回溯时恢复现场
26     }
27     dfs(u+1, curw);//不选该货物
28 }
29
30 int main(){
31     int cas=1;
32     while (cin>>n) {
33         best.resize(n+1, '0');
34         cur.resize(n+1, '0');
35         sum=0;
36         bestw=0;
37         for (int i=1; i<=n; i++) {
38             cin>>w[i];
39             sum+=w[i];
40         }
41         cin>>c1>>c2;
42         dfs(1,0);
43         cout<<"Case "<<cas++<<"\n";
44         if (bestw+c2<sum) {
45             //两艘船能装的最大重量仍装不下 则不能装载
46             cout<<"No\n";
47             continue;
48         }
49         best=best.substr(1,n);
50         cout<<bestw<<" "<<best<<"\n";
51     }
52 }

```