2019-2020学年第一学期算法设计与分析

上机实验报告

实验名称：装载问题

学 号	秦敏浩	姓 名	17122490	评 分	
专 业	计算机科学与技术	实验类型	综合	任课教师	岳晓冬
完成日期	2019.10.17	实验学时	2		

一、实验问题描述

1、实验内容

装载问题：有 n 个集装箱要装上 2 艘载重量分别为 c_1 和 c_2 的轮船，其中第 i 个集装箱的重量为 w_i，要求确定是否有一个合理的装载方案可将这个集装箱装上这 2 艘轮船。如果有，找出一种装载方案。

2、实验目的

- 分析问题，说明原理和方法
- 使用 BFS 或 DFS 求解该问题
- 将时间复杂度限制在 $O(2^n)$

3、实验环境

操作系统：Linux/Windows

编译环境：GNU G++14

二、算法设计与分析

1、算法基本思想

Brute force。考虑 n 个物品最多有 2^n 种状态可以枚举。如果采用 DFS 的方法进行，每一层决策当前物品分给 1 号船还是 2 号船，通过参数维护当前船的重量，即可在总体时间复杂度为 $O(2^n)$ 下通过此题。

特别的，当物品总重量超过船的总承载能力时，答案一定是不存在的。当递归到某一层物品既无法放入 1 号船又无法放入 2 号船时，可以进行**可行性剪枝**。

2、算法设计与求解步骤

ans，bestans：用 01 串的形式记录当前决策和最佳决策

a：每个物品的重量

w1，w2：当前船 1 和船 2 的剩余载重能力

- 输入 w1，w2 和 a
- 从第一个物品开始 dfs，直至深度为 n
- 对于每一次 dfs，尽可能将物品往 w1 船放，其次尝试往往 w2 船放
- 当能成功放置完最后一个物品时输出答案，并更新最佳决策

3、算法分析

考虑到这是一个深度为 n 的递归，而每一层分支数为 2 个，处理复杂度为 $O(1)$，故总时间复杂度为$O(2^n)$。

4、算法程序实现

篇幅原因，见附录或附件。

三、调试与运行

输入：

```
3
10 40 40
50 50
3
20 40 40
50 50
```

输出：

```
3
10 40 40
50 50
Case 1
answer: 50 110
answer: 50 101
answer: 40 010
answer: 40 001
Best answer: 50 110
3
20 40 40
50 50
Case 2
No
```

四、结果分析

本题属于最基本的 Brute Force 做法。按照最朴素的想法求解问题即可。

由于题目输入没有保证每一个物品重量之间的关系，又规定了当有多种方案时输出字典序最小的方案，代码优化空间不大。

但如果要求输出任意一种方案时，可以先 $O(n\log n)$ 将每个物品按重量从大到小排序，这样剪枝的深度就会更浅一些。

五、本次实验的收获、心得体会

本次实验可以说是所有实验中最简单的一个，使用最朴素的思路和解法。

实际上本次实验的代码的本质 一个二进制枚举。若将 n 个物品的选择映射到 0 至 2 的 n 次方减一的话，那么每一个 n 位二进制数恰好对应一种枚举方式。但由于如此做的检查合法复杂度为 $O(n)$ ，影响到总体时间复杂度为 $O(n\cdot 2^n)$ 。考虑到可以边枚举边维护，就自然的产生了 DFS 的做法。

附录：代码实现（也可见附件）

```cpp
#include <iostream>
#include <string>
using namespace std;

int n,w1,w2,bestw; // w1,w2: the weight the boats can take currently
int a[25];
string ans,bestans; // string that takes the answer

void dfs(int p,int sum){
    if (p==n){
        if (sum>bestw){ // if sum==bestw then the lexicographical order must be lower
            bestw=sum;
            bestans=ans;
        }
        cout<<"answer: "<<sum<<' '<<ans<<endl; // a feasible answer (not necessarily the best)
        return;
    }
    if (a[p]<=w1){ // boat 1 takes
        w1-=a[p];
        ans.push_back('1'); // add 1
        dfs(p+1,sum+a[p]);
        ans.erase(--ans.end()); // delete the 1
        w1+=a[p];
    }
    if (a[p]<=w2){ // boat 2 takes
        w2-=a[p];
        ans.push_back('0');
        dfs(p+1,sum);
        ans.erase(--ans.end());
        w2+=a[p];
    }
}

```

```cpp
int main(){
    int kase=0;
    while (cin>>n){ // do until EOF
        int sum=0;
        for (int i=0;i<n;++i){
            cin>>a[i];
            sum+=a[i];
        }
        cin>>w1>>w2;
        cout<<"Case "<<++kase<<endl;
        if (sum>w1+w2){ // obviously "No"
            cout<<"No"<<endl;
            continue;
        }
        bestw=-1;
        ans=bestans="";
        dfs(0,0); // O(2^n) to find all the solutions
        if (bestw==-1){
            cout<<"No"<<endl; // still possible that there is no solution existing
        }
        else{
            cout<<"Best answer: "<<bestw<<' '<<bestans<<endl; // else output the best solution
        }
    }
    return 0;
}
```