实验七

1. 问题描述与实验目的：

有 n 个集装箱要装上 2 艘载重量分别为 c1 和 c2 的轮船，其中第 i 个集装箱的重量为 wi，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。

注意，在满足 的条件下才可能将这个集装箱装上这 2 艘轮船。

2. 要点分析和算法描述：

本题装载问题，可以使用子集树方式解决。采用 dfs 算法，树的深度为 n,第 i 层表示第 i 个集装箱的装在方案。向左分支表示装入第一艘船，向右分支表示装入第二艘船。

从根结点到第 n 层每个节点的一条路径就表示一种装载方案。在每次递归中，如果全使用 if,将会搜索所有可能的装载方案。即尝试将每个物品装入第一艘船或第二艘船，n 个集装箱，每个集装箱两种选择，整个树的规模为指数级，故算法时间复杂度为 $O(2^n)$。

改进：题目要求输出的 01 序列中字典序最大的。在本题中，即每个集装箱在两艘船都能装下的情况下，优先装第一艘，即左边。可以使用 else if，每次只会选择一种情况进行递归。采用该方法进行剪枝。每个物品只会有一种选择,“有左装左，无左装右”。能将算法时间复杂度控制在 O(n)。即放弃多种方案搜索，只寻找优先装第一艘船的情况。

3. 代码实现：

```
#include <iostream>
#include <cstring>
using namespace std;
const int N = 1e5 + 10;
int n;
int a[N];  // 记录每个物品的重量
int st[N]; // 1 表示装在左边，0 表示装在右边
bool flag; // 记录是否找到答案
int cnt = 0;
// u 表示当前处理的物品，lf 表示左边的重量，rf 表示右边的重量
void dfs(int u, int lf, int rf)
{
    if (flag)
        return;
    if (u == n + 1)
    {
        flag = 1;
        int x = 0;  // 记录第一艘船装的重量
        string ans; // 记录装载方案，1 表装左边，0 表示装右边
        for (int i = 1; i <= n; i++)
        {
            if (st[i])
                x += a[i];
```

```
            ans += to_string(st[i]);
        }
        cout << x << " " << ans << endl;
        return;
    }
    if (lf >= a[u])
    {
        st[u] = 1;
        dfs(u + 1, lf - a[u], rf);
    }
    else if (rf >= a[u])
    {
        st[u] = 0;
        dfs(u + 1, lf, rf - a[u]);
    }
}
int main()
{
    int k = 1;
    while (cin >> n)
    {
        for (int i = 1; i <= n; i++)
        {
            cin >> a[i];
            st[i] = 1; // 初始化为1,表示装在左边
        }
        int lf, rf;
        cin >> lf >> rf;
        flag = 0;
        cout << "Case " << k++ << endl;
        dfs(1, lf, rf);
        if (!flag) // 没有找到答案
            cout << "No" << endl;
    }
    return 0;
}
```

4. 运行结果

```
3
10 40 40
50 50
Case 1
50 110
3
20 40 40
50 50
Case 2
No
```

5. 实验体会

本题装载问题，是一个典型的子集树问题，在该树中，每层表示每个集装箱的装法，第 n 层的每个节点表示了一个装载方案。如果搜索所有可行方案，复杂度在指数级，但如果只寻找一种字典序最大的装载方案，我觉得复杂度在 O(n)即可完成