# 【华为OD机考 统一考试机试C卷】寻找最富裕的小家庭（C++ Java JavaScript Python C语言）

## 华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

目前在考C卷，经过两个月的收集整理，C卷真题已基本整理完毕
抽到原题的概率为2/3到3/3，也就是最少抽到两道原题。请注意：大家刷完C卷真题，最好要把B卷的真题刷一下，因为C卷的部分真题来自B卷。
另外订阅专栏还可以联系笔者开通在线 OJ 进行刷题，提高刷题效率。
真题目录：华为OD机考机试 真题目录（C卷 + D卷 + B卷 + A卷） + 考点说明
专栏：2023华为OD机试( B卷+C卷+D卷)（C++JavaJSPy）
华为OD面试真题精选：华为OD面试真题精选
在线OJ：点击立即刷题，模拟真实机考环境

## 题目描述

在一棵树中，每个节点代表一个家庭成员，节点的数字表示其个人的财富值，一个节点及其直接相连的子节点被定义为一个小家庭。

现给你一棵树，请计算出最富裕的小家庭的财富和。

## 输入描述

第一行为一个数N，表示成员总数，成员编号1-N,1<=N<=1000
第二行为N个空格分隔的数，表示编号1-N的成员的财富值。0<=财富值<=1000000
接下来N-1行，每行两个空格分隔的整数(N1N2)，表示N1是N2的父节点。

## 输出描述

最富裕的小家庭的财富和

## 用例

输入

```
4
100 200 300 500
1 2
1 3
2 4
```

输出

```
700
```

说明

> 成员1，2，3组成的小家庭财富值为600
> 成员2，4组成的小家庭财富值为700

## 解题思路

1. 首先，读取成员总数N和每个成员的财富值。为了方便处理，我们将财富值存储在一个数组wealth中，下标从1开始。
2. 初始化一个数组familyWealth，用于存储每个小家庭的财富和。初始时，每个小家庭的财富和等于对应成员的财富值。
3. 初始化一个变量maxWealth，用于存储最大的财富和。初始值为0。
4. 遍历每个父子关系，对于每个关系，执行以下操作：
   a. 读取父子关系中的两个成员N1和N2。
   b. 将N2的财富值累加到N1所在小家庭的财富和中，即更新familyWealth[N1]。
   c. 更新最大的财富和maxWealth，使其始终为当前已遍历的小家庭中财富和的最大值。
5. 遍历完所有父子关系后，maxWealth即为最富裕的小家庭的财富和。输出maxWealth作为结果。

这种解题思路的时间复杂度为O(N)，因为我们只需要遍历一次父子关系，就可以计算出每个小家庭的财富和，并在过程中更新最大的财富和。这种方法相对高效。

## C++

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int main() {
    int N; // 成员总数
    cin >> N;
    vector<int> wealth(N + 1); // 存储每个成员的财富值
    vector<int> familyWealth(N + 1); // 存储每个小家庭的财富和
    for (int i = 1; i <= N; i++) {
        cin >> wealth[i]; // 读取每个成员的财富值
        familyWealth[i] = wealth[i]; // 初始化每个小家庭的财富和
    }
    int maxWealth = wealth[1]; // 存储最大的财富和
    for (int i = 1; i < N; i++) {
        int N1, N2; // 父子关系
        cin >> N1 >> N2;
        familyWealth[N1] += wealth[N2]; // 累加小家庭的财富和
        maxWealth = max(maxWealth, familyWealth[N1]); // 更新最大的财富和
    }
    cout << maxWealth << endl; // 输出结果
    return 0;
}
```

## Java

```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int N = scanner.nextInt(); // 读取成员总数
        int[] wealth = new int[N + 1]; // 存储每个成员的财富值
        int[] familyWealth = new int[N + 1]; // 存储每个小家庭的财富和
        for (int i = 1; i <= N; i++) {
            wealth[i] = scanner.nextInt(); // 读取每个成员的财富值
            familyWealth[i] = wealth[i]; // 初始化每个小家庭的财富和
        }
        int maxWealth =  wealth[1] ; // 存储最大的财富和，如果只有一个成员，则最大财富和为该成员财富
```

```
        // 当成员数大于1时，才执行读取父子关系的循环
        for (int i = 1; i < N; i++) {
            int N1 = scanner.nextInt();
            int N2 = scanner.nextInt(); // 读取父子关系
            familyWealth[N1] += wealth[N2]; // 累加小家庭的财富和
            maxWealth = Math.max(maxWealth, familyWealth[N1]); // 更新最大的财富和
        }
        System.out.println(maxWealth); // 输出结果
    }
}
```

## javaScript

```
const readline = require('readline').createInterface({
    input: process.stdin,
    output: process.stdout
});

let input = [];
readline.on('line', (line) => {
    input.push(line);
}).on('close', () => {
    const N = parseInt(input[0]); // 成员总数
    const wealth = input[1].split(' ').map(Number); // 存储每个成员的财富值
    wealth.unshift(0); // 为了使数组下标从1开始
    const familyWealth = [...wealth]; // 存储每个小家庭的财富和
    let maxWealth = wealth[1]; // 存储最大的财富和
    for (let i = 2; i < N + 1; i++) {
        const [N1, N2] = input[i].split(' ').map(Number); // 父子关系
        familyWealth[N1] += wealth[N2]; // 累加小家庭的财富和
        maxWealth = Math.max(maxWealth, familyWealth[N1]); // 更新最大的财富和
    }
    console.log(maxWealth); // 输出结果
});
```

## Python

```python
N = int(input()) # 成员总数
wealth = list(map(int, input().split())) # 存储每个成员的财富值
wealth.insert(0, 0) # 为了使数组下标从1开始
familyWealth = wealth.copy() # 存储每个小家庭的财富和
maxWealth =  wealth[1] # 存储最大的财富和
for _ in range(N - 1):
    N1, N2 = map(int, input().split()) # 父子关系
    familyWealth[N1] += wealth[N2] # 累加小家庭的财富和
    maxWealth = max(maxWealth, familyWealth[N1]) # 更新最大的财富和
print(maxWealth) # 输出结果
```

## C语言

```c
#include <stdio.h>

int main() {
    int N; // 成员总数
    scanf("%d", &N);

    int wealth[N + 1]; // 存储每个成员的财富值
    int familyWealth[N + 1]; // 存储每个小家庭的财富和
    for (int i = 1; i <= N; i++) {
        scanf("%d", &wealth[i]); // 读取每个成员的财富值
        familyWealth[i] = wealth[i]; // 初始化每个小家庭的财富和
    }

    int maxWealth = wealth[1]; // 存储最大的财富和
    for (int i = 1; i < N; i++) {
        int N1, N2; // 父子关系
        scanf("%d %d", &N1, &N2);
        familyWealth[N1] += wealth[N2]; // 累加小家庭的财富和
        if (familyWealth[N1] > maxWealth) {
            maxWealth = familyWealth[N1]; // 更新最大的财富和
        }
    }

    printf("%d\n", maxWealth); // 输出结果
    return 0;
}
```

## 完整用例

### 用例1

```
4
100 200 300 400
1 2
1 3
2 4
```

### 用例2

```
1
1000000
```

### 用例3

```
3
1 2 3
1 2
2 3
```

### 用例4

```
3
1 5 5
1 2
1 3
```

### 用例5

```
3
1000000 1000000 1000000
1 2
2 3
```

### 用例6

```
3
1 0 1
1 2
1 3
```

### 用例7

```
20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1 2
2 3
3 4
4 5
5 6
6 7
7 8
8 9
9 10
10 11
11 12
12 13
13 14
```

### 用例8

```
15
100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400 1500
1 2
1 3
2 4
2 5
3 6
3 7
4 8
4 9
5 10
5 11
6 12
6 13

```

### 用例9

```
10
0 0 0 0 0 1000000 0 0 0 0
1 2
2 3
3 4
4 5
5 6
6 7
7 8
8 9
9 10
```

### 用例10

```
10
123 234 345 456 567 678 789 890 901 101
1 2
2 3
3 4
4 5
5 6
6 7
7 8
8 9
9 10
```

### 文章目录


- 华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷
- 题目描述
- 输入描述
- 输出描述
- 用例
- 解题思路
- C++

Java
javaScript
Python
C语言
完整用例
    用例1
    用例2
    用例3
    用例4
    用例5
    用例6
    用例7
    用例8
    用例9
    用例10