【题解】2023 牛客 NOIP 赛前集训营-普及组（第三场）

T1-减法和求余

求余题

考察求余的性质

观察到 $a\%b = c$ 的式子中一定有 $c < b$，也就是说求余的结果一定小于除数。假设给定的 n 个数字的最小值是 min，那么假设 min 作为除数，那么此题的答案一定小于 min，所以我们应该让 min 作为被除数。也就是说用 min 求余所有其它数字。

再观察求余的性质，观察到 $a\%b = c$ 的式子中，假如 $a < b$，则 $a\%b = a$。因为 min 小于其它所有数字，所以用 min 做被除数去求余其它所有数字，得到的答案是 min。

本题的关键：每个数字只出现一次。

所以本题等价于求解最小值，是一道 for 循环入门题。 。

减法题

通过大眼观察法可以观察到样例的性质：所有数字之和减去两倍的最小值。

T2-回文串

要求至多改两个字符：

首先考虑先通过修改把字符串改为回文串，先统计一个修改次数。

之后考虑如何继续修改：

若没够两次修改，可以再在回文串基础上继续减小字典序。

- 若当前位置之前修改过，相当于只需要把对应位置都改为$'a'$，修改次数加一（因为已经在前面统计过一次了）

- 若当前位置没修改过，把两个位置都改为$'a'$，修改次数加二

时刻注意修改次数不能大于2。修改次数等于2时停止修改即可。

T3-涂色仪式

一个大小为 s 的连通块可以染 $s-1$ 个 （连通是指题目描述的边，即都为白色且和是质数），因为可以每次把叶子染了然后删掉，最后剩下一个。

所以这个问题和树形态无关，初始答案为 $n-1$ 然后减去有多少个不满足条件的边即可。

本题需要用到质数筛对判断素数进行预处理，使得时间复杂度降到 $O(nlogn)$ 或以下。

```
#include <iostream>

using namespace std;

const int maxn = 2e6;

bool vis[2000005];

int prime[2000005], a[maxn], cnt;

void init(){

    for(int i = 2; i < maxn; i++){

        if(!vis[i]){
```

```
            prime[++cnt] = i;
        }
        for(int j = 1; j <= cnt && i * prime[j] < maxn; j++){
            vis[i*prime[j]] = 1;
            if(i % prime[j] == 0)    break;
        }
    }
}
int main(){
    init();
    int n, ans;
    cin >> n;
    ans = n - 1;
    for (int i = 1; i <= n; i++)   cin >> a[i];
    int ww, yy;
    for (int i = 1; i < n; i++) {
        cin >> ww >> yy;
        if (vis[a[ww] + a[yy]])    ans--;
    }
    cout << ans << endl;
    return 0;
}
```

T4-除法来喽

当 $a_i < 5 \times 10^6$ 时，商的取值范围 $[1,5 \times 10^6]$。

- 因为这个题给定了除数的取值范围 $[1,10^6]$，可以考虑枚举除数。观察发现，除数把 a_i 的权值范围分成了若干段。假设除数是 3，那么 $0\sim 2$ 除以该除数变成同一个数字 0，$3\sim 5$除以该除数变成同一个数字 1，$6\sim 8$ 除以该除数变成同一个数字 2。以此类推。

- 考虑枚举除数统计答案。详细的说，枚举除数 i，$ans[k](k \geq 0)$ 加上序列 a 在 $[i*k\sim i*(k+1))$ 的出现次数，出现次数用前缀和快速统计。

- 上述做法存在一个细节问题，例如 $a_i = 3$，枚举 $i = 2$ 时，会使得 $ans[1]$ 加上 1，枚举 $i = 3$ 时，会使得 $ans[1]$ 加上 1，同一个 a_i 多次产生相同的商，计算重复。

- 考虑如何修复该细节问题，设 p_j 表示商为 j 的当前被除数范围。例如枚举除数 $i = 2$ 时，$p_1 = [2,3]$。当枚举除数$i = 3$时，$p_1 = [3,5]$。重复部分是 $[3,3]$，该部分不重复计数即可。

- ans 数组的最大值即为答案。

```
#include<bits/stdc++.h>
using namespace std ;
int main() {
    std::ios::sync_with_stdio(false) , cin.tie(0) ;
    int n ;
    cin >> n ;
    vector<int> a(n + 1 , 0) ;
    for(int i = 1 ; i <= n ; i ++)  cin >> a[i] ;
    const int up = 5e6 ;
    const int R = 1e6 ;
```

```
int mx = 0 ;
for(int i = 1 ; i <= n ; i ++)  mx += (a[i] < R) ;
vector<int> pre(up + 1 , 0) ;
for(int i = 1 ; i <= n ; i ++)  pre[a[i]] += 1 ;
for(int i = 2 ; i <= up ; i ++)  pre[i] += pre[i - 1] ;
auto query = [&](int l , int r) {
    return pre[r] - pre[l - 1] ;
} ;
vector<int> t(up + 1 , 0) ;
vector<pair<int , int>> p(up + 1 , {-1 , -1}) ;
for(int i = 1 ; i <= R ; i ++) {
    for(int j = i , d = 1 ; j <= up ; j += i , d += 1) {
        int L = p[d].first ;
        int R = p[d].second ;
        int L2 = j ;
        int R2 = min(j + i - 1 , up) ;
        if(R < L2) {
            p[d] = {L2 , R2} ;
            t[d] += query(L2 , R2) ;
        } else {
            if(R < R2) {
                p[d] = {L2 , R2} ;
                t[d] += query(R + 1 , R2) ;
```

```
                }
            }
        }
    }
    mx = max(mx , *max_element(t.begin() + 1 , t.end())) ;
    for(int i = 0; i < t.size(); i++) {
        if(t[i] == mx) {
            cout << i << " ";
        }
    }
    cout << mx << '\n' ;
}
```