E1H 任务达人莫卡 I

题目描述

莫卡穿越到了一个新的世界并且化身成了名为任务达人的 NPC。

现在莫卡手里有 n 个任务，编号为 $1,2,\ldots,n$，难度分别为$a_1,a_2,\ldots,a_n$。对于每一名前来领取任务的旅行者，莫卡会从中任意选取 3 个不同的任务，设编号为 i,j,k（$i\neq j,j\neq k,k\neq i$）。但是每个旅行者对任务的难度有独特的要求，他们会提出两个参数 l 和 r，要求 $2\cdot a_i\leq a_j\leq l\cdot a_i$ 且 $2\cdot a_j\leq a_k\leq r\cdot a_j$。现在莫卡想知道，对于每位旅行者，满足其难度要求的任务选择方案有多少种？由于答案可能很大，你只需要输出答案对 10^9+7 取模后的结果。

两个选择方案（设任务编号分别为 i,j,k 和 i',j',k'）被视作不同的方案当且仅当 $i\neq i'$ 或 $j\neq j'$ 或 $k\neq k'$。

输入格式

第一行一个正整数 n（$3\leq n\leq 10^5$），表示任务的数量。

第二行 n 个正整数 $a_1,a_2,\ldots,a_n$（$1\leq a_i\leq 10^3$），表示每个任务的难度。

第三行一个正整数 q（$1\leq q\leq 10^5$），表示旅行者的数量。

接下来 q 行每行两个正整数l_i,r_i（$2\leq l_i,r_i\leq 10^3$），表示第 i 个旅行者提出的两个参数。

输出格式

输出 q 行，第 i 行一个非负整数，表示满足第 i 个旅行者难度要求的选择方案数对10^9+7 取模后的结果。

题解思路

首先，这里我们读入可以大概确定a_j的范围，然后遍历a_j，注意到a_j一旦确定，那么a_i和a_k的范围就可以定下来，但是有一个问题：

虽然a_i和a_k的范围定下来了,但是怎么确定数组a当中有多少个数字在这个范围内呢，确定一个范围就遍历一遍数组a么?那复杂度就是O(nq),这样可能TLE。

为了解决这个问题，可以考虑提前"打表"，我们可以先利用数组a读入x，然后a[x]++，即记录每个数字出现的次数，这样也就排好序了（没出现的数字比如m就a[m] = 0 ），然后遍历一遍数组a，用另一个数组b[j]来记录读入的n个数字当中有多少个数字在[1, j]中，这样可以通过b[j+k] - b[j] 来求读入的n个数字当中有多少个数字在[j, j+k]中。

这样做完后，我们只需要遍历所有可能的a_j，然后根据a_i和a_k的范围结合数组b，实现目的。比如我们求出a_i的范围是[p,q],这里要先对p，q向上取整得到p_1,q_1，然后满足条件的a_i就是q_1-p_1(可能不存在，即小于等于0);

至此问题解决，那么这个问题的复杂度是多少呢？因为"打表"了，所以只需要遍历一次数组a，复杂度也就是O（n+q）。

代码

```
#include<stdio.h>
#define M 1000000007
int a[1005], num = 0;
struct f{
```

```
    int front;
};
struct f b[1005];
int main(){
    int n, i, j, q, l, r, x, y, max = -1, k, h, u, p, much = 0, flag = 0;
    long long s, d;
    scanf("%d",&n);
    for( i = 0 ; i < n ; i++ ){
        scanf("%d",&x);
        if( x > max )max = x;
        a[x]++;
    }
    for( i = 1 ; i <= max+1 ; i++ ){
        b[i].front = much;
        if( a[i] > 0 )much += a[i];
    }
    scanf("%d",&q);
    for( i = 0 ; i < q ; i++ ){
        scanf("%d%d",&l,&r);
        p = (max+1)/2;
        for( j = 2 ; j <= p ; j++ ){
            flag = a[j];
            if( flag > 0 ){
                x = (j+l-1)/l;
                y = j/2+1;
                h = ( r*j < max ? r*j : max );
                u = j*2;
                if( j*2 > max )continue;
                s = b[y].front - b[x].front;
                d = b[h+1].front - b[u].front;
//              printf("%d %d\n",s,d);
                if( s > 0 && d > 0 )num = (num + (((flag*s)%M)*d)%M)%M;
            }
        }
        printf("%d\n",num);
        num = 0;
    }
    return 0;
}
```