TỔNG QUAN ĐÈ THI

| Bài | Tên bài | Tên file | Tên file input | Tên file output | Điểm |
|-----|------------------------|-----------|----------------|-----------------|------|
| 1 | Bài 1. Số gần hoàn hảo | GНН.* | GHH.INP | GHH.OUT | 5 |
| 2 | Bài 2. Độ cao | DOCAO.* | DOCAO.INP | DOCAO.OUT | 7 |
| 3 | Bài 3. Trò chơi | TROCHOI.* | TROCHOLINP | TROCHOLOUT | 5 |
| 4 | Bài 4. Phần thưởng | BONUS.* | BONUS.INP | BONUS.OUT | 3 |

Dấu * là PY hoặc CPP tùy theo ngôn ngữ lập trình là PYTHON hay C++

BÀI 1. SỐ GẦN HOÀN HẢO

* Thuật toán đề xuất:

Subtask 1: $N \leq 10^3$:

Duyệt từng phần tử A[i], sau đó tính tổng các ước của A[i], so sánh tổng đó với A[i] để đưa ra kết luận A[i] có gần hoàn hảo không?

Độ phức tạp: $O(N * \sqrt{A[i]})$

Subtask 2: $N \leq 10^6$:

Điều kiện X là số gần hoàn hảo là $2*X \le T$, trong đó T là tổng các ước của X, suy ra điều kiện để X là số gần hoàn hảo là $X \le T1$, trong đó T1 là tổng các ước của X (không tính giá trị X)

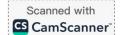
Cần chuẩn bị trước một mảng c[] với ý nghĩa: c[x] = tổng các ước (không kể chính nó) của x, cách tạo mảng c[] giống như sàng nguyên tố (tham khảo code).

Với mỗi giá trị x trong dữ liệu đầu vào chỉ cần kiểm tra $c[x] \ge x$ để in ra 1 hay 0.

Độ phức tạp: $O(10^6 * log 10^6)$

Chủ đề: Số học

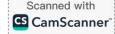
```
#include <bits/stdc++.h>
#define N int(1e6)
using namespace std;
int c[N+3];
int n;
void init()
```



```
fill(c+1,c+N+1,1);
    for (int i = 2; i*i <= N; i++)
        for (int j=i*i; j <= N; j+= i)
            c[j] += i;
            if (i*i != j) c[j] += j/i;
        }
int main()
{
    init();
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    freopen("ghh.inp","r",stdin);
    freopen("ghh.out","w",stdout);
    cin >> n;
    for (int x,i=1; i<=n; i++)
    {
        cin >> x;
        if (c[x] >= x) cout << 1 << '\n';
        else cout << 0 << '\n';
    return 0;
```

```
N = int(1e6)
def init():
    k = int(N ** 0.5)
    for i in range(2,k+1):
        for j in range(i*i,N+1,i):
            c[j] += i;
            if (i*i != j): c[j] += j/i;
import sys
sys.stdin = open("ghh.inp","r");
sys.stdout = open("ghh.out","w");
c = [1] * (N+3)
init()
n = int(input())
a = list(map(int,input().split()))
for x in a:
    if (c[x] >= x): print(1)
    else: print(0);
```

* Ghi chú: Cách giải này đối với Python khi chấm bằng phần mềm Themis cần đặt thời gian lên 3 giây mới qua được hết test.



BÀI 2. ĐỘ CAO

* Thuật toán đề xuất:

Bài toán đơn giản chỉ là duyệt giá trị x từ 2 đến n, nếu x thỏa mãn điều kiện đề bài thì in ra trị x và tăng biến đếm số lượng x thỏa mãn.

Subtask 1: có 20/35 test ứng với 4 điểm n ≤ 10³
 Kiểm tra x có là số nguyên tố bằng 1 hàm kiểm tra nguyên tố.

Độ phức tạp: $O(n * \sqrt{n})$

Subtask 2, 3: có 15/35 test ứng với 3 điểm n ≤ 10⁶
 Kiểm tra x có là số nguyên tố bằng cách sàng số nguyên tố.

Độ phức tạp thuật toán:

- Độ phức tạp sàng số nguyên tố: O(nlogn)
- \circ Độ phức tạp duyệt tìm x: O(n)
- → Độ phức tạp: O(nlogn)

Chủ đề: Số học

```
#include <bits/stdc++.h>
#define N int(1e6)
using namespace std;
int p[N+3];
void sang_ngto()
    fill(p+2,p+N+1,1);
    for (int i=2; i*i<=N; i++)
        if (p[i] == 1)
            for (int j=i*i; j<=N; j+=i) p[j] = 0;
int tong_cs(int x)
    int s = 0;
    while (x > 0)
        s += x \% 10;
        x /= 10;
    return s;
int main()
    sang_ngto();
    freopen("docao.inp","r",stdin);
   freopen("docao.out","w",stdout);
    int h, n, d = 0;
    cin >> n >> h;
    for (int i = 2; i <= n; i++)
```



```
if (p[i] == 1 && tong_cs(i) == h)
{
     cout << i << '\n';
     d++;
}
cout << d;
return 0;
}</pre>
```

```
N = int(1e6)
def sang_ngto():
    p[0] = p[1] = 0
    k = int(N ** 0.5)
    for i in range(2,k+1):
        if (p[i] == 1):
             for j in range(i*i,N,i):
                 p[j] = 0;
def tong_cs(x):
    s = 0;
    while (x > 0):
        s += x \% 10;
        x //= 10;
    return s;
import sys
sys.stdin = open("docao.inp","r")
sys.stdout = open("docao.out","w")
p = [1] * (N+3)
sang_ngto()
n = int(input())
h = int(input())
d = 0
for i in range(2,n+1):
    if (p[i] == 1 \text{ and } tong_cs(i) == h):
        print(i);
        d += 1;
print(d)
```

BÀI 3. TRÒ CHƠI

- * Thuật toán đề xuất:
- Subtask 1: có 15/25 test, ứng với 3 điểm $n \le 10^3$

Duyệt hai vòng lặp lồng nhau để xét A_i và B_j , từ đó so sánh kết quả với $|A_i + B_j|$ để lưu lại kết quả nhỏ nhất.

Độ phức tạp: $O(n^2)$

Subtask 2: có 10/25 test, ứng với 3 điểm n ≤ 10⁶
 Sắp xếp hai dãy A và B tăng dần.



Sử dụng biến chỉ số l chạy từ đầu bên trái sang phải dãy A, biến chỉ số j chạy từ đầu bên phải sang trái dãy B.

Nhận xét: để $|A_l + B_r|$ đạt giá trị nhỏ nhất thì tổng $A_l + B_r$ càng gần 0 càng tốt. Các bước dưới đây được thực hiện liên tục cho đến khi l > n hoặc r < 0:

- Nếu $A_l + B_r = 0$ thì kết quả bằng 0 và kết thúc.
- Nếu A_l + B_r < 0 khi đó sẽ tăng giá trị tổng này để nó gần về 0 bằng cách tăng l
 thêm 1 đơn vị.
- Nếu A_l + B_r > 0 khi đó sẽ giảm giá trị tổng này đến nó gần về đến 0 bằng cách giảm r đi 1 đơn vị.
- So sánh kết quả với $|A_l + B_r|$ để cập nhật kết quả nhỏ nhất.

Độ phức tạp:

- Sắp xếp dãy với độ phức tạp là O(nlogn)
- Xử lý tìm kết quả với độ phức tạp là O(n)
- → độ phức tạp của cả bài toán là O(nlogn)

Chủ đề: Hai con trỏ

```
#include <bits/stdc++.h>
#define N int(1e6)
#define inf int(2e9)
using namespace std;
int a[N+3],b[N+3];
int n, ans = inf;
int main()
{
    freopen("trochoi.inp","r",stdin);
    freopen("trochoi.out", "w", stdout);
    cin >> n;
    for (int i=1; i<=n; i++) cin >> a[i];
    for (int i=1; i<=n; i++) cin >> b[i];
    sort(a+1,a+n+1);
    sort(b+1,b+n+1);
    int l = 1, r = n;
    while (1 <= n \&\& r > 0)
    {
        ans = min(ans,abs(a[1] + b[r]));
        if (ans == 0) break;
        if (a[1] + b[r] < 0) 1++;
        else r--;
    cout << ans;
    return 0;
```



```
import sys
sys.stdin = open("trochoi.inp","r")
sys.stdout = open("trochoi.out","w")
n = int(input())
a = list(map(int,input().split()))
b = list(map(int,input().split()))
a.sort(); b.sort()
a = [0] + a; b = [0] + b;
l = 1; r = n; ans = int(2e9)
while (1 \le n \text{ and } r > 0):
    ans = min(ans,abs(a[1] + b[r]));
    if (ans == 0): break;
    if (a[1] + b[r] < 0): 1 += 1
    else: r -= 1
print(ans)
```

BÀI 4. PHẦN THƯỜNG

* Thuật toán đề xuất:

Subtask 1: có 10/30 test, ứng với 1 điểm $n \le 10^3$

Duyệt vòng lặp thứ nhất $i: 1 \rightarrow n$

Duyệt vòng lặp thứ hai $j: i \rightarrow 1$ tìm giá trị lớn nhất và nhỏ nhất trong đoạn từ a[i] đến a[i] và bổ sung hiệu của giá trị lớn nhất và nhỏ nhất đó vào kết quả.

Độ phức tạp: $O(n^2)$

Subtask 2: có 20/30 test, ứng với 1 điểm $n \le 10^6$

Trước hết hãy xét một ví dụ: có dãy 3 1 7 2

Trong số đoan có 1 phần tử:

(3):
$$3-3$$
; (1): $1-1$; (7): $7-7$; (2): $2-2$

Trọng số đoạn có 2 phần tử:

$$(3, 1): 3 - 1; (1, 7): 7 - 1;$$
 $(7, 2): 7 - 2$

$$(7, 2): 7 - 2$$

Trọng số đoạn có 3 phần tử:

$$(3, 1, 7): 7 - 1;$$
 $(1, 7, 2): 7 - 1;$

Trong số đoạn có 4 phần tử:

$$(3, 1, 7, 2): 7 - 1$$

Tổng trọng số các đoạn:

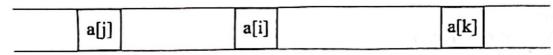
$$(3-3)+(1-1)+(7-7)+(2-2)+(3-1)+(7-1)+(7-2)+(7-1)+(7-1)+(7-1)$$

$$= 2 * 3 + 1 * 1 + 6 * 7 + 1 * 2 - 1 * 3 - 6 * 1 - 1 * 7 - 2 * 2 = 31$$

Từ biểu thức trên nhận thấy:

- + 2 * 3: thể hiện 3 là giá trị lớn nhất của 2 đoạn con liên tiếp là (3), (3, 1)
- + 1 * 1: thể hiện 1 là giá trị lớn nhất của 1 đoạn con liên tiếp là (1)
- + 6 * 7: thể hiện 7 là giá trị lớn nhất của 6 đoạn con liên tiếp là (7), (1, 7), (7, 2), (3, 1, 7), (1, 7, 2) và (3, 1, 7, 2)
- + 1 * 2: thể hiện 2 là giá trị lớn nhất của 1 đoạn con liên tiếp là (2)
- 1 * 3: thể hiện 3 là giá trị nhỏ nhất của 1 đoạn con liên tiếp là (3)
- 6 * 1: thể hiện 1 là giá trị nhỏ nhất của 6 đoạn con liên tiếp là (1), (3, 1), (1, 7), (3, 1, 7), (1, 7, 2) và (3, 1, 7, 2)
- 1 * 7: thể hiện 7 là giá trị nhỏ nhất của 1 đoạn con liên tiếp là (7)
- 2 * 2: thể hiện 2 là giá trị nhỏ nhất của 2 đoạn con liên tiếp là (2), (7, 2)

Tổng quát: với mỗi giá trị a[i] cần đếm xem nó lớn nhất trong bao nhiều đoạn con liên tiếp và nhỏ nhất trong bao nhiều đoạn con liên tiếp.



Nhìn trên hình:

- với mỗi giá trị a[i] cần tìm a[j] bên trái gần với a[i] nhất sao cho a[i] < a[j], suy ra trong đoạn con a[j+1], a[j+2], ..., a[i] mọi giá trị đều nhỏ hơn a[i], độ dài đoạn con này là x = i j;
- với mỗi giá trị a[i] cần tìm a[k] bên phải gần với a[i] nhất sao cho a[i] < a[k], suy ra trong đoạn con a[i], a[i+1], ..., a[k-1] mọi giá trị đều nhỏ hơn a[i], độ dài đoạn con này là y=k-i.

Từ đó suy ra số đoạn con mà a[i] đạt giá trị lớn nhất là x * y

Tương tự lập luận ở trên cũng có thể tìm đoạn số đoạn con mà a[i] đạt giá trị nhỏ nhất.

Sử dụng 4 mảng $maxL[\], maxR[\], minL[\], minR[\]$ với ý nghĩa như sau:

- maxL[i] là vị trí của phần từ bên trái gần với a[i] nhất mà có giá trị > a[i];
- maxR[i] là vị trí của phần tử bên phải gần với a[i] nhất mà có giá trị > a[i];
- minL[i] là vị trí của phần tử bên trái gần với α[i] nhất mà có giá trị < α[i];
- minR[i] là vị trí của phần tử bên phải gấn với a[i] nhất mà có giá trị < a[i].



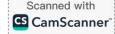
Giá trị của 4 mảng này có thể tìm được bằng cách sử dụng cấu trúc dữ liệu Stack với độ phức tạp là O(n).

Lưu ý: trường hợp có 2 phần tử trở lên giá trị cùng lớn nhất (hoặc nhỏ nhất) trong 1 đoạn liên tiếp nào đó, ví dụ dãy: 2.73.74, giá trị a[2] và a[4] cùng lớn nhất trong đoạn từ a[1] đến a[5] thì maxL[2] = 0 còn maxL[4] = 2 bởi nếu maxL[4] = 0 thì sẽ bị hiệu 7-2 của đoạn từ a[1] đến a[5] sẽ bị lặp nhiều lần, dẫn đến kết quả sai.

Chương trình mẫu đã xử lý cả tình huống cần lưu ý trên.

Độ phức tạp: O(n)Chủ đề: Duyệt + CTDL

```
#include <bits/stdc++.h>
#define N 1000000
#define 11 long long
#define inf int(1e9)
using namespace std;
11 a[N+3];
11 minL[N+3], maxL[N+3], minR[N+3], maxR[N+3];
int n;
ll ans;
stack <int> sL, sR;
int main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    freopen("weight.inp","r",stdin);
freopen("weight.out","w",stdout);
    cin >> n;
    for (int i=1; i<=n; i++) cin >> a[i];
    a[0] = a[n+1] = inf;
    minL[0] = 0; minR[n+1] = n+1;
    sL.push(0);
    for (int i=1; i<=n; i++)
        while (a[i] > a[sL.top()]) sL.pop();
         maxL[i] = sL.top();
         sL.push(i);
    sR.push(n+1);
    for (int i=n; i>0; i--)
        while (a[i] >= a[sR.top()]) sR.pop();
        maxR[i] = sR.top();
        sR.push(i);
    a[0] = a[n+1] = 0;
```



```
while (!sL.empty()) sL.pop();
while (!sR.empty()) sR.pop();
sL.push(0);
for (int i=1; i<=n; i++)
{
    while (a[i] < a[sL.top()]) sL.pop();</pre>
    minL[i] = sL.top();
    sL.push(i);
}
sR.push(n+1);
for (int i=n; i>0; i--)
    while (a[i] <= a[sR.top()]) sR.pop();</pre>
    minR[i] = sR.top();
    sR.push(i);
for (int i=1; i<=n; i++)
    ans += (i - maxL[i]) * (maxR[i] - i) * a[i];
    ans -= (i - minL[i]) * (minR[i] - i) * a[i];
cout << ans;
return 0;
```

```
import sys
sys.stdin = open("bonus.inp","r")
sys.stdout = open("bonus.out","w")
n = int(input())
a = list(map(int,input().split()))
a = [0] + a + [0]
minL = [0] * (n + 3)
minR = [0] * (n + 3)
maxL = [0] * (n + 3)
maxR = [0] * (n + 3)
a[0] = a[n+1] = int(1e9)
sL = []; sR = []
sL.append(0)
for i in range(1,n+1):
    while (a[i] > a[sL[len(sL)-1]]): sL.pop();
    maxL[i] = sL[len(sL)-1];
    sL.append(i);
sR.append(n+1);
for i in range(n, 0, -1):
    while (a[i] >= a[sR[len(sR)-1]]): sR.pop();
    maxR[i] = sR[len(sR)-1];
    sR.append(i);
a[0] = a[n+1] = 0
sL.clear(); sR.clear();
```