CÂU LẠC BỘ LẬP TRÌNH CODER QO

ĐỀ MINH HỌA - ĐÁP ÁN

CUỘC THI KHOA HỌC KHU VỰC PHÍA TÂY HÀ NỘI 2023

HanoiSTEM West Challenge 2023

Ngày thi: *18/11/2023*

Thời gian làm bài: *300 phút*

1 Commander Coder QO - Tin (*1 điểm*)

2 Commander Coder QO- Toán (*1 điểm*)

Trong vũ trụ rộng lớn, sâu trong trạm vũ trụ của chủ tịch Coder QO, tồn tại một cấu trúc bí ẩn và phức tạp được biết đến là thiết bị hủy diệt hỗ trợ học tập (Coder QO doomsday device). Thiết bị độc đáo này chiếm một phần quan trọng của không gian nội thất của trạm, dẫn đến một bố cục công việc có hình thức khá không thông thường. Các khu vực làm việc này được xếp theo dạng một mẫu tam giác, và người lao động chăm chỉ được gán các ID số duy nhất, bắt đầu từ góc và mở rộng ra ngoài. Việc gán ID này tuân theo một quy tắc cụ thể, có thể được mô phỏng theo cách sau:

|..

7 ..

4 8 ..

2 5 9 ..

1 3 6 10 ..

Mỗi ô trong cấu trúc tam giác này có thể được xác định bằng một bộ tọa độ (x, y) với 'x' biểu thị khoảng cách từ tường dọc và 'y' đại diện cho độ cao từ mặt đất. Ví dụ, một người lao động tại tọa độ (1, 1) có ID là 1, trong khi một người lao động thỏ tại (3, 2) sở hữu ID là 9. Tương tự, người lao động thỏ tìm thấy tại (2, 3) có ID là 8. Hệ thống đánh số độc đáo này tiếp tục mãi mãi, và chủ tịch Coder QO liên tục thêm những người lao động mới vào trạm.

Nhiệm vụ của bạn là tạo ra một chương trình có tên là "Commander Coder QO", khi nhập vào các tọa độ (x, y), sẽ trả về ID của người lao động thỏ được gán cho vị trí cụ thể đó.

Rằng buộc:

- 50% số test ứng với: x và y đồng thời $\leq 10^6$.
- 50% số test còn lại ứng với: x và y đồng thời $\leq 10^{18}$

Ví dụ

Input	Output
1 1	1
3 2	9
3 3	13

Input	Output
2 3	8
4 4	25

2.1 Lời giải trâu bò

- Ta sử dụng vòng lặp để tìm giá trị ở ô (y, 1), sau đó tính giá trị ở ô (y, x).

Độ phức tạp O(x + y).

Python

```
x, y = map(int, input().split())
su1 = 1

# tim gia tri o o(y, 1)
for i in range(1, y):
  su1 += i
su2 = 0
# tim gia tri o o (y, x)
for i in range(y + 1, x + y):
  su2 += i

print(su1 + su2)
```

cpp

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
signed main() {
    ll x, y;
    cin >> x >> y;
    ll su1 = 1;
    for (ll i = 1; i < y; i++)
        su1 += i;
    ll su2 = 0;
    for (ll i = y + 1; i < x + y; i++)
        su2 += i;
    cout << su1 + su2;
}
```

2.2 Lời giải cải tiến

Thay vì sử dụng vòng lặp để tính toán giá trị của ô (y, 1) và ô (y, x) ta có thể tính toán như sau.

- Kết quả vòng lặp thứ nhất (su1) bằng:

$$su1 = 1 + 1 + 2 + 3 + ... + (y - 1) = 1 + \frac{y \times (y - 1)}{2}$$

- Kết quả vòng lặp thứ 2 (su2) bằng:

$$su2 = y+1+y+2+....+...+(x+y-1) = y\times(x-1)+\frac{((x-1)+1)\times(x-1)}{2} = y\times(x-1)+\frac{(x)\times(x-1)}{2}$$

Vậy kết quả của bài toán được tính bằng công thức:

$$su1 + su2 = 1 + \frac{y\times(y-1)}{2} + y\times(x-1) + \frac{x\times(x-1)}{2}$$

Độ phức tạp tổng quát O(1)

python

```
x, y = map(int, input().split())

print(1 + (y * (y - 1) // 2) + y * (x - 1) + x * (x - 1) // 2)
```

Cpp

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
signed main() {
    ll x, y;
    cin >> x >> y;
    cout << 1 + (y * (y - 1) / 2) + y * (x - 1) + x * (x - 1) / 2;
}
```

3 Số Pi - Tin (*1 điểm*)

Số Pi có thể được tính toán bằng cách sử dụng công thức sau:

$$\pi = \frac{4}{1+\frac{1}{3+\frac{4}{5+\frac{9}{7+\frac{16}{\cdots}}}}}$$

Viết một chương trình để tính giá trị của sử dụng công thức trên.

Input: Số lượng phép cộng trong mẫu số

Output: Giá trị ước lượng của số (làm tròn đến 5 chữ số sau dấu thập phân)

Ví dụ:

Nếu đầu vào là 2, thì kết quả là:

$$\pi = \frac{4}{1+\frac{1}{3+4}}$$

Rằng buộc

- 50% số test đầu vào ≤ 20
- 50% số test đầu vào $\leq 10^7$

Ví dụ

Input	Output
2	3.5
8	3.14161
20	3.14159

Lời giải

Để bài toán trở lên đơn giản ta sẽ tính kết quả từ dưới lên trên bằng cách sử dụng vòng lặp.

- Với mỗi gạch ngang ở phép chia ta sẽ coi là một hàng

Vậy hàng cuối sẽ được tính bằng công thức: (n * 2) - 1 + x.

sau khi biết được các chỉ số của hàng cuối ta cứ tính lên trên dần và cuối cùng chỉ cần lấy 4 cho giá trị vừa tính được sẽ in ra được kết quả.

Độ phức tạp tổng quát: O(n).

cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
signed  main() {
    ll n; cin >> n;
    ll x = 1, k = 3;
    for (int i = 0; i < n - 1; i++) {
        x = x + k;
        k += 2;
    }
    k -= 2;
    double kq = x + (k * 1.0);
    for (int i = 0; i < n - 1; i++) {
        kq = (k - 2) + ((x - k) / (kq * 1.0));
        x -= k;
        k -= 2;
    }
    cout << fixed << setprecision(5) << 4 / kq;
}
```

Pyhon

```python
n = int(input())
x, k = 1, 3
for i in range(n - 1):
    x = x + k;
```

```
    k += 2
k -= 2
kq = x + k
for i in range(n - 1):
    kq = (k - 2) + (x - k) / kq;
    k, x = k - 2, x - k
kq = 4 / kq
print('{0:.{1}f}'.format(kq, 5))
```

4 Số Pi - Toán (*1 điểm*)

Sử dụng kiến thức về phân tích (chỉ toán học), chứng minh rằng công thức trên được ước lượng chính xác khoảng 10 chữ số thập phân với 20 phép cộng trong mẫu số. (Gợi ý: Liên quan đến kiến thức về dãy số).

Mình cũng không biết giải

5 Make It Good - Tin (*1 điểm*)

Bạn được cho một mảng a gồm n số nguyên. Bạn cần tìm độ dài của đoạn prefix (phần đầu nhỏ nhất) mà bạn cần xóa từ a để biến nó thành một mảng tốt. Hãy nhớ rằng đoạn prefix của mảng $a = [a_1, a_2, ..., a_n]$ là một mảng con chứa một số phần tử đầu tiên: đoạn prefix của mảng a có độ dài k là mảng $[a_1, a_2, ..., a_k] (0 \leq k \leq n)$.

Mảng b có độ dài m được gọi là tốt, nếu bạn có thể tạo ra một mảng không giảm $c = [c_1 \leq c_2 \leq \cdots \leq c_m]$ từ nó, lặp lại thao tác sau m lần (ban đầu, c là rỗng):

Chọn một trong hai phần tử đầu hoặc cuối của b, loại bỏ nó khỏi b và thêm vào cuối mảng c. Ví dụ, nếu bạn thực hiện 4 thao tác: chọn b_1, sau đó b_m, sau đó b_{m-1} và cuối cùng là b_2, thì b trở thành $[b_3, b_4, ..., b_{m-3}]$ và $c = [b_1, b_m, b_{m-1}, b_2]$.

Xem xét ví dụ sau: $b = [1, 2, 3, 4, 4, 2, 1]$. Mảng này là tốt vì bạn có thể tạo ra mảng không giảm c từ nó bằng dãy thao tác sau:

- Chọn phần tử đầu tiên của b, vì vậy $b = [2, 3, 4, 4, 2, 1], c = [1]$.
- Chọn phần tử cuối cùng của b, vì vậy $b = [2, 3, 4, 4, 2], c = [1, 1]$.
- Chọn phần tử cuối cùng của b, vì vậy $b = [2, 3, 4, 4], c = [1, 1, 2]$.
- Chọn phần tử đầu tiên của b, vì vậy $b = [3, 4, 4], c = [1, 1, 2, 2]$.
- Chọn phần tử đầu tiên của b, vì vậy $b = [4, 4], c = [1, 1, 2, 2, 3]$.
- Chọn phần tử cuối cùng của b, vì vậy $b = [4], c = [1, 1, 2, 2, 3, 4]$.

Chọn duy nhất phần tử của b, vì vậy $b = [], c = [1, 1, 2, 2, 3, 4, 4]$ — c là mảng không giảm.

Lưu ý rằng mảng chỉ gồm một phần tử cũng được coi là tốt.

In ra độ dài của đoạn prefix ngắn nhất của a cần xóa (hoặc không cần xóa) để biến a thành một mảng tốt.

Bạn cần trả lời t số lượng test case độc lập.

Input

- Dòng đầu tiên của input chứa một số nguyên t ($1 \leq t \leq 2 \cdot 10^4$) — số lượng test case. Tiếp theo là t test case.
- Mỗi test case bắt đầu bằng một số nguyên n ($1 \leq n \leq 2 \cdot 10^5$) — độ dài của mảng a. Dòng tiếp theo của test case chứa n số nguyên $a_1, a_2, ..., a_n (1 \leq a_i \leq 2 \cdot 10^5)$, trong đó ai là phần tử thứ i của a.

Đảm bảo tổng của n không vượt quá $2 \cdot 10^5 (\sum n \leq 2 \cdot 10^5)$.

Output

Cho mỗi test case, in ra câu trả lời: độ dài của đoạn prefix ngắn nhất cần xóa từ a để biến nó thành một mảng tốt.

Input

```
5
4
1 2 3 4
7
4 3 3 8 4 5 2
3
1 1 1
7
1 3 1 4 5 3 2
5
5 4 3 2 3
```

Output

```
0
4
0
2
3
```

5.1 Lời giải trâu bò

- Ta gọi x là số cần loại bỏ của a, chạy x từ 1 đến n đến khi tim được kết quả thì in ra và dừng chương trình.

5.2 Lời giải tối ưu

- Nhận thấy với nếu số cần loại bỏ nhỏ đúng thì số lượng số cần loại bỏ to hơn cũng đúng => Đây là cơ của để ta có thể tìm kiếm nhị phân.

Code cpp

```
#include <bits/stdc++.h>
#define fi first
#define se second
```

```cpp
using namespace std;

bool check(int k, vector<int> &a){
    int d = k + 1, c = a.size() - 1;
    vector<int> x, y;
    for(int i = k + 1; i <= a.size() - 1; i++)
        x.push_back(a[i]);
    sort(x.begin(), x.end());
    while(d <= c){
        if(a[d] >= a[c]){
            y.push_back(a[c]);
            c -= 1;
        }
        else{
            y.push_back(a[d]);
            d += 1;
        }
    }
    if(x == y)
        return true;
    return false;
}
signed main(){
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    int t; cin >> t;
    while(t--){
        int n; cin >> n;
        vector<int> a(n + 1, 0);
        for(int i = 1; i <= n; i++)
            cin >> a[i];
        int d = 0, c = n, res = 0;
        while(d <= c){
            int g = (d + c) / 2;
            if(check(g, a)){
                res = g;
                c = g - 1;
            }
            else
                d = g + 1;
        }
        cout << res << "\n";
    }
}
```

6 Phần thưởng

An là người thắng cuộc trong cuộc thi "Tìm hiểu Đoàn Thanh niên Cộng sản Hồ Chí Minh" và được nhận phần thưởng của Ban tổ chức. Ban tổ chức chuẩn bị một bảng kích thước m * n. Các dòng của bảng được đánh số từ 1 tới m, từ trên xuống dưới, dòng i ($1 \leq i \leq m$) có trọng số là a_i. Các cột của bảng được đánh số từ 1 tới n, từ trái qua phải, cột j ($1 \leq j \leq n$) có trọng số là b_j. Ô nằm trên giao của dòng i và cột j được gọi là ô (i,j) và trên ô đó ghi một số nguyên có giá trị $a_i + b_j$ ($1 \leq i \leq m; 1 \leq j \leq n$).

Để nhận phần thưởng, An được phép chọn một bảng con kích thước $w \times h$ chiếm trọn $w \times h$ ô của bảng và phần thưởng mà An nhận được sẽ có giá trị bằng tổng giá trị các ô nằm trong bảng con đó.

Yêu cầu: Hãy giá định giá trị phần thưởng lớn nhất mà An có thể nhận được?

Mô tả đầu vào

- Dòng đầu tiên chứa bốn số nguyên dương m,n,w,h ($w \leq m; h \leq n$).
- Dòng thứ hai chứa m số nguyên $a_1, a_2, ...a_m$
- Dòng thứ ba chứa n số nguyên $b_1, b_2, ...b_n$

Ràng buộc

- $1 \leq m, n \leq 10^5$
- $|a_i|, |b_j| \leq 10^6$;$\forall i, j : 1 \leq i \leq m, 1 \leq j \leq n$.

Mô tả đầu ra

Một số nguyên duy nhất là giá trị phần thưởng lớn nhất mà An có thể nhận được.

Test case mẫu

Input

```
3 4 2 2
1 -1 2
1 1 1 1
```

Output

```
6
```

Lời giải

Ta chỉ cần lấy tổng của w số liên tiếp lớn nhất trong mảng a và h số liên tiếp lớn nhất trong mảng b, ta gọi hai tổng ấy là x, y vậy kết quả bài toán sẽ là: x * h + y * w.

Để lấy tổng đoạn liên tiếp như vậy nhanh ta sẽ sử dụng kỹ thuật tổng tiền tố.

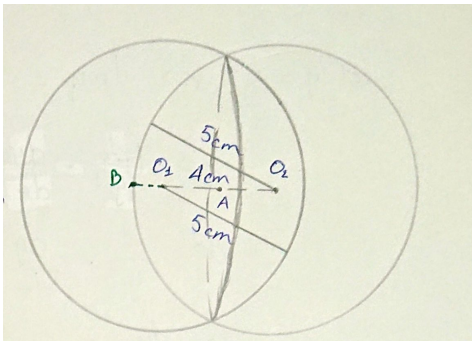
```
#include<bits/stdc++.h>
using namespace std;
#define int long long
#define f first
#define s second
```

```
const int ll = 1e6 + 1;
int m, n, w, h, a[ll], b[ll], suma[ll], sumb[ll];
void enter()
{
    cin >> m >> n >> w >> h;
    for(int i = 1; i <= m; i++)
    {
        cin >> a[i];
        suma[i] = suma[i - 1] + a[i];
    }
    for(int j = 1; j <= n; j++)
    {
        cin >> b[j];
        sumb[j] = sumb[j - 1] + b[j];
    }
}

void build()
{
        int res = -999999999999;
        for(int i = w; i <= m; i++)
            res = max(res, (suma[i] - suma[i - w]));
        int res2 = -999999999999;
        for(int i = h; i <= n; i++)
            res2 = max(res2, (sumb[i] - sumb[i - h]));
        cout << (res * h) + (res2 * w);
}
main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    // freopen("a.inp","r",stdin);
    // freopen("a.out","w",stdout);
    enter();
    build();

}
```

7 Quả cầu - Toán (*1 điểm*)

Được cho 2 quả cầu có bán kính R = 5 cm. Tâm của hai quả cầu này cách nhau 4 cm. Hãy tính thể tích của phần giao nhau giữa hai quả cầu và diện tích xung quanh của hình được tạo bởi hai quả cầu trên.

.
.
.
.



Lời giải

Cho A là trung điểm của đoạn thẳng OA, có chiều dài 5 cm. ta có thể chia phần giao nhau của các quả cầu thành hai phần nón cầu bằng cách sử dụng một mặt phẳng vuông góc với đoạn thẳng OA.

Đặt O B là bán kính của quả cầu ở đầu (quả cầu có trung tâm là O_2) chứa cả O_1 và A.

$$=> O_2B = O_1B + O_1O_2 => O_1B = O_2B - O_1O_2 = 5 - 4 = 1\text{cm}$$

$$=> AB = O_1A + O_1B = \frac{O_1O_2}{2} + O_1B = \frac{4}{2} + 1 = 3\text{cm}.$$

Thể tích của phần giao nhau của các quả cầu là tổng thể tích của hai phần nón cầu:

$$V = 2 \times \frac{\pi \times AB^2}{3}(3 \times O_2B - AB)$$

$$= 2 \times \frac{\pi \times 3^2}{3}(3 \times 5 - 3) = 2\pi \times (15 - 3) = 72\pi cm^3$$

Diện tích bề mặt của hình được tạo ra bởi các quả cầu có thể tính bằng cách trừ tổng diện tích bề mặt của các quả cầu cho diện tích bề mặt cong của hai phần nón cầu:

$$S = 2 \times 4\pi \times O_2B^2 - 2 \times 2\pi \times O_2B \times AB$$

$$S = 2 \times 4\pi \times 5^2 - 2 \times 2\pi \times 5 \times 3$$

$$= 4 \times 5 \times (2 \times 5 - 3) = 140\,\text{cm}^2$$

8 Tree - Tin (*1 điểm*)

CLB coder QO cung cấp sản phẩm là những chiếc cây bằng nhựa để trang trí trong gia đình. Những chiếc cây bao gồm n tán lá và n - 1 cành nối sao cho các tán lá đều liên thông với nhau. Để phục vụ công tác vận chuyển, các cành cây và tán lá đều có thể tháo lắp tùy ý thành các phần có kích thước nhỏ hơn, sau đó khách hàng sẽ tự lắp ráp các phần lại với nhau. Tuy nhiên, các khách hàng khó tính của công ty không muốn nhận được những phần cây có kích thước là số lẻ (số tán lá lẻ). Mặt khác, công ty lại muốn tháo được càng nhiều cành cây càng tốt để các phần cây nhỏ có thể được sắp xếp gọn gàng hơn trong khoang chứa hàng.

Yêu cầu: Hãy tìm cách tháo nhiều cành cây ra nhất có thể sao cho các phần cây còn lại đều có kích thước là số chẵn?

Mô tả đầu vào

- Dòng đầu tiên ghi số nguyên dương n - 1 dòng tiếp theo, mỗi dòng ghi một cặp số nguyên (u,v) - thể hiện một cành cây nối giữa hai tán lá u và v.

Ràng buộc :

- $1 \leq n \leq 10^5$

Mả đầu ra

Đưa ra số lượng cành nhiều nhất có thể tháo ra sao cho các phần còn lại của cây đều có kích thước chẵn. Nếu không thể tháo cành cây nào thì đưa ra -1.

Ví dụ

Input 1

```
3
1 2
1 3
```

Output 1

```
-1
```

Giai thích 1

Không thể bỏ đi cạnh nào cả.

Input 2

```
4
2 4
4 1
3 1
```

Output 2

```
1
```

Giai thích 2

Bỏ cạnh (1, 4)

```
#include<bits/stdc++.h>
#define int long long
#define endl "\n"
using namespace std;
const int ll = 1e6 + 7;
int n, children[ll], res = 0;
vector<int> adj[ll];
bool visited[ll];
void enter(){
    cin >> n;
```

```
    for(int i = 1; i <= n - 1; i++){
        int u, v; cin >> u >> v;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
}
void dfs(int u){
    visited[u] = true;
    children[u] = 1;
    for(auto v: adj[u]){
        if(!visited[v]){
            dfs(v);
            children[u] += children[v];
        }
    }
    if(children[u] % 2 == 0)
        res ++;
}
void build(){
    if(n % 2 != 0){
        cout << -1;
    }
    else{
        dfs(1);
        cout << res - 1;
    }
}
signed main(){
   // freopen("a.inp", "r", stdin);
   // freopen("a.out", "w", stdout);
    enter();
    build();
}
```

9 Mã hóa - Toán *(1 điểm)*

Khi nghiên cứu xây dựng thuật toán mã hóa, Nam cần giải quyết bài toán sau: Với bốn số nguyên dương L, R, A, K, cần đếm số lượng số nguyên dương mà và (A.S) % K = 0 , trong đó % là phép toán chia lấy dư.

Hãy giúp Nam giải bài toán trên.

Yêu cầu 1: Xây dựng công thức thể hiện mỗi liên hệ các chữ số L, R, A, K để đưa ra số lượng số nguyên S thỏa mãn

Yêu cầu 2: Đêm số lượng số S thảo mãn với L = 1232, R = 324343434665, A = 25, K = 525.

Lời giải

Khoảng cách giữa 2 số s gần kề nhau sẽ bằng ước chung lớn nhất của a, b.

Vậy ta chỉ cần tìm số đầu và số cuối rồi đếm số số hạng là được

Mình lười quá ae tự gõ cái code này vào máy để in ra kết quả cụ thể như đề nhé

```
l, r, a, k = map(int, input().split())

import math
x = math.lcm(a, k) / a

d, c = math.ceil(l / x) * x, (r // x) * x

print(int((c - d) // x + 1))
```