Hướng dẫn học lập trình C++ cơ bản đến nâng cao

Dinh Xuan Minh

Ngày 19 tháng 10 năm 2025

Mục lục

Lời nói đầu

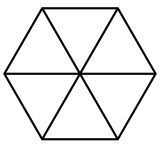
Cuốn sách này tổng hợp các đề thi và lời giải chi tiết các bài lập trình dành cho học sinh giỏi lớp 9. Mỗi bài gồm phần đề, phân tích, hướng dẫn giải và code mẫu.

1 ĐỀ HSG 9 THCS TỈNH HẢI DƯƠNG NĂM HỌC 2024-2025

Thời gian làm bài: 150 phút
Độ khó:

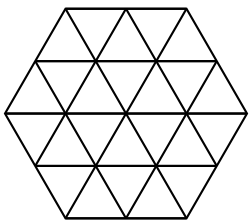
1.1 Bài 1: Số tam giác

Đề bài: Một lục giác đều với độ dài cạnh là số nguyên dương có thể được ghép bằng một số tam giác đều có độ dài cạnh bằng 1. Ví dụ dưới đây là hai hình lục giác đều được ghép bằng các tam giác đều cạnh 1.



Hình 1

Hình lục giác đều độ dài cạnh 1
được ghép bằng 6 tam giác đều độ dài cạnh 1



Hình lục giác đều độ dài cạnh 2 được ghép bằng
24 tam giác đều độ dài cạnh 1

Yêu cầu: Hỏi rằng số tam giác đều tối thiểu là bao nhiêu để ghép được n hình lục giác đều với độ dài các cạnh lần lượt là $1, 2, ..., n$.

Dữ liệu đầu vào:

Gồm $T + 1$ dòng:

- Dòng đầu tiên chứa số nguyên dương T ($1 \leq T \leq 10^5$) là số lượng test.
- Tiếp theo là T dòng, mỗi dòng tiếp theo chứa một số nguyên dương n ($1 \leq n \leq 10^6$) mô tả một bộ dữ liệu.

Dữ liệu đầu ra: Gồm T dòng, mỗi dòng in ra số tam giác đều tối thiểu cần thiết cho bộ dữ liệu tương ứng.

Ràng buộc dữ liệu:

- Subtask 1 (60 điểm): $T \leq 10$, $n \leq 1000$.
- Subtask 2 (40 điểm): $10 < T \leq 10^5$.

Ví dụ:

Input

```
3
1
2
3
```

Output

```
6
30
84
```

Hướng dẫn giải:

- **Kiến thức cần có:** Mảng cộng dồn, toán.
- **Subtask 1:** Một hình lục giác đều có độ dài cạnh n có thể được chia thành 6 tam giác đều lớn (mỗi tam giác có độ dài cạnh n). Mỗi tam giác đều lớn được ghép từ n^2 tam giác đều nhỏ cạnh 1. Do đó, tổng số tam giác nhỏ để ghép một hình lục giác đều cạnh n là: $6 \times n^2$.

  Vậy kết quả là $6 \times 1^2 + 6 \times 2^2 + ... + 6 \times n^2$.

  Với mỗi truy vấn, duyệt từ 1 đến n để tính tổng số tam giác đều của n hình lục giác đều.

  Độ phức tạp: $O(T \times n)$.

- **Subtask 2:**
  - **Cách 1:** Thay vì duyệt từ 1 đến n trong mỗi truy vấn, mình có thể tính trước các trường hợp bằng mảng cộng dồn.

    Độ phức tạp: $O(T + n)$.

– **Cách 2:** Kết quả bài toán là $6 \times 1^2 + 6 \times 2^2 + ... + 6 \times n^2 = (1^2 + 2^2 + ... + n^2) \times 6$.
Ta có công thức tính $(1^2 + 2^2 + ... + n^2) = \frac{n(n+1)(2n+1)}{6}$.
Suy ra $(1^2 + 2^2 + ... + n^2) \times 6 = n(n+1)(2n+1)$.
Vậy kết quả của mỗi truy vấn là $n(n+1)(2n+1)$.
Độ phức tạp: $O(T)$.

Solution C++:

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
#define FASTIO ios_base::sync_with_stdio(NULL); cin.tie(NULL);
   cout.tie(NULL);

const ll M = 1e9 + 7;
const ll N = 2e6;

ll t, n, f[N + 5];

void sub1() {
    while(t--) {
        ll res = 0;
        cin >> n;
        for(ll i = 1; i <= n; i++) {
            res += (i * i) * 6;
        }
        cout << res << "\n";
    }
}

void sub2() {
    for(ll i = 1; i <= N; i++) {
        f[i] = f[i - 1] + (i * i) * 6;
    }
    while(t--) {
        cin >> n;
        cout << f[n] << "\n";
    }
}

void sub2_2() {
    while(t--) {
        cin >> n;
        cout << n * (n + 1) * (2 * n + 1) << "\n";
    }
}

int main() {
    FASTIO;
    cin >> t;
```

```
43      if(t <= 10 && n <= 1000) {
44          sub1();
45      } else {
46          sub2();
47          // sub2_2();
48      }
49  }
```

1.2 Bài 2: Chia phần

Đề bài: Cho dãy số nguyên dương $a_1, a_2, ..., a_n$. Chia dãy này thành hai phần

- Phần thứ nhất gồm các số $a_1, a_2, ..., a_k$.
- Phần thứ hai gồm các số còn lại.

Yêu cầu: Gọi T_1 và T_2 lần lượt là tổng các số trong phần thứ nhất và phần thứ hai. Hãy tìm giá trị nhỏ nhất của $|T_1 - T_2|$.

Dữ liệu đầu vào: Gồm hai dòng:

- Dòng đầu tiên chứa số nguyên dương $n (2 < n \leq 10^6)$ là số lượng phần tử trong dãy.
- Dòng thứ hai chứa n số nguyên dương $a_1, a_2, ..., a_n$ $(|a_i| \leq 10^9, \forall i = 1, 2, ..., n)$ cách nhau bởi dấu cách.

Dữ liệu đầu ra: Gồm một dòng duy nhất chứa giá trị nhỏ nhất của $|T_1 - T_2|$.

Ràng buộc dữ liệu:

- Subtask 1 (75 điểm): $n \leq 5000$.
- Subtask 2 (25 điểm): $n > 5000$.

Ví dụ:

Input

```
5
1 2 3 4 5
```

Output

```
3
```

Hướng dẫn giải:

- **Kiến thức cần có:** Toán.
- **Subtask 1:** Với mỗi vị trí i trong dãy, mình duyệt từ i về 1 để tính T_1.

  Duyệt từ $i + 1$ đến n để tính T_2 và lấy giá trị nhỏ nhất của $|T_1 - T_2|$ là đáp án bài toán.

  Độ phức tạp: $O(n^2)$.

- **Subtask 2:**

  Với mỗi vị trí i từ 1 đến $n-1$:

  – Cộng a_i vào T_1.

  – Trừ a_i khỏi T_2.

  – Cập nhật min của $|T_1 - T_2|$.

  Với T_2 ban đầu là tổng tất cả phần tử của dãy a.

  Độ phức tạp: $O(n)$.

Solution C++:

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
#define FASTIO ios_base::sync_with_stdio(NULL); cin.tie(NULL);
   cout.tie(NULL);

const ll M = 1e9 + 7;
const ll N = 2e6;

ll n, a[N], f[N], res = 1e18, sum2 = 0, sum1 = 0;

void sub1() {
    for(ll i = 1; i <= n; i++) {
        ll res1 = 0, res2 = 0;
        for(ll j = i; j >= 1; j--) {
            res1 += a[j];
        }
        for(ll j = i + 1; j <= n; j++) {
            res2 += a[j];
        }
        res = min(res, abs(res1 - res2));
    }
    cout << res;
}

void sub2() {
    for(ll i = 1; i < n; i++) {
        sum1 += a[i];
        sum2 -= a[i];
        res = min(res, abs(sum1 - sum2));
    }
    cout << res;
}

int main() {
    FASTIO;
    cin >> n;
    for(ll i = 1; i <= n; i++) {
        cin >> a[i];
```

```
40          sum2 += a[i];
41      }
42      if(n <= 5000) sub1();
43      else sub2();
44  }
```

1.3 Bài 3: Kiểm tra số chẵn lẻ

Đề bài: An có n đoạn thẳng. Cậu ta nhận thấy rằng một số đoạn thẳng cùng chiều dài nên có thể xếp thành những hình vuông.

Yêu cầu:

Hỏi rằng số hình vuông nhiều nhất An có thể xếp được là bao nhiêu?

Dữ liệu đầu vào: Gồm hai dòng:

- Dòng đầu tiên chứa số nguyên dương n ($1 \leq n \leq 3 \times 10^5$) là số lượng đoạn thẳng.
- Dòng thứ hai chứa n số nguyên dương $a_1, a_2, ..., a_n$ ($1 \leq a_i \leq 10^{18}$) là độ dài của các đoạn thẳng.

Dữ liệu đầu ra: Gồm một số nguyên duy nhất là số lượng hình vuông nhiều nhất An có thể xếp được. **Ràng buộc dữ liệu:**

- Subtask 1 (30%): $n \leq 2000, a_i \leq 10^6$.
- Subtask 2 (30%): $n > 2000, a_i \leq 10^6$.
- Subtask 3 (40%): $n > 2000, a_i \leq 10^{18}$.

Ví dụ:

Input

```
9
2 2 2 9 2 2 2 2 2
```

Output

```
2
```

Hướng dẫn giải:

- **Kiến thức yêu cầu:** map.
- **Subtask 3:** Để tạo thành một hình vuông thì cần 4 đoạn thẳng có độ dài bằng nhau. Nên chỉ cần tính tổng của $\frac{f(a_i)}{4}$ (với $f(x)$ là số lần xuất hiện của x trong dãy a).

  Độ phức tạp: $O(n)$.

Solution C++:

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
#define FASTIO ios_base::sync_with_stdio(NULL); cin.tie(NULL);
   cout.tie(NULL);

const ll M = 1e9 + 7;
const ll N = 2e6;

ll n, a[N], mx, res;
map<ll, ll> f;

void sub3() {
    for(pair<ll, ll> i : f) {
        res += i.second / 4;
    }
    cout << res;
}

int main() {
    FASTIO;
    cin >> n;
    for(ll i = 1; i <= n; i++) {
        cin >> a[i];
        f[a[i]]++;
        mx = max(mx, a[i]);
    }
    if(n > 2000 && mx <= 1e18) sub3();
}
```

1.4 Bài 4: Tích lớn nhất

Đề bài: Cho một dãy số nguyên dương $a_1, a_2, ..., a_n$.

Yêu cầu: Hãy tính giá trị lớn nhất của biểu thức $a_i \times a_j \times a_k$ với $1 \leq i < j < k \leq n$.

Dữ liệu đầu vào: Gồm hai dòng:

- Dòng đầu tiên chứa số nguyên dương n ($3 \leq n \leq 3 \times 10^5$) là số lượng phần tử trong dãy.
- Dòng thứ hai chứa n số nguyên dương $a_1, a_2, ..., a_n$ ($|a_i| \leq 10^6$) cách nhau bởi dấu cách.

Dữ liệu đầu ra: Gồm một số nguyên duy nhất là giá trị lớn nhất của biểu thức $a_i \times a_j \times a_k$.

Ràng buộc dữ liệu:

- Subtask 1 (40%): $n \leq 100$.
- Subtask 2 (60%): $n > 100$.

Ví dụ:

Input

6
5 2 10 1 3 2

Output

150

Hướng dẫn giải:
Solution C++:

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

int main() {
    int n;
    cin >> n;
    long long max_product = 0;
    long long a[3];

    for (int i = 0; i < n; i++) {
        long long x;
        cin >> x;
        if (i < 3) {
            a[i] = x;
        } else {
            if (x > a[0]) {
                a[0] = x;
            }
            if (x > a[1]) {
                a[1] = a[0];
                a[0] = x;
            } else if (x > a[2]) {
                a[2] = x;
            }
        }
    }

    max_product = a[0] * a[1] * a[2];
    cout << max_product << endl;

    return 0;
}
```

1.5 Bài 5: Đếm số cặp

Đề bài: Cho một dãy số nguyên dương $a_1, a_2, ..., a_n$. **Yêu cầu:** Hãy đếm số cặp (i, j) với $1 \leq i < j \leq n$ thỏa mãn tính chất: Số $a_i \times a_j$ là một số chính phương (số nguyên dương

x được gọi là chính phương nếu tồn tại một số nguyên dương y sao cho $x = y^2$). **Dữ liệu đầu vào:** Gồm hai dòng:

- Dòng đầu tiên chứa số nguyên dương n ($n \leq 10^6$) là số lượng phần tử trong dãy.
- Dòng thứ hai chứa n số nguyên dương $a_1, a_2, ..., a_n$ ($1 \leq a_i \leq 10^6, \forall i = 1, 2, ..., n$) cách nhau bởi dấu cách.

Dữ liệu đầu ra: Gồm một số nguyên duy nhất là số lượng cặp (i, j) thỏa mãn tính chất đã nêu. **Ràng buộc dữ liệu:**

- Subtask 1 (40%): $n \leq 2000$.
- Subtask 2 (40%): $n > 2000, a_i \leq 10^4$.
- Subtask 3 (20%): $n > 2000, a_i \leq 10^6$.

Ví dụ:

Input

```
5
2 8 3 75 27
```

Output

```
4
```

Hướng dẫn giải:
Solution C++:

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    int n;
    cin >> n;
    vector<int> a(n);
    for (int i = 0; i < n; i++) {
        cin >> a[i];
    }

    unordered_map<int, int> freq;
    for (int i = 0; i < n; i++) {
        freq[a[i]]++;
    }

    long long count = 0;
    for (auto& pair1 : freq) {
        for (auto& pair2 : freq) {
            if (pair1.first <= pair2.first) {
                int product = pair1.first * pair2.first;
                int root = sqrt(product);
                if (root * root == product) {
                    if (pair1.first == pair2.first) {
```

```
                        count += (long long)pair1.second * (pair1.
                            second - 1) / 2;
                    } else {
                        count += (long long)pair1.second * pair2.
                            second;
                    }
                }
            }
        }

        cout << count << endl;

        return 0;
}
```

2 ĐỀ HSG 9 THCS TỈNH ĐIỆN BIÊN NĂM HỌC 2024-2025

Thời gian làm bài: 150 phút Độ khó:

2.1 Bài 1: Tính tiền

Đề bài: Trong đợt Hội chợ thương mai Điện Biên năm 2024. Để kích cầu một doanh nghiệp đã đưa ra chương trình khuyến mãi. Theo đó, nếu tổng giá trị hóa đơn lớn hơn hoặc bằng 2,000,000 đồng, khách hàng sẽ được giảm giá 15% trên tổng giá trị hóa đơn. Nếu không đạt điều kiện trên, khách hàng sẽ phải thanh toán toàn bộ giá trị hóa đơn mà không được giảm giá.

Yêu cầu:

Viết chương trình để tính số tiền thực tế khách hàng phải thanh toán dựa trên: Số lượng hàng bán (ký hiệu là a), đơn giá của mỗi mặt hàng (ký hiệu là b đồng).

Dữ liệu đầu vào:

Gồm một dòng duy nhất chứa hai số nguyên dương a và b ($1 \leq a \leq 10^4, 2{\times}10 \leq b \leq 10^9$) cách nhau bởi dấu cách.

Dữ liệu đầu ra:

Gồm một dòng duy nhất số tiền thực tế khách hàng cần thanh toán.Kết quả đảm bảo là số nguyên.

Ràng buộc dữ liệu:

- Subtask 1 (80%): $1 \leq a \leq 100$, $2 \times 10 \leq b \leq 10^6$.
- Subtask 2 (20%): $10^2 < a \leq 10^4$, $10^6 < b \leq 10^9$.

Ví dụ:

Input
2 1000000

Output

1700000

Hướng dẫn giải: Tổng giá trị hoá đơn là $a \times b$ (với a là số lượng hàng bán, b là đơn giá mỗi mặt hàng).

Nếu tổng giá trị hoá đơn lớn hơn 2000000 thì kết quả bài toán là $(a \times b) - (a \times b \times 15\%)$.

Còn lại là $a \times b$.

Độ phức tạp: $O(1)$.

Solution C++:

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
#define FASTIO ios_base::sync_with_stdio(NULL); cin.tie(NULL);
   cout.tie(NULL);

const ll M = 1e9 + 7;
const ll N = 1e7;

ll a, b;

int main() {
    FASTIO;
    cin >> a >> b;
    if(a * b >= 2000000) cout << (a * b) - (a * b * 15 / 100);
    else cout << a * b;
}
```

2.2 Bài 2: Số cùng nhau

Đề bài: Trong một ngôi làng cổ, các số tự nhiên thường kết đôi để cùng nhau thực hiện những nhiệm vụ đặc biệt. Nhưng không phải cặp số nào cũng có thể đồng hành, chỉ những cặp "tương thích hoàn hảo"mới được chọn. Một cặp số i và j được xem là tương thích hoàn hảo nếu chúng không có bất kỳ ước chung nào khác ngoài số 1, nghĩa là $\text{UCLN}(i, j) = 1$.

Yêu cầu:

Nhiệm vụ của bạn là giúp trưởng làng đếm xem có bao nhiêu cặp số cùng nhau trong đoạn $[a, b]$ được chọn làm bạn đồng hành lý tưởng.

Dữ liệu đầu vào:

Một dòng duy nhất chứa số nguyên dương a và b, cách nhau bởi dấu cách ($1 \leq a < b \leq 10^3$).

Dữ liệu đầu ra:

In ra một số nguyên là số lượng cặp số (i, j) thỏa mãn điều kiện trên.

Ràng buộc dữ liệu:

- Subtask 1 (40%): $1 \leq a, b \leq 10$.
- Subtask 2 (30%): $10 < a, b \leq 100$.

- Subtask 3 (30%): $100 < a, b \leq 1000$.

Ví dụ:

Input

1 5

Output

9

Giải thích: Trong đoạn $[1, 5]$, các cặp số cùng nhau là: $(1, 2)$, $(1, 3)$, $(1, 4)$, $(1, 5)$, $(2, 3)$, $(2, 5)$, $(3, 4)$, $(3, 5)$ và $(4, 5)$. Tổng cộng có 9 cặp.

Hướng dẫn giải: Chỉ cần duyệt lần lượt tất cả các cặp số trong dãy a và kiểm tra nếu các cặp đó có ước chung lớn nhất là 1

Để tính ước chung lớn nhất, có thể dùng thuật toán Euclid.

Độ phức tạp: $O((b-a)^2 \log min(a, b))$.

Solution C++:

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
#define FASTIO ios_base::sync_with_stdio(NULL); cin.tie(NULL);
   cout.tie(NULL);

const ll M = 1e9 + 7;
const ll N = 1e7;

ll a, b, res;

ll gcd(ll a, ll b) {
    if(b == 0) return a;
    return gcd(b, a % b);
}

int main() {
    FASTIO;
    cin >> a >> b;
    for(ll i = a; i < b; i++) {
        for(ll j = i + 1; j <= b; j++) {
            if(gcd(i, j) == 1) {
                res++;
            }
        }
    }
    cout << res;
}
```

2.3 Bài 3: Thu nhập

Đề bài: Để biết được mức thu nhập trung bình hàng tháng của các hộ dân trong thành phố. Thành phố đã tiến hành khảo sát N hộ dân, hộ dân thứ i có thu nhập a_i triệu đồng. Nhìn vào số liệu thống kê nhận thấy rằng mỗi hộ dân có thu nhập khác nhau, lãnh đạo thành phố muốn biết mức thu nhập thấp nhất và cao nhất cũng như mức thu nhập nào phổ biến trong các hộ dân nhất để thành phố có kế hoạch phát triển kinh tế giàu mạnh.

Yêu cầu: Hãy cho biết thu nhập thấp nhất, cao nhất và đếm các hộ dân có thu nhập phổ biến nhất trong thành phố.

Dữ liệu đầu vào: Gồm hai dòng:

- Dòng đầu tiên chứa số nguyên dương N ($1 \leq N \leq 10^5$) là số lượng hộ dân.
- Dòng thứ hai chứa N số nguyên dương a_i ($1 \leq a_i \leq 10^9, i = 1...N$) là thu nhập của các hộ dân, cách nhau bởi dấu cách.

Dữ liệu đầu ra: Gồm hai dòng:

- Dòng đầu tiên in ra thu nhập thấp nhất và cao nhất, cách nhau bởi dấu cách.
- Dòng thứ hai in ra số nguyên dương là số hộ dân nhiều nhất có mức thu nhập bằng nhau.

Ràng buộc dữ liệu:

- Subtask 1 (40%): $1 \leq N \leq 10^2$.
- Subtask 2 (30%): $10^2 < N \leq 10^3$.
- Subtask 3 (30%): $10^3 < N \leq 10^5$.

Ví dụ:

Input

```
9
5 1 5 8 6 2 3 6 3
```

Output

```
1 8
2
```

Hướng dẫn giải:

- **Kiến thức cần có:** map.
- Lấy giá trị nhỏ nhất và giá trị cao nhất trong dãy a.

  Đếm số lần xuất hiện của mỗi phần tử trong dãy và duyệt qua dãy để lấy số lần xuất hiện lớn nhất.

  Độ phức tạp: $O(n)$.

Solution C++:

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
#define FASTIO ios_base::sync_with_stdio(NULL); cin.tie(NULL);
   cout.tie(NULL);

const ll M = 1e9 + 7;
const ll N = 1e7;

ll n, a[N], mx = -1e18, mi = 1e18, res;
map<ll, ll> f;

int main() {
    FASTIO;
    cin >> n;
    for(ll i = 1; i <= n; i++) {
        cin >> a[i];
        mx = max(mx, a[i]);
        mi = min(mi, a[i]);
        f[a[i]]++;
    }
    for(pair<ll, ll> i : f) {
        res = max(res, i.second);
    }
    cout << mi << "␣" << mx << "\n";
    cout << res;
}
```

2.4 Bài 4: Đếm dãy con liên tiếp

Đề bài:

Cho dãy số A có n số nguyên $a_1, a_2, \ldots, a_n$. Một dãy con liên tiếp các số hạng của dãy A là dãy các số hạng từ số hạng a_i đến số hạng a_j $(1 \leq i \leq j \leq n)$.

Yêu cầu:

Hãy cho biết dãy A có bao nhiêu dãy con liên tiếp mà giá trị tuyệt đối của tổng các số hạng trong dãy con đó lớn hơn một số nguyên dương S cho trước.

Dữ liệu đầu vào:

Gồm hai dòng:

- Dòng thứ nhất chứa hai số nguyên dương n và S $(n \leq 10^5, S \leq 10^{14})$;
- Dòng thứ hai chứa n số nguyên $a_1, a_2, \ldots, a_n$ $(|a_i| \leq 10^9)$. Hai số liên tiếp trên cùng dòng được ghi cách nhau bởi dấu cách.

Dữ liệu đầu ra:

Gồm một số nguyên duy nhất là số dãy con liên tiếp thỏa mãn yêu cầu của bài toán.

Ràng buộc dữ liệu:

- Subtask 1 (50%): $n \leq 100$.

- Subtask 2 (30%): $n \leq 10^3$.
- Subtask 3 (20%): $n \leq 10^5$.

Ví dụ 1:

Input

```
4 4
5 -1 8 -5
```

Output

```
6
```

Giải thích: Các dãy con liên tiếp có tổng tuyệt đối lớn hơn 4 là:

$$\{5\}, \{8\}, \{-1, 8\}, \{5, -1, 8\}, \{5, -1, 8, -5\}$$

.

Ví dụ 2:

Input

```
10 7
-4 9 2 -11 -3 8 -6 5 -3 1
```

Output

```
12
```

Hướng dẫn giải:

Sử dụng prefix sum và hai con trỏ để đếm số lượng dãy con liên tiếp có tổng tuyệt đối lớn hơn S.

Solution C++:

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
int main() {
    int n;
    ll S;
    cin >> n >> S;
    vector<ll> a(n+1, 0), prefix(n+1, 0);
    for (int i = 1; i <= n; ++i) {
        cin >> a[i];
        prefix[i] = prefix[i-1] + a[i];
    }
    ll res = 0;
    for (int l = 1; l <= n; ++l) {
        for (int r = l; r <= n; ++r) {
            ll sum = prefix[r] - prefix[l-1];
            if (abs(sum) > S) res++;
```

```
        }
    }
    cout << res << endl;
    return 0;
}
```

3 ĐỀ HSG 9 THCS TỈNH NGHỆ AN NĂM HỌC 2024-2025

Thời gian làm bài: 150 phút Độ khó:

3.1 Bài 1: Số chính phương

Đề bài:

Trong buổi ôn tập hôm nay, thầy giáo đã chuẩn bị một số món quà để trao tặng cho các bạn trong đội tuyển học sinh giỏi trả lời đúng bài toán về số học của thầy như sau:

Cho hai số nguyên dương $L, R (1 \leq L \leq R \leq 10^{18})$. Hãy đếm số lượng số chính phương trong đoạn $[L, R]$.

Biết rằng số chính phương là số bằng bình phương của một số nguyên dương. Ví dụ: $1, 4, 9, 16, 25, ...$ là các số chính phương vì chúng là bình phương của các số nguyên dương $1, 2, 3, 4, 5, ...$

Rất nhanh chóng An đã đưa ra kết quả của bài toán. Em hãy lập trình giải quyết bài toán trên để biết xem An có được nhận phần thưởng của thầy giáo hay không.

Yêu cầu:

Hãy viết chương trình đếm số lượng số chính phương trong đoạn $[L, R]$.

Dữ liệu đầu vào:

Gồm một dòng duy nhất chứa hai số nguyên dương L, R. $(1 \leq L \leq R \leq 10^{18})$.

Dữ liệu đầu ra:

Gồm một số nguyên duy nhất là số lượng số chính phương trong đoạn $[L, R]$.

Ràng buộc dữ liệu:

- Subtask 1 (50%): $1 \leq L, R \leq 10^6$.
- Subtask 2 (50%): $10^6 < L, R \leq 10^{18}$.

Ví dụ:

Input
4 30

Output
4

Giải thích: Trong đoạn $[4, 30]$ có các số chính phương là: $4, 9, 16, 25$. Tổng cộng có 4 số chính phương.

Hướng dẫn giải: Sử dụng hàm căn bậc hai để tìm số lượng số chính phương trong đoạn $[L, R]$.

Độ phức tạp: $O(1)$.
Solution C++:

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
#define FASTIO ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cout.tie(nullptr);

int main() {
    FASTIO;
    ll l, r;
    cin >> l >> r;
    long double sl = sqrt((long double)l), sr = sqrt((long double)
       r);
    ll l2 = ceil(sl - 1e-12), r2 = floor(sr + 1e-12);

    cout << max(0LL, r2 - l2 + 1) << '\n';
}
```

3.2 Bài 2: Cây thông

Đề bài: Chào đón Giáng sinh an lành, một cửa hàng có chương trình quà tặng đặc biệt. Lối vào của cửa hàng được trang trí bởi hai cây thông, cây thứ nhất treo n tấm thẻ ghi các giá trị lần lượt là $A_1, A_2, \ldots, A_n$ và cây thứ hai cũng có n tấm thẻ ghi các giá trị lần lượt là $B_1, B_2, \ldots, B_n$. Người khách nào chọn được cặp thẻ A_i và $B_j (1 \leq i, j \leq n)$ sao cho $|A_i + B_j|$ là giá trị nhỏ nhất thì sẽ được tặng một cây thông mình thích nhất trong cửa hàng.

Yêu cầu:

Em hãy giúp cửa hàng xác định giá trị nhỏ nhất của $|A_i + B_j|$ để tặng quà cho người khách lựa chọn được cặp thẻ may mắn.

Dữ liệu đầu vào: Gồm ba dòng:

- Dòng đầu tiên chứa số nguyên dương n $(1 \leq n \leq 10^6)$ là số lượng tấm thẻ trên mỗi cây thông.
- Dòng thứ hai chứa n số nguyên $A_1, A_2, \ldots, A_n$ $(-10^9 \leq A_i \leq 10^9)$ là giá trị ghi trên các tấm thẻ của cây thông thứ nhất.
- Dòng thứ ba chứa n số nguyên $B_1, B_2, \ldots, B_n$ $(-10^9 \leq B_i \leq 10^9)$ là giá trị ghi trên các tấm thẻ của cây thông thứ hai.

Dữ liệu đầu ra: Gồm một số nguyên duy nhất là giá trị nhỏ nhất của $|A_i + B_j|$.
Ràng buộc dữ liệu:

- Subtask 1 (60%): $1 \leq n \leq 10^3$.
- Subtask 2 (40%): $10^3 < n \leq 10^6$.

Ví dụ:

Input

```
5
-9 3 -17 -5 3
-1 7 2 3 20
```

Output

```
2
```

Giải thích: Trong ví dụ trên, ta có các cặp thẻ A_i và B_j như sau:

- $A_1 + B_1 = -9 - 1 = -10$.
- $A_1 + B_2 = -9 + 7 = -2$.
- $A_1 + B_3 = -9 + 2 = -7$.
- $A_1 + B_4 = -9 + 3 = -6$.
- $A_1 + B_5 = -9 + 20 = 11$.
- $A_2 + B_1 = 3 - 1 = 2$.
- $A_2 + B_2 = 3 + 7 = 10$.
- $A_2 + B_3 = 3 + 2 = 5$.
- $A_2 + B_4 = 3 + 3 = 6$.
- $A_2 + B_5 = 3 + 20 = 23$.
- $A_3 + B_1 = -17 - 1 = -18$.
- $A_3 + B_2 = -17 + 7 = -10$.
- $A_3 + B_3 = -17 + 2 = -15$.
- $A_3 + B_4 = -17 + 3 = -14$.
- $A_3 + B_5 = -17 + 20 = 3$.
- $A_4 + B_1 = -5 - 1 = -6$.
- $A_4 + B_2 = -5 + 7 = 2$.
- $A_4 + B_3 = -5 + 2 = -3$.
- $A_4 + B_4 = -5 + 3 = -2$.
- $A_4 + B_5 = -5 + 20 = 15$.
- $A_5 + B_1 = 3 - 1 = 2$.
- $A_5 + B_2 = 3 + 7 = 10$.

- $A_5 + B_3 = 3 + 2 = 5$.
- $A_5 + B_4 = 3 + 3 = 6$.
- $A_5 + B_5 = 3 + 20 = 23$.

Từ các cặp trên, ta thấy rằng giá trị nhỏ nhất của $|A_i + B_j|$ là 2, với các cặp (A_2, B_1) và (A_4, B_2).

Hướng dẫn giải:

- **Kiến thức cần có:** Hai con trỏ.
- **Subtask 1:** Duyệt qua lần lượt các cặp số (A_i, B_j) và lấy giá trị nhỏ nhất của $|A_i + B_j|$.

  Độ phức tạp: $O(n^2)$.
- **Subtask 2:** Sắp xếp hai mảng A và B, sau đó sử dụng hai con trỏ để tìm cặp thẻ có tổng gần bằng 0 nhất.

  Nếu tổng nhỏ hơn 0 thì tăng vị trí đang xét trong A lên.

  Ngược lại giảm vị trí đang xét trong B xuống.

  Và lấy $|A_i + B_j|$ nhỏ nhất khi đang xét.

  Độ phức tạp: $O(n \log n)$.

Solution C++:

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
#define FASTIO ios_base::sync_with_stdio(NULL); cin.tie(NULL);
   cout.tie(NULL);

const ll M = 1e9 + 7;
const ll N = 1e7;

ll n, a[N], b[N], res = 1e18;

void sub1() {
    for(ll i = 1; i <= n; i++) {
        for(ll j = 1; j <= n; j++) {
            res = min(res, abs(a[i] + b[j]));
        }
    }
    cout << res;
}

void sub2() {
    sort(a + 1, a + n + 1);
    sort(b + 1, b + n + 1);
    ll i = 1, j = n;
    while(i <= n && j >= 1) {
```

```
        ll sum = a[i] + b[j];
        res = min(res, abs(sum));
        if(sum < 0) {
            i++;
        } else if(sum > 0) {
            j--;
        } else {
            break;
        }
    }
    cout << res;
}

int main() {
    FASTIO;
    cin >> n;
    for(ll i = 1; i <= n; i++) {
        cin >> a[i];
    }
    for(ll i = 1; i <= n; i++) {
        cin >> b[i];
    }
    if(n <= 1000) sub1();
    else sub2();
}
```

3.3 Bài 3: Trồng cây

Đề bài: An là chủ nhiệm của CLB Sống Xanh nơi mình sinh sống. Nhân dịp lễ Giáng sinh và chuẩn bị đón tết Nguyên Đán, CLB phát động chiến dịch "Xanh quê hương"với nhiều hoạt động có ý nghĩa nhằm tạo môi trường Xanh - Sạch - Đẹp. Hoạt động đầu tiên trong chiến dịch là thực hiện trồng một hàng cây chạy dọc theo một tuyến đường.

Trên tuyến đường đã được đánh dấu n vị trí cách đều nhau để trồng cây, trong đó có một số vị trí đã được trồng cây từ trước. CLB gồm An và k thành viên sẽ trồng $k+1$ cây vào $k+1$ vị trí trống (mỗi thành viên sẽ trồng một cây). Để thuận tiện quản lí, An muốn tìm một vị trí trồng cây của mình và vị trí của k thành viên, sao cho vị trí thành viên xa nhất đến vị trí của An là ngắn nhất.

Yêu cầu: Hãy lập trình giúp An xác định giá trị nhỏ nhất của khoảng cách từ vị trí từ vị trí thành viên xa nhất đến vị trí của An.

Dữ liệu đầu vào: Gồm hai dòng:

- Dòng đầu tiên chứa hai số nguyên dương n và k ($1 \leq k \leq n \leq 10^5$) là số vị trí trồng cây và số thành viên trong CLB.
- Dòng thứ hai chứa một xâu nhị phân s gồm n phần tử biểu diễn trạng thái của n vị trí. Giá trị 0 biểu diễn vị trí trống, giá trị 1 biểu diễn vị trí đã trồng cây. (Dữ liệu đảm bảo số phần tử có giá trị 0 trong xâu s luôn lớn hơn k).

Dữ liệu đầu ra: Gồm một số nguyên duy nhất là giá trị nhỏ nhất của khoảng cách từ vị trí thành viên xa nhất đến vị trí của An.

Ràng buộc dữ liệu:

- Subtask 1 (60%): $1 \leq n \leq 10^3$.
- Subtask 2 (40%): $10^3 < n \leq 10^5$.

Ví dụ:

Input

7 2
1010100

Output

2

Giải thích:

- Cách 1: Chọn các vị trí $2, 4, 6$. An ở vị trí số 4 và khoảng cách đến thành viên xa nhất là $|6 - 4| = |2 - 4| = 2$.
- Cách 2: Chọn các vị trí $4, 6, 7$. An ở vị trí số 6 và khoảng cách đến thành viên xa nhất là $|4 - 6| = 2$.

Hướng dẫn giải: Solution C++:

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    int n, k;
    cin >> n >> k;
    string s;
    cin >> s;

    vector<int> empty_positions;
    for (int i = 0; i < n; i++) {
        if (s[i] == '0') {
            empty_positions.push_back(i);
        }
    }

    int left = 0, right = empty_positions.size() - 1;
    int min_distance = INT_MAX;

    while (right - left + 1 > k) {
        min_distance = min(min_distance, empty_positions[right] -
            empty_positions[left]);
        left++;
    }

    cout << min_distance << endl;

    return 0;
}
```

3.4 Bài 4: Giáng sinh

Đề bài: Giáng sinh là khoảng thời gian đẹp nhất trong năm. Hai anh em William và Jacica là hai diễn viên múa chính của đoàn nghệ thuật đỉnh cao số một thế giới. Các vở diễn của họ góp phần hồi sinh các giá trị đạo đức, văn hóa truyền thông, đem lại cho người xem năng lượng tích cực, cảm nhận sâu sắc về các giá trị tốt đẹp và sự bình yên trong tâm hồn. Wiliam và Jacica vừa trở về nhà sau chuyến lưu diễn vòng quanh thế giới và bắt đầu trang trí cây thông Noel của họ bằng những món đồ xinh xắn đã mua trong quá trình lưu diễn.

Họ đã mua n món đồ trang trí cây thông được xếp cạnh nhau trong một hộp dài, món đồ trang trí thứ i có màu A_i. Hộp được mở ở cả hai đầu, vì vậy các món đồ có thể được lấy ra từ bên trái hoặc bên phải của hộp. Hộp này trong suốt, nên Wiliam và Jacica có thể nhìn thấy màu sắc của từng món đồ trang trí.

Jacica nghĩ ra một trò chơi để việc trang trí cây thông trở nên thú vị hơn. Trò chơi diễn ra như sau: Wiliam và Jacica thay phiên nhau chơi, Wiliam là người bắt đầu. Người chơi trong lượt của mình sẽ lấy một món đồ trang trí từ hộp (có thể từ bên trái hoặc bên phải) và đặt nó lên cây thông. Nếu món đồ được lấy có màu chưa từng được người nào lấy trước đó, người chơi sẽ ghi được một điểm. Trò chơi kết thúc khi món đồ trang tri cuối cùng được lấy ra khỏi hộp. Người chiến thắng là người ghi được nhiều điểm hơn. Vì cả Wiliam và Jacica đều là những người chơi xuất sắc, họ sẽ chơi một cách tối ưu.

Yêu cầu: Em hãy đưa ra kết quả cuối cùng của trò chơi.

Dữ liệu đầu vào: Gồm hai dòng:

- Dòng đầu tiên chứa số nguyên dương n ($1 \leq n \leq 3000$) là số lượng món đồ trang trí trong hộp.
- Dòng thứ hai chứa n số nguyên dương $A_1, A_2, ..., A_n$ với A_i ($1 \leq A_i \leq n$) là màu sắc của các món đồ trang trí, cách nhau bởi dấu cách.

Dữ liệu đầu ra: Gồm một dòng duy nhất gồm hai số, được nối bằng một ký tự ":"(Dấu hai chấm), lần lượt là điểm số của Wiliam và Jacica.

Ràng buộc dữ liệu:

- Subtask 1 (25%): $1 \leq A_i \leq 2$ với mọi $\forall i = 1, 2, ..., n$.
- Subtask 2 (20%): $1 \leq n \leq 20$.
- Subtask 3 (10%): $1 \leq A_i \leq 20$ với mọi $\forall i = 1, 2, ..., n$.
- Subtask 4 (20%): $1 \leq n \leq 300$.
- Subtask 5 (25%): $300 < n \leq 3000$.

Ví dụ 1:

Input

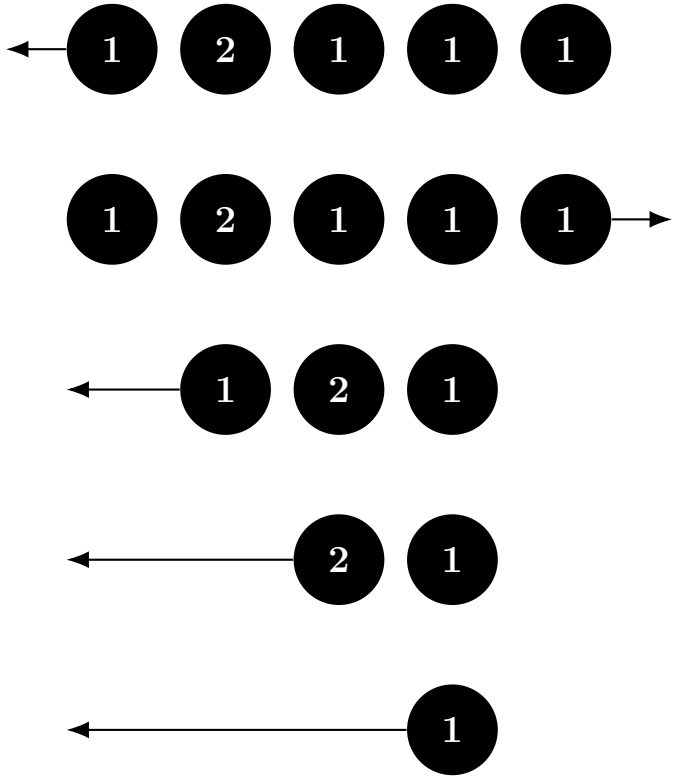
```
5
1 1 2 1 1
```

Output

```
1:1
```

Giải thích: Trong ví dụ này, Wiliam và Jacica sẽ chơi như sau:

- Đầu tiên Wiliam chọn món đồ trang trí có màu 1 từ bên trái, ghi được 1 điểm.
- Tiếp theo Jacica chọn món đồ trang trí có màu 1 từ bên phải và không có điểm vì màu này đã được Wiliam lấy.
- Tiếp theo Wiliam chọn màu 1 ở bên trái và không có điểm vì màu 1 đã được lấy.
- Tiếp theo Jacica chọn màu 2 ở vế bên trái và được 1 điểm vì màu 2 chưa được lấy.
- Cuối cùng Wiliam chọn màu 1 và không được điểm vì màu 1 đã được lấy.



Hướng dẫn giải: Solution C++:

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    int n;
    cin >> n;
    vector<int> A(n);
    for (int i = 0; i < n; i++) {
        cin >> A[i];
    }

    vector<vector<int>> dp(n, vector<int>(n, 0));

    for (int len = 1; len <= n; len++) {
        for (int l = 0; l + len - 1 < n; l++) {
            int r = l + len - 1;
            if (l == r) {
```

```
                dp[l][r] = 1; // only one item
            } else {
                int left = A[l], right = A[r];
                if (left != right) {
                    dp[l][r] = max(dp[l + 1][r] + (A[l] != A[l +
                        1]), dp[l][r - 1] + (A[r] != A[r - 1]));
                } else {
                    dp[l][r] = max(dp[l + 1][r], dp[l][r - 1]);
                }
            }
        }
    }

    int william_score = (n + dp[0][n - 1]) / 2;
    int jacica_score = n - william_score;

    cout << william_score << ":" << jacica_score << endl;

    return 0;
}
```

4 ĐỀ HSG 9 THCS TỈNH THANH HÓA NĂM HỌC 2024-2025

Thời gian làm bài: 150 phút Độ khó:

4.1 Bài 1: Diện tích

Đề bài: Biết Nam đang ôn tập để tham gia kỳ thi học sinh giỏi cấp tỉnh môn Tin học. Bố đã đố bạn ấy lập trình để giải bài toán có nội dung như sau:

Cho 4 thanh sắt có độ dài lần lượt là a, b, c, d. Bố muốn Nam tạo một khung hình chữ nhật từ 4 thanh sắt đó (phải sử dụng cả 4 thanh), những đoạn dư của các thanh sắt (nếu có) sẽ được cắt bỏ. Hãy tìm diện tích lớn nhất của khung sắt được tạo thành.

Yêu cầu: Em hãy giúp bạn Nam giải bài toán trên.

Dữ liệu đầu vào: Gồm một dòng duy nhất chứa 4 số nguyên dương a, b, c, d ($0 < a, b, c, d \leq 10^5$).

Dữ liệu đầu ra: Gồm một số nguyên duy nhất là diện tích lớn nhất của khung hình chữ nhật được tạo thành từ 4 thanh sắt.

Ví dụ 1:

Input

```
7 3 4 6
```

Output

```
18
```

Ví dụ 2:

Input

5 5 5 5

Output

25

Hướng dẫn giải:

Để tạo thành một khung hình chữ nhật thì mình cần hai cặp thanh sắt, với hai thanh sắt trong một cặp có độ dài bằng nhau.

Vì bài toán yêu cầu tìm khung hình chữ nhật được tạo thành từ 4 thanh sắt mà có diện tích lớn nhất. Nên mình chỉ cần tìm hai cặp thanh sắt có với hai thanh sắt trong mỗi cặp bằng nhau và có độ dài lớn nhất.

Mình có thể sắp xếp lại 4 số, cặp số có độ dài lớn nhất có thể tạo thành bằng cách cắt thanh sắt lớn nhất thành thanh sắt khác mà lớn hơn 2 thanh sắt còn lại mà nhỏ hơn thanh sắt lớn nhất.

Cặp số có độ dài nhỏ hơn còn lại thì được tạo thành bằng cách cắt thanh sắt nhỏ thứ 2 thành thanh sắt nhỏ nhất.

Ví dụ: 1 2 3 4

Mình có thể cắt thanh sắt lớn nhất là 4 thành thanh sắt lớn hơn hai thanh còn lại nhưng nhỏ hơn nó là thanh 3, mình sẽ được cặp thanh sắt đầu tiên là 3 3.

Với cặp tiếp theo, mình có thể cắt thanh sắt nhỏ thứ 2 trong dãy là thanh có độ dài 2 thành thanh sắt nhỏ thứ nhất là thanh có độ dài 1, mình sẽ được cặp thanh sắt thứ hai là 1 1.

Đáp án bài toán là diện tích hình chữ nhật: 1×3.

Solution C++:

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
#define FASTIO ios_base::sync_with_stdio(NULL); cin.tie(NULL);
   cout.tie(NULL);

const ll M = 1e9 + 7;
const ll N = 1e7;

ll a[6];

int main() {
    FASTIO;
    cin >> a[1] >> a[2] >> a[3] >> a[4];
    sort(a + 1, a + 5);
    cout << a[1] * a[3];
}
```

4.2 Bài 2: Số lớn

Đề bài: Việt và Nam cùng nhau ôn luyện chủ đề xâu kí tự. Để buổi học trở nên thú vị hơn, mỗi bạn sẽ lần lượt đưa ra một bài toán dành cho bạn của mình. Bài toán của Việt dành cho Nam như sau: Cho một xâu ST bao gồm các kí tự chữ cái tiếng Anh (thường và hoa) và các kí tự số. Hãy thực hiện xóa đi các kí tự trong xâu ST để được một xâu mới $ST1$ chỉ còn K kí tự đều là kí tự số và khi giữ nguyên trật tự ban đầu thì xâu $ST1$ là xâu số lớn nhất có thể.

Yêu cầu: Em hãy giúp Nam tìm ra xâu $ST1$ theo đúng yêu cầu của bài toán.

Dữ liệu đầu vào: Gồm hai dòng:

- Dòng đầu tiên chứa một xâu ST ($1 \leq |ST| \leq 10^5$) là xâu kí tự cần xử lý.
- Dòng thứ hai chứa một số nguyên dương K ($1 \leq K \leq |ST|$). Biết rằng trong xâu ST luông đảm bảo có ít nhất K kí tự là số.

Dữ liệu đầu ra: Gồm một dòng duy nhất là xâu $ST1$ theo đúng yêu cầu của bài toán.

Ràng buộc dữ liệu:

- Subtask 1 (40%): $|ST| \leq 18$, xâu ST chỉ gồm các kí tự số.
- Subtask 2 (60%): Không có ràng buộc gì thêm.

Ví dụ 1:

Input

```
AmN69pQ3e6
2
```

Output

```
96
```

Ví dụ 2:

Input

```
Fish36colo99
3
```

Output

```
699
```

Hướng dẫn giải: Để giải bài toán này, ta cần thực hiện các bước sau:

- Đầu tiên, ta sẽ duyệt qua xâu ST và lưu trữ tất cả các kí tự số vào một danh sách.
- Sau đó, ta sẽ sắp xếp danh sách các kí tự số theo thứ tự giảm dần.
- Cuối cùng, ta sẽ lấy K kí tự đầu tiên trong danh sách đã sắp xếp để tạo thành xâu $ST1$.

Solution C++:

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    string ST;
    int K;
    cin >> ST >> K;

    vector<char> digits;
    for (char c : ST) {
        if (isdigit(c)) {
            digits.push_back(c);
        }
    }

    sort(digits.rbegin(), digits.rend());

    string ST1 = "";
    for (int i = 0; i < K; i++) {
        ST1 += digits[i];
    }

    cout << ST1 << endl;
    return 0;
}
```

4.3 Bài 3: Hộp số

Đề bài: Có n chiếc hộp được đánh số theo thứ tự từ 1 đến n và xếp chúng theo một hàng ngang. Mỗi chiếc hộp có một trong hai giá trị 0 hoặc 1. Thực hiện một lần thay đổi giá trị của tất cả các hộp từ vị trí $i, j (1 \leq i \leq j \leq n)$ theo quy tắc "Những hộp có giá trị bằng 1 sẽ được thay đổi bằng 0 và ngược lại", để sau khi theo đổi thì trong n chiếc hộp nhận được số chiếc hộp có giá trị bằng 1 là nhiều nhất.

Yêu cầu: Đếm ố hộp cóa giá trị bằng 1 nhiều nhất sau khi thay đổi như trên.

Dữ liệu đầu vào: Gồm hai dòng:

- Dòng đầu tiên chứa một số nguyên dương n $(0 < n \leq 10^6)$ là số lượng hộp.
- Dòng thứ hai chứa n số nguyên $a_1, a_2, \ldots, a_n$ $(a_i \in \{0, 1\})$ là giá trị của các hộp.

Dữ liệu đầu ra: Gồm một số nguyên duy nhất là số lượng hộp có giá trị bằng 1 nhiều nhất sau khi thay đổi.

Ràng buộc dữ liệu:

- Subtask 1 (30%): $1 \leq n \leq 500$.
- Subtask 2 (30%): $500 < n \leq 8000$.
- Subtask 3 (40%): $8000 < n \leq 10^6$.

Ví dụ 1:

Input

```
8
1 0 0 1 1 0 0 0
```

Output

```
6
```

Hướng dẫn giải: Để giải bài toán này, ta có thể sử dụng thuật toán quét hai đầu (two-pointer) hoặc thuật toán tiền tố (prefix sum) để tìm đoạn con có số lượng hộp có giá trị bằng 1 nhiều nhất sau khi thay đổi. **Solution C++:**

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    int n;
    cin >> n;
    vector<int> a(n);
    for (int i = 0; i < n; i++) {
        cin >> a[i];
    }

    int max_ones = 0;
    for (int i = 0; i < n; i++) {
        for (int j = i; j < n; j++) {
            int count = 0;
            for (int k = i; k <= j; k++) {
                count += (a[k] == 1) ? 0 : 1;
            }
            max_ones = max(max_ones, count);
        }
    }

    cout << max_ones << endl;
    return 0;
}
```

4.4 Bài 4: Bội số chung nhỏ nhất

Đề bài: Bội số chung nhỏ nhất của hai số nguyên x và y là số nguyên dương nhỏ nhất chia hết cho cả x và y, ký hiệu là $LCM(x, y)$.

Cho hai số nguyên dương a và b ($a \leq b$).

Yêu cầu: Hãy đếm số cặp số nguyên dương x, y sao cho $LCM(x, y)$ bằng tích các số nguyên liên tiếp từ a đến b (trường hợp a bằng b thì tích này bằng a).

Dữ liệu đầu vào: Gồm hai dòng:

- Dòng đầu ghi số nguyên dương T ($T \leq 10$) là số lượng bộ test.

- T dòng tiếp theo, mỗi dòng chứa hai số nguyên dương a, b ($1 \leq a \leq b \leq 10^6$) là đầu và cuối của đoạn cần xét.

Dữ liệu đầu ra: Gồm T dòng, mỗi dòng chứa một số nguyên duy nhất là số lượng cặp số tìm được khi chia hết cho $10^9 + 7$.

Ràng buộc dữ liệu:

- Subtask 1 (40%): $1 \leq a, b \leq 10$.
- Subtask 2 (30%): $1 \leq a, b \leq 100$.
- Subtask 3 (30%): Không có ràng buộc gì thêm.

Ví dụ 1:

Input

```
2
3 4
7 7
```

Output

```
15
3
```

Giải thích: Trong ví dụ trên, ta có hai bộ test:

- Bộ test đầu tiên: Tích các số nguyên liên tiếp từ 3 đến 4 là $3 \times 4 = 12$. Các cặp (x, y) sao cho $LCM(x, y) = 12$ là: $(1, 12)$, $(2, 6)$, $(3, 4)$, $(4, 3)$, $(6, 2)$, $(12, 1)$, $(2, 12)$, $(3, 12)$, $(4, 12)$, $(6, 12)$ và các hoán vị của chúng. Tổng cộng có 15 cặp.
- Bộ test thứ hai: Tích các số nguyên liên tiếp từ 7 đến 7 là 7. Các cặp (x, y) sao cho $LCM(x, y) = 7$ là: $(1, 7); (7, 1); (7, 7)$. Tổng cộng có 3 cặp.

Hướng dẫn giải: Để giải bài toán này, ta cần tính tích các số nguyên liên tiếp từ a đến b và sau đó đếm số cặp (x, y) sao cho $LCM(x, y)$ bằng tích đó. Ta có thể sử dụng công thức:

$$LCM(x, y) = \frac{x \times y}{GCD(x, y)}$$

Trong đó $GCD(x, y)$ là ước số chung lớn nhất của x và y. **Solution C++:**

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    int T;
    cin >> T;
    while (T--) {
        int a, b;
        cin >> a >> b;
        long long product = 1;
```

```
        for (int i = a; i <= b; i++) {
            product *= i;
        }
        int count = 0;
        for (int x = 1; x <= product; x++) {
            for (int y = 1; y <= product; y++) {
                if (lcm(x, y) == product) {
                    count++;
                }
            }
        }
        cout << count % (1000000007) << endl;
    }
    return 0;
}
```

5 ĐỀ HSG 10 THCS TỈNH THÁI BÌNH NĂM HỌC 2024-2025

Thời gian làm bài: 150 phút
Độ khó:

5.1 Bài 1: Số đặc biệt

Đề bài: Nam rất yêu thích các con số, đặc biệt là số nguyên tố. Một lần, trong giờ học Nam nhận được câu hỏi của thầy như sau: Số đặc biệt là một số nguyên dương mà có tổng các chữ số là một số nguyên tố. Cho số nguyên dương N. Hãy kiểm tra xem N có phải là số đặc biệt hay không?

Yêu cầu: Em hãy giúp Nam kiểm tra xem N có phải là số đặc biệt hay không.

Dữ liệu đầu vào: Gồm một dòng duy nhất chứa một số nguyên dương N ($1 \leq N \leq 10^{255}$).

Dữ liệu đầu ra: Gồm một dòng gồm:

- Thông báo: `YES` nếu N là số đặc biệt.
- Thông báo: `NO` nếu N không phải là số đặc biệt.

Ràng buộc dữ liệu:

- Subtask 1 (50%): $0 < N \leq 10^9$.
- Subtask 2 (40%): $10^9 < N \leq 10^{18}$.
- Subtask 3 (10%): $10^{18} < N \leq 10^{255}$.

Ví dụ 1:

Input
23

Output

YES

Giải thích: Số 23 có tổng các chữ số là $2 + 3 = 5$, mà 5 là một số nguyên tố.

Ví dụ 2:

Input

17

Output

NO

Giải thích: Số 17 có tổng các chữ số là $1 + 7 = 8$, mà 8 không phải là một số nguyên tố.

Hướng dẫn giải:

Vì N có thể lớn đến 10^{255} nên kiểu dữ liệu long long sẽ không thể đáp ứng đủ, vì vậy có thể dùng string để nhập số.

Duyệt qua từng chữ số, tính tổng chữ số của N. Sau đó, kiểm tra tổng chữ số của N có phải là số nguyên tố không.

Để kiểm tra số nguyên tố, mình có thể duyệt đến $\sqrt{x}$ (với x là số cần kiểm tra).

n có thể được viết dưới dạng $n = a \times b$, nếu cả a và b đều lớn hơn $\sqrt{x}$, thì $a \times b > \sqrt{n} \times \sqrt{n} = n$.

Vì vậy, sẽ có ít nhất 1 ước của x nhỏ hơn $\sqrt{x}$, nên chỉ cần duyệt đến $\sqrt{x}$ để kiểm tra số nguyên tố.

Độ phức tạp: $O(n\sqrt{n})$.

Solution C++:

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
#define FASTIO ios_base::sync_with_stdio(NULL); cin.tie(NULL);
   cout.tie(NULL);

const ll M = 1e9 + 7;
const ll N = 1e7;

ll sum;
string s;

bool check(ll n) {
    if(n == 1) return false;
    for(ll i = 2; i * i <= n; i++) {
        if(n % i == 0) return false;
    }
    return true;
}

int main() {
```

```
22      FASTIO;
23      cin >> s;
24      for(ll i = 0; i < s.size(); i++) {
25          sum += s[i] - '0';
26      }
27      if(check(sum)) {
28          cout << "YES";
29      } else {
30          cout << "NO";
31      }
32  }
```

5.2 Bài 2: Vườn cây

Đề bài: Một mảng vườn hình chữ nhật được chia thành các ô đất nhỏ gồm M hàng, N cột. Trên các ô đất đó, bác Ba trồng các loại cây ăn quả, cây ở hàng i và cột j có sản lượng quả là a_{ij}. Mỗi đợt cuối năm, bác Ba muốn xem tổng sản lượng quả của các cây trên mỗi hàng dọc (cột) của khu vườn để bác có biện pháp chăm sóc hàng cây đó cho phù hợp.

Yêu cầu: Tính tổng sản lượng trái cây của các cây trên các hàng dọc (cột) trong khu vườn giúp bác Ba.

Dữ liệu đầu vào: Gồm $M + 1$ dòng:

- Dòng đầu tiên chứa hai số nguyên dương M, N $(1 \leq M, N \leq 10^4)$.
- M dòng tiếp theo, mỗi dòng chứa N số nguyên không âm. Giá trị ở dòng thứ i và cột thứ j là a_{ij} với $(1 \leq i \leq M, 1 \leq j \leq N)$ để mô tả sản lượng tại thời điểm thống kê của cây được trồng tại ô ở hàng i cột j của mảnh vườn.

Dữ liệu đầu ra: Gồm một dòng duy nhất chứa N số nguyên dương, mỗi số ghi cách nhau một dấu cách là tổng sản lượng trái cây của các cây trên các hàng dọc (cột) theo thứ tự.

Ràng buộc dữ liệu:

- Có 50% số test (tương ứng 50% số điểm) với $1 \leq M, N \leq 10^2$, $0 \leq a_{ij} \leq 10^3$.
- Có 40% số test (tương ứng 40% số điểm) với $10^2 < M, N \leq 10^3$, $0 \leq a_{ij} \leq 10^8$.
- Có 10% số test (tương ứng 10% số điểm) với $10^3 < M, N \leq 10^4$, $0 \leq a_{ij} \leq 10^{12}$.

Ví dụ 1:

Input

```
3 4
1 3 5 7
2 4 6 9
5 6 9 0
```

Output

```
8 13 20 16
```

Giải thích: Trong ví dụ trên, ta có:

- Cột 1: $1 + 2 + 5 = 8$.
- Cột 2: $3 + 4 + 6 = 13$.
- Cột 3: $5 + 6 + 9 = 20$.
- Cột 4: $7 + 9 + 0 = 16$.

Hướng dẫn giải:
Vì n và m có thể lên đến 10^4 nên mình không thể lưu các giá trị bằng mảng 2 chiều.
Thay vì dùng mảng 2 chiều, mình có thể sử dụng một biến riêng lẻ để nhập đầu vào.
Gọi fakedp[i] là tổng các giá trị trong cột i. Vì không dùng mảng 2 chiều nên khi chạy vòng lặp để nhập giá trị thì mình có thể dựa vào giá trị vòng lặp là cột để cộng vào.
Ví dụ, mình đang nhập giá trị ở cột 1 thì giá trị được nhập sẽ cộng vào fakedp[1].
Nhập giá trị ở cột j thì giá trị được nhập sẽ cộng vào fakedp[j] ($1 \leq j \leq n$).
Độ phức tạp: $O(n \times m)$.
Solution C++:

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
#define FASTIO ios_base::sync_with_stdio(NULL); cin.tie(NULL);
   cout.tie(NULL);

const ll M = 1e9 + 7;
const ll N = 1e7;

int n, m;
ll f, fakedp[10004];

int main() {
    FASTIO;
    cin >> m >> n;
    for(int i = 1; i <= m; i++) {
        for(int j = 1; j <= n; j++) {
            cin >> f;
            fakedp[j] += f;
        }
    }
    for(int i = 1; i <= n; i++) {
        cout << fakedp[i] << "␣";
    }
}
```

5.3 Bài 3: Dãy con thịnh vượng

Xét dãy số gồm n phần tử $a_1, a_2, \ldots, a_n$. Một dãy con liên tiếp của dãy $a_1, a_2, \ldots, a_n$ là dãy số nguyên có dạng $a_i, a_{i+1}, \ldots, a_j$ với $1 \leq i \leq j \leq n$.

Một dãy con liên tiếp được gọi là thịnh vượng nếu tổng các phần tử của nó là một số nguyên dương. Một dãy con liên tiếp được gọi là dãy con thịnh vượng nếu tổng của các phần tử trong dãy con liên tiếp đó là lớn nhất trong tất cả các dãy con liên tiếp.

Yêu cầu: Cho trước một dãy số nguyên $a_1, a_2, \ldots, a_n$. Hãy tìm tổng của một dãy con thịnh vượng của dãy đã cho.

Ví dụ: Cho dãy $5, 3, 7, -9$. Một dãy con thịnh vượng có các phần tử là $5, 3, 7$. Khi đó, tổng của dãy con thịnh vượng là $S = 5 - 3 + 7 = 9$ à tổng các phần tử liên tiếp lớn nhất.

Dữ liệu đầu vào: Gồm hai dòng:

- Dòng đầu tiên chứa một số nguyên dương n ($1 \leq n \leq 10^6$) là số lượng phần tử của dãy.
- Dòng thứ hai chứa n số nguyên $a_1, a_2, \ldots, a_n$ ($-10^9 \leq a_i \leq 10^9$), các số trên cùng dòng viết cách nhau một dấu cách.

Dữ liệu đầu ra: Gồm một số nguyên duy nhất là tổng của dãy con thịnh vượng lớn nhất trong dãy đã cho.

- Subtask 1 (50%): $n \leq 100$.
- Subtask 2 (30%): $100 < n \leq 5000$.
- Subtask 3 (20%): $5000 < n \leq 10^6$.

Ví dụ 1:

Input

```
4
8 -2 7 -17
```

Output

```
13
```

Ví dụ 2:

Input

```
3
2 1 -9
```

Output

```
3
```

Ví dụ 3

Input

```
3
-5 4 -9
```

Output

4

Hướng dẫn giải:

- **Kiến thức cần có**: Quy hoạch động cơ bản, mảng cộng dồn.
- **Subtask 1 + 2:**

  Duyệt qua từng phần tử, với mỗi vị trí i, mình chạy thêm một vòng lặp đến cuối dãy, để xét tất cả dãy liên tiếp có phần tử đầu tiên là a_i.

  Khi xét các dãy liên tiếp, mình sẽ lấy giá trị lớn nhất của tổng các phần tử trong dãy là kết quả bài toán.

  Độ phức tạp: $O(n^2)$.
- **Subtask 3:**
  - **Cách 1:** Quy hoạch động.

    Gọi dp[i] là tổng lớn nhất trong các dãy con liên tiếp kết thúc ở vị trí i.

    Với mỗi a_i, mình sẽ có thể thêm a_i vào dãy trước đó để tính tổng lớn nhất các dãy liên tiếp kết thúc ở vị trí i.

    Hoặc tạo thành một dãy mới bắt đầu và kết thúc ở vị trí i.

    Nên mình sẽ lấy giá trị lớn nhất của hai trường hợp trên.

    Từ đó, mình có công thức truy hồi là: dp[i] = max(dp[$i-1$] + a[i], a[i]).

    Kết quả bài toán là giá trị lớn nhất của dp[i] ($1 \leq i \leq n$).

    Độ phức tạp: $O(n)$.
  - **Cách 2:** Mảng cộng dồn.

    Mình sẽ dùng mảng cộng dồn, tổng dãy con từ L đến R là pre[R] - pre[$L-1$].

    Để dãy con liên tiếp có tổng lớn nhất thì pre[$L-1$] là nhỏ nhất.

    Vì vậy, mình có thể xây dựng mảng min, với mi[i] là giá trị nhỏ nhất của pre[j] (với $1 \leq j \leq i$).

    mi[i] = min(mi[$i-1$], pre[i]).

    Vậy mình chỉ cần lấy giá trị lớn nhất của pre[i] - mi[$i-1$] (với $1 \leq i \leq n$)

    Độ phức tạp: $O(n)$.

Solution C++:

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
#define FASTIO ios_base::sync_with_stdio(NULL); cin.tie(NULL);
   cout.tie(NULL);

const ll M = 1e9 + 7;
const ll N = 2e6;
```

```
9
10  ll n, a[N], ans = -1e18, dp[N], pre[N], mi[N];
11
12  void sub1_2() {
13      for(ll i = 1; i <= n; i++) {
14          ll res = 0;
15          for(ll j = i; j <= n; j++) {
16              res += a[j];
17              ans = max(ans, res);
18          }
19      }
20      cout << ans;
21  }
22
23  void sub3_1() {
24      for(ll i = 1; i <= n; i++) {
25          dp[i] = max(dp[i - 1] + a[i], a[i]);
26          ans = max(ans, dp[i]);
27      }
28      cout << ans;
29  }
30
31  void sub3_2() {
32      for(ll i = 1; i <= n; i++) {
33          ans = max(ans, pre[i] - mi[i - 1]);
34      }
35      cout << ans;
36  }
37
38  int main() {
39      FASTIO;
40      cin >> n;
41      for(ll i = 1; i <= n; i++) {
42          cin >> a[i];
43          pre[i] = pre[i - 1] + a[i];
44          mi[i] = min(mi[i - 1], pre[i]);
45      }
46      if(n <= 5000) sub1_2();
47      else {
48          // sub3_1();
49          sub3_2();
50      }
51  }
```

5.4 Bài 4: Mã mặt hàng

Đề bài: Trong một hệ thống quản lý mặt hàng của một siêu thị, mã mặt hàng được lưu trữ dưới dạng một xâu ký tự hỗn hợp chỉ gồm các chữ cái (in hoa hoặc in thường) và chữ số (các số có mặt trong mã mặt hàng không vượt quá 10^{255}).

Ví dụ, một mã mặt hàng có thể là `789Abc123xyZ456deF789acb1235656`.

Hệ thống quản lý mặt hàng của siêu thị cần tìm ra số lớn nhất xuất hiện trong mã mặt hàng này để phục vụ công tác phân tích và quản lý của siêu thị.

Yêu cầu: Bằng khả năng lập trình của mình em hãy giúp siêu thị thực hiện yêu cầu trên.

Dữ liệu đầu vào: Gồm một dòng duy nhất chứa một xâu ký tự S ($1 \leq |S| \leq 10^6$) là mã mặt hàng cần phân tích. Dữ liệu đảm bảo rằng trong xâu S đảm bảo luôn có chữ số.

Dữ liệu đầu ra: Gồm một dòng duy nhất là số lớn nhất xuất hiện trong mã mặt hàng S.

Ràng buộc dữ liệu:

- Có 50% số test tương ứng với 50% số điểm của bài có độ dài xâu S không quá 255 ký tự và số xuất hiện trong xâu không quá 10^9.
- Có 40% số test tương ứng với 40% số điểm của bài có độ dài xâu S không quá 10^4 ký tự và số xuất hiện trong xâu không quá 10^{18}.
- Có 10% số test tương ứng với 10% số điểm của bài có độ dài xâu S không quá 10^6 ký tự và số xuất hiện trong xâu không quá 10^{255}.

Ví dụ 1:

Input

789Abc123xyZ456deF789acb1235656

Output

1235656

Ví dụ 2:

Input

789aBc0004578978Xyz456Def789aCb1235

Output

4578978

Hướng dẫn giải: Để giải bài toán này, ta cần thực hiện các bước sau:

- Duyệt qua từng ký tự trong xâu S và tách các chuỗi số liên tiếp.
- So sánh các chuỗi số đã tách để tìm ra chuỗi số lớn nhất.
- In ra chuỗi số lớn nhất tìm được.

Solution C++:

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    string S;
    cin >> S;
```

```
6
7       string max_num = "";
8       string current_num = "";
9
10      for (char c : S) {
11          if (isdigit(c)) {
12              current_num += c;
13          } else {
14              if (current_num.length() > max_num.length() ||
15                  (current_num.length() == max_num.length() &&
                        current_num > max_num)) {
16                  max_num = current_num; //
17              }
18              current_num = "";
19          }
20      }
21
22      if (!current_num.empty()) {
23          if (current_num.length() > max_num.length() ||
24              (current_num.length() == max_num.length() &&
                    current_num > max_num)) {
25              max_num = current_num;
26          }
27      }
28
29      cout << max_num << endl;
30
31      return 0;
32  }
```

6 ĐỀ HSG 9 THCS TỈNH HÀ TĨNH 2024 - 2025

Thời gian làm bài: 150 phút
Độ khó:

6.1 Bài 1: Số nguyên dương k

Đề bài: Cho một số nguyên dương $n (n \leq 10^{18})$.
Yêu cầu: Hãy tìm số nguyên dương k lớn nhất thỏa mãn điều kiện $1+2+3+...+k \leq n$.
Dữ liệu đầu vào: Gồm một dòng duy nhất chứa một số nguyên dương n.
Dữ liệu đầu ra: Gồm một số nguyên dương k thỏa mãn yêu cầu bài toán.
Ràng buộc dữ liệu:

- Subtask 1 (80%): $n \leq 10^6$.
- Subtask 2 (20%): Không có ràng buộc gì thêm.

Ví dụ 1:

Input

5

Output

2

Giải thích: Với $n = 5$ thì giá trị $k = 2$ là lớn nhất thỏa mãn $1 + 2 \leq 5$.
Ví dụ 2:

Input

6

Output

3

Giải thích: Với $n = 6$ thì giá trị $k = 3$ là lớn nhất thỏa mãn $1 + 2 + 3 \leq 6$.
Hướng dẫn giải:

- **Kiến thức cần có:** Tìm kiếm nhị phân.
- **Subtask 1:** Duyệt từ 1 đến n, cộng từng số đang duyệt vào tổng, xét nếu tổng lớn hơn n thì in ra $i - 1$, với $i - 1$ là số đã duyệt qua trước đó và kết thúc chương trình.

  Độ phức tạp: $O(n)$.
- **Subtask 2:**

  Đặt $S = 1 + 2 + 3 + ... + k$

  $S = k + (k - 1) + ... + 3 + 2 + 1$

  Cộng S với S theo từng cột ta có: $2S = (k + 1) + (k + 1) + ... + (k + 1) + (k + 1)$

  Số hạng $(k + 1)$ được lặp lại k lần vì có k số hạng trong S

  Suy ra $2S = k \times (k + 1)$, hay $S = \frac{k \times (k+1)}{2}$.

  Nên $1 + 2 + 3 + ... + k = \frac{k \times (k+1)}{2}$.

  Theo đề: $1 + 2 + 3 + ... + k \leq n$

  Suy ra $\frac{k \times (k+1)}{2} \leq n$.

  Ta có: $k \times (k + 1) \leq 2 \times n$.

  $k^2 + k \leq 2 \times n$

  Vì k nguyên dương nên suy ra $k^2 < 2 \times n$.

  $k < \sqrt{2 \times n}$.

  Vì vậy, mình chỉ cần kiểm tra trong đoạn từ 1 đến 2×10^9 vì n có thể lên đến 10^{18}.

  Ta có thể tính tổng $1 + 2 + 3 + ... + k$ bằng công thức $\frac{k \times (k+1)}{2} \leq n$, mình cần tìm k sao cho tổng này không vượt quá n.

Ta thấy, khi k tăng thì $1+2+3+...+k$ cũng tăng, vì vậy ta có thể dùng tìm kiếm nhị phân.

Với phạm vi tìm kiếm là từ 1 đến 2×10^9, nếu $\frac{k \times (k+1)}{2}$ nhỏ hơn hoặc bằng n, ta sẽ tăng k để tìm giá trị k lớn nhất mà thoả mãn điều kiện trên.

Nếu $\frac{k \times (k+1)}{2}$ lớn hơn n giảm giá trị của k để tìm giá trị k thoả mãn điều kiện trên.

Độ phức tạp: $O(\log n)$.

Solution C++:

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
#define FASTIO ios_base::sync_with_stdio(NULL); cin.tie(NULL);
   cout.tie(NULL);

const ll M = 1e9 + 7;
const ll N = 1e7;

ll n;

void sub1() {
    for(ll i = 1; ; i++) {
        if(i * (i + 1) / 2 > n) {
            cout << i - 1;
            return;
        }
    }
}

void sub2() {
    ll l = 1, r = 2e9, res = 0;
    while(l <= r) {
        ll mid = (l + r) / 2;
        if(mid * (mid + 1) / 2 <= n) {
            res = mid;
            l = mid + 1;
        } else r = mid - 1;
    }
    cout << res;
}

int main() {
    FASTIO;
    cin >> n;
    if(n <= 1e6) {
        sub1();
    } else sub2();
}
```

6.2 Bài 2: Nuôi cá cảnh

Đề bài: BigZero có một bể cá với đàn cá nhiều màu sắc. Hằng ngày sau những giờ học bài, cậu thường ngồi ngắm đàn cá và cho chúng ăn. Thức ăn của cá được đựng trong các gói đóng sẵn. Mỗi ngày đàn cá ăn hết đúng 3 gói, giá bán thức ăn thường xuyên biến động. Cửa hàng cho biết trước giá bán trong n ngày lần lượt là $a_1, a_2, ..., a_n$ mỗi ngày được mua nhiều gói với giá bán của ngày đó, thức ăn thừa có thể được dùng cho các ngày tiếp theo. BigZero đang lên kế hoạch để mua thức ăn cho đàn cá trong n ngày sao cho tiết kiệm nhất

Yêu cầu: Cho một số nguyên dương n và các số nguyên dương $a_1, a_2, ..., a_n$ trong đó a_i là giá bán một gói thức ăn trong ngày thứ $i (1 \leq i \leq n \leq 10^6; a_i \leq 10^9)$. Hãy xác định số tiền tối thiểu để mua thức ăn cho đàn cá trong n ngày.

Dữ liệu đầu vào: Gồm hai dòng:

- Dòng thứ nhất chứa một số nguyên dương $n (1 \leq n \leq 10^6)$.
- Dòng thứ hai chứa n số nguyên dương $a_1, a_2, ..., a_n (1 \leq i \leq n; a_i \leq 10^9)$.

Dữ liệu đầu ra: Gồm một số nguyên duy nhất là số tiền tối thiểu để mua thức ăn cho đàn cá trong n ngày.

Ràng buộc dữ liệu:

- Subtask 1 (30%): $a_1 \leq a_2 \leq ... \leq a_n$.
- Subtask 2 (30%): $a_1 \geq a_2 \geq ... \geq a_n$.
- Subtask 3 (40%): Không có ràng buộc gì thêm.

Ví dụ 1:

Input

```
3
2 3 5
```

Output

```
18
```

Giải thích: Kế hoạch mua thức ăn là:

- Ngày 1 mua 9 gói với giá là 2.
- Ngày 2, 3 không mua gói nào.

Số tiền tối thiểu để mua thức ăn là $9 \times 2 + 0 \times 3 + 0 \times 5 = 18$.

Ví dụ 2:

Input

```
3
5 3 2
```

Output

30

Giải thích: Kế hoạch mua thức ăn là:

- Ngày 1 mua 3 gói với giá là 5.
- Ngày 2 mua 3 gói với giá là 3.
- Ngày 3 mua 3 gói với giá là 2.

Số tiền tối thiểu để mua thức ăn là $3 \times 5 + 3 \times 3 + 3 \times 2 = 30$.

Ví dụ 3:

Input

3
5 2 3

Output

27

Giải thích: Kế hoạch mua thức ăn là:

- Ngày 1 mua 3 gói với giá là 5.
- Ngày 2 mua 6 gói với giá là 2.
- Ngày 3 không mua gói nào.

Số tiền tối thiểu để mua thức ăn là $3 \times 5 + 6 \times 2 + 0 \times 3 = 27$.

Hướng dẫn giải:

- **Subtask 1:**

  Với a là dãy tăng dần, nên giá bán trong ngày thứ 1 là giá trị nhỏ nhất trong dãy.

  Vì vậy, mình chỉ cần mua thức ăn cho đàn cá trong cả n ngày với giá ngày đầu tiên.

  Với mỗi ngày 3 gói, nên giá của tổng tất cả n ngày là $3 \times n \times$ giá ngày đầu.

  Độ phức tạp: $O(1)$.

- **Subtask 2:**

  Với a là dãy giảm dần, nên giá bán trong ngày i luôn lớn hơn các ngày $i + 1$.

  Vì vậy, với mỗi ngày mình sẽ mua 3 gói với giá của ngày đó, vì giá ngày sau a_{i+1} luôn nhỏ hơn giá của ngày trước a_i nên không thể mua cho ngày i với giá của ngày $i + 1$ vì giá sẽ đắt.

  Vì mỗi ngày mua 3 gói với giá của ngày hôm đó, nên giá của tổng tất cả n ngày là $(3 \times a_1 + 3 \times a_2 + ... + 3 \times a_n) = 3 \times (a_1 + a_2 + ... + a_n)$.

  Độ phức tạp: $O(n)$.

- **Subtask 3:**

  Dễ dàng nhận thấy rằng, nếu giá của ngày i nhỏ hơn các ngày sau đó, thì có thể mua cho những ngày sau.

  Vì vậy, với mỗi ngày i, mình sẽ mua 3 gói cho ngày đó với giá nhỏ nhất từ ngày 1 đến ngày i.

  Độ phức tạp: $O(n)$.

Solution C++:

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
#define FASTIO ios_base::sync_with_stdio(NULL); cin.tie(NULL);
   cout.tie(NULL);

const ll M = 1e9 + 7;
const ll N = 1e7;

ll n, a[N], mi = 1e18, d, res, checksub1 = 1, checksub2 = 1;

void sub1() {
    cout << 3 * a[1] * n;
}

void sub2() {
    for(ll i = 1; i <= n; i++) {
        res += 3 * a[i];
    }
    cout << res;
}

void sub3() {
    for(ll i = 1; i <= n; i++) {
        if(mi > a[i]) {
            mi = a[i];
        }
        res += 3 * mi;
    }
    cout << res;
}

int main() {
    FASTIO;
    cin >> n;
    for(ll i = 1; i <= n; i++) {
        cin >> a[i];
        if(a[i] > a[i - 1] && i > 1) {
            checksub2 = 0;
        }
        if(a[i] < a[i - 1] && i > 1) {
```

```
42              checksub1 = 0;
43          }
44      }
45
46      if(checksub1 == 1) {
47          sub1();
48          return 0;
49      }
50      if(checksub2 == 1) {
51          sub2();
52          return 0;
53      }
54      sub3();
55  }
```

6.3 Bài 3: Số nguyên tố

Đề bài: Số nguyên tố là số tự nhiên lớn hơn 1 và chỉ có đúng hai ước là 1 và chính nó. Ví dụ các số tự nhiên $2, 3, 5, 7, 11, 13, 17, 19, 23, ...$ là các số nguyên tố.

Yêu cầu: Cho số tự nhiên n, hãy tìm số tự nhiên p thỏa mãn điều kiện p là số nguyên tố nhỏ nhất và $p \geq n$.

Dữ liệu đầu vào: Gồm hai dòng:

- Dòng thứ nhất chứa số nguyên dương $Q (Q \leq 10^6)$ là số bộ test.
- Q dòng tiếp theo, mỗi dòng chứa một số tự nhiên $n (n \leq 10^9)$.

Dữ liệu đầu ra: Gồm Q dòng, mỗi dòng ghi một số nguyên tố tìm được tương ứng với Dữ liệu đầu vào.

Ràng buộc dữ liệu:

- Có 30% số test ứng với 30% số điểm của bài thỏa mãn: $Q = 1, n \leq 10^3$;
- Có 40% số test khác ứng với 40% số điểm của bài thỏa mãn: $Q \leq 10^2, n \leq 10^9$;
- 30% số test còn lại ứng với 30% số điểm của bài thỏa mãn: $Q \leq 10^6, n \leq 10^6$.

Ví dụ 1:

Input

```
2
5
8
```

Output

```
5
11
```

Giải thích:

- Với $n = 5$, số nguyên tố nhỏ nhất $p \geq n$ là 5.

- Với $n = 8$, số nguyên tố nhỏ nhất $p \geq n$ là 11.

Hướng dẫn giải:
Solution C++:

6.4 Bài 4: Dãy con

Đề bài: Cho một dãy a gồm n số nguyên dương $a_1, a_2, ..., a_n$ và một số nguyên dương m.
Yêu cầu: Hãy tìm số nguyên dương L nhỏ nhất sao cho tất cả các dãy con gồm L phần tử liên tiếp của dãy a đều có tổng lớn hơn hoặc bằng m.
Dữ liệu đầu vào: Gồm hai dòng:

- Dòng thứ nhất chứa hai số nguyên dương n và $m (1 \leq n \leq 10^6; m \leq 10^{18})$.
- Dòng tiếp theo chứa n số nguyên dương $a_1, a_2, ..., a_n (1 \leq i \leq n; a_i \leq 10^9)$.

Dữ liệu đầu ra: Gồm một số nguyên dương L nhỏ nhất tìm được thỏa mãn yêu cầu bài toán. Nếu không tìm được giá trị thỏa mãn thì ghi -1.
Ràng buộc dữ liệu:

- Subtask 1 (30%): $a_1 \leq a_2 \leq ... \leq a_n$.
- Subtask 2 (40%): $n \leq 10^3$.
- Subtask 3 (30%): Không có ràng buộc gì thêm.

Ví dụ 1:

Input

```
5 6
3 2 1 4 5
```

Output

```
3
```

Ví dụ 2:

Input

```
4 16
7 1 2 5
```

Output

```
-1
```

Hướng dẫn giải:

- **Kiến thức cần có:** Tìm kiếm nhị phân, mảng cộng dồn.

- **Subtask 1:**

  Vì dãy a là một dãy tăng dần, nên trong các dãy con liên tiếp có độ dài L.

  Dãy con bắt đầu từ phần tử đầu tiên sẽ luôn có giá trị nhỏ nhất. Điều này là do trong dãy tăng dần, mỗi phần tử sau luôn lớn hơn các phần tử trước đó. Do đó, các dãy con liên tiếp có độ dài L bắt đầu từ các phần tử sau sẽ luôn có giá trị lớn hơn dãy con bắt đầu từ phần tử đầu tiên.

  Độ phức tạp: $O(n)$.

- **Subtask 2:**

  Với $n \leq 10^3$ thì mình có thể duyệt hết tất cả các dãy liên tiếp trong a có độ dài $L(1 \leq L \leq n)$ để tìm giá trị L nhỏ nhất.

  Độ phức tạp: $O(n^2)$.

- **Subtask 3:**

  Nhận thấy các phần tử có giá trị dương, nên khi độ dài của dãy liên tiếp tăng thì tổng giá trị dãy đó cũng tăng.

  Từ đó, ta có thể tối ưu từ **Subtask 2** bằng cách sử dụng tìm kiếm nhị phân.

  Ta tìm kiếm nhị phân trong khoảng từ 1 đến n. Với mỗi mid, ta xét hết tất cả dãy liên tiếp có độ dài mid. Nếu tất cả dãy độ dài mid lớn hơn m thì giảm mid xuống để tìm độ dài nhỏ nhất. Nếu không thì tăng mid vì độ dài dãy tăng thì tổng giá trị tăng.

  Để xét tổng các dãy có độ dài L thì ta có thể dùng mảng cộng dồn.

  Độ phức tạp: $O(n \log n)$.

Solution C++:

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
#define FASTIO ios_base::sync_with_stdio(NULL); cin.tie(NULL);
   cout.tie(NULL);

const ll M = 1e9 + 7;
const ll N = 1e6;

ll n, m, a[N + 10], check = 1, lol;
vector<ll> f(N + 10, 0);

void sub1() {
    for(ll i = 1; i <= n; i++) {
        if(f[i] >= m) {
            cout << i;
            return;
        }
    }
    cout << -1;
}
```

```
22
23  void sub2() {
24      ll res = 0;
25      for(ll i = 1; i <= n; i++) {
26          ll check2 = 1;
27          for(ll j = 1; j <= n - i + 1; j++) {
28              if(f[j + i - 1] - f[j - 1] >= m) {
29                  res = i;
30              } else {
31                  res = 0;
32                  check2 = 0;
33                  break;
34              }
35          }
36          if(check2 == 1) break;
37      }
38      if(res == 0) cout << -1;
39      else cout << res;
40  }
41
42  ll checkday(ll x) {
43      for(ll i = 1; i <= n - x + 1; i++) {
44          if(f[i + x - 1] - f[i - 1] < m) {
45              return 0;
46          }
47      }
48      return 1;
49  }
50
51  void sub3() {
52      ll res = 0, l = 1, r = n;
53      while(l <= r) {
54          ll mid = (l + r) / 2;
55          if(checkday(mid) == 1) {
56              res = mid;
57              r = mid - 1;
58          } else {
59              l = mid + 1;
60          }
61      }
62      if(res == 0) cout << -1;
63      else cout << res;
64  }
65
66  int main() {
67      FASTIO;
68      f[0] = 0;
69      cin >> n >> m;
70      for(ll i = 1; i <= n; i++) {
71          cin >> a[i];
72          f[i] = f[i - 1] + a[i];
```

```
73          if(a[i] < a[i - 1]) check = 0;
74      }
75      if(check == 1) {
76          sub1();
77      } else if(n <= 1000) {
78          sub2();
79      } else {
80          sub3();
81      }
82  }
```

7 ĐỀ HSG 9 THCS TỈNH VĨNH PHÚC 2024 - 2025

7.1 Bài 1: Quân Hậu

Đề bài: Huy là một học sinh yêu thích cờ vua, toán học và lập trình. Huy biết rằng quân cờ mạnh nhất trên bàn cờ vua là quân Hậu, vì nó có thể di chuyển như quân Xe (trên cùng một cột hoặc một hàng) và như quân Tượng (theo đường chèo).

Yêu cầu: Huy có một bàn cờ hình chữ nhật kích thước $N \times M$. Huy muốn biết nếu đặt một quân Hậu lên bàn cờ này thi số lượng ô tối đa mà nó có thể kiểm soát là bao nhiêu. Chẳng hạn, nếu $N = M = 8$ thì một quân Hậu có thể kiểm soát tối đa 27 ô (không tính ô đặt quân Hậu, xem giải thích test ví dụ 1).

Dữ liệu đầu vào: Gồm hai dòng:

- Dòng 1: số nguyên $N (1 \leq N \leq 10^9)$ là kích thước bàn cờ theo chiều dọc.
- Dòng 2: số nguyên $M (1 \leq M \leq 10^9)$ là kích thước bàn cờ theo chiều ngang.

Dữ liệu đầu ra: Gồm một số nguyên là số lượng ô tối đa mà quân Hậu có thể kiểm soát trên bàn cờ kích thước $N \times M$.

Ràng buộc dữ liệu:

- Subtask 1 (42%): $N, M \leq 10$.
- Subtask 2 (38%): $N, M \leq 500$.
- Subtask 3 (20%): Không có ràng buộc gì thêm.

Ví dụ 1:

Input

```
8
8
```

Output

```
27
```

Giải thích:

x				x			
	x			x			x
		x		x		x	
			x	x	x		
x	x	x	x	Q	x	x	x
			x	x	x		
		x		x		x	
	x			x			x

Ví dụ 2:

Input

```
3
4
```

Output

```
9
```

Giải thích:

x	x	x	
x	Q	x	x
x	x	x	

Hướng dẫn giải:
Solution C++:

7.2 Bài 2: Trung vị lớn nhất

Đề bài: Trung vị của một dãy số $X = (x_1, x_2, ..., x_N)$ được xác định như sau:

- Xét dãy $Y = (y_1, y_2, ..., y_N)$ là kết quả của việc sắp xếp dãy X theo thứ tự không giảm;
- Nếu $N = 2k$ trung vị dãy X là y_k, nếu $N = 2k + 1$, trung vị của dãy X là y_{k+1}.

Chẳng hạn, trung vị của dãy $X = (3, 1, 2, 4)$ là 2, trung vị của dãy $X = (1, 3, 2, 3, 5)$ là 3.

Huy có một dãy số $A = (a_1, a_2, ..., a_N)$. Huy muốn biến đổi dãy số về dạng dãy hằng (dãy có tất cả các phần tử bằng nhau) bằng cách sử dụng số lần phép biến đổi:

- Chọn hai chỉ số l và $r(1 \leq l < r \leq N)$, gọi x là trung vị của đoạn con $(a_l, a_{l+1}, ..., a_r)$;
- Gán tất cả các phần tử $a_l, a_{l+1}, ..., a_r$ thành x.

Chẳng hạn, nếu $A = (1, 3, 5, 2, 4)$, thực hiện biến đổi trên v ới $l = 3$ và $r = 4$ thì dãy trở thành $A = (1, 3, 2, 2, 4)$.

Yêu cầu: Hãy giúp Huy xác định giá trị lớn nhất của phần tử dãy hằng có thể nhận được từ dãy A.

Dữ liệu đầu vào: Gồm hai dòng:

- Dòng 1: số nguyên $N(2 \leq N \leq 10^5)$;
- Dòng 2: N số nguyên $a_1, a_2, ..., a_N(1 \leq a_i \leq 10^9, \forall i = 1...N)$.

Dữ liệu đầu ra: Gồm một số nguyên ra kết quả

Ràng buộc dữ liệu:

- Subtask 1 (30%): $2 \leq N \leq 10^2; 1 \leq a_i \leq 10^5, \forall i$;
- Subtask 2 (30%): $10^2 \leq N \leq 10^3; 10^5 \leq a_i \leq 10^6, \forall i$;
- Subtask 3 (40%): Không có rằng buộc gì thêm.

Ví dụ:

Input

```
5
1 2 3 4 5
```

Output

```
4
```

Giải thích: Có thể thực hiện 3 phép biến đổi sau:

- $(l, r) = (4, 5)$ thì dãy mới $A = [1, 2, 3, 4, 4]$
- $(l, r) = (3, 5)$ thì dãy mới $A = [1, 2, 4, 4, 4]$
- $(l, r) = (1, 5)$ thì dãy mới $A = [4, 4, 4, 4, 4]$

Hướng dẫn giải:

Solution C++:

7.3 Bài 3: Xâu rút gọn

Đề bài: Một xâu A được gọi là rút gọn của xâu B nếu ta có thể tạo ra A bằng cách xóa đi 0 hoặc nhiều ký tự trong B mà không thay đổi thứ tự các ký tự còn lại. Theo định nghĩa này, một xâu luôn là xâu rút gọn của chính nó.

Chẳng hạn:

- `ac`, `ab`, `aa` là các xâu rút gọn của `aabc`;
- `d`, `aaa`, `ba` không phải là xâu rút gọn của `aabc`.

Yêu cầu: Cho hai xâu S và T chỉ gồm các ký tự chữ cái thường trong bảng chữ cái tiếng Anh. Gọi T^n là xâu được tạo ra bằng cách nối n xâu T lại với nhau. Hãy tìm giá trị nhỏ nhất của n sao cho S là một xâu rút gọn của xâu T^n.

Dữ liệu đầu vào: Gồm hai dòng:

- Dòng 1: xâu S với độ dài $|S|(1 \leq |S| \leq 10^6)$.
- Dòng 2: xâu T với độ dài $|T|(1 \leq |T| \leq 10^5)$.

Dữ liệu đầu ra: Gồm một số nguyên là giá trị n nhỏ nhất sao cho S là xâu rút gọn của T. Nếu không tồn tại giá trị n như vậy thì in ra -1.

Ràng buộc dữ liệu:

- Subtask 1 (8%): S và T chỉ chứa ký tự `a`.
- Subtask 2 (13%): $|S|, |T| \leq 100$.
- Subtask 3 (21%): $|S| \leq 10^4, |T| \leq 100$.
- Subtask 4 (34%): $|T| \leq 1000$.
- Subtask 5 (24%): Không có ràng buộc bổ sung.

Ví dụ 1:

Input

```
caa
ac
```

Output

```
3
```

Giải thích: Ta có: $T^1 = T =$ `ac`, $T^2 = T =$ `acac`, $T^3 = T =$ `acacac`; $n = 3$ là giá trị nhỏ nhất để xâu S trở thành xâu rút gọn của T^n.

Ví dụ 2:

Input

```
cab
acca
```

Output

```
-1
```

Giải thích: Không tìm được n thỏa mãn điều kiện.

Hướng dẫn giải:

Solution C++:

7.4 Bài 4: Dãy đẹp

Đề bài: Cho dãy $A = (a_1, a_2, ..., a_n)$. Độ đẹp của dãy A được định nghĩa là tổng lớn nhất của một đoạn con liên tiếp (có thể rỗng) của dãy. Chẳng hạn, dãy $A = (-3, 8, 4, -2, 12)$ có độ đẹp bằng 22 (đoạn con $(8, 4, -2, 12)$), dãy $B = (-1, -2, -3, -4, -5)$ có độ đẹp bằng 0 (đoạn con rỗng).

Yêu cầu: Để gia tăng độ đẹp của dãy A, bạn được phép chọn tối đa một đoạn con liên tiếp của dãy và nhân từng phần tử trong đoạn con đó lên X lần. Xác định độ đẹp lớn nhất có thể đạt được của dãy.

Dữ liệu đầu vào: Gồm hai dòng:

- Dòng 1: Hai số nguyên $N, X (1 \leq N \leq 4 \times 10^5; -100 \leq X \leq 100)$;
- Dòng 2: N số nguyên $a_1, a_2, ..., a_N (|a_i| \leq 10^9)$.

Dữ liệu đầu ra: Gồm một số nguyên là độ đẹp tối đa của dãy A sau khi thực hiện không quá một thao tác nói trên.

Ràng buộc dữ liệu:

- 20% số điểm dành cho các test có $1 \leq N \leq 50$;
- 30% số điểm dành cho các test có $1 \leq N \leq 300$;
- 20% số điểm dành cho các test có $a_i \geq 0$ với mọi i;
- 30% số điểm còn lại không có ràng buộc bổ sung.

Ví dụ 1:

Input

```
5 -2
-3 8 -2 1 -6
```

Output

```
22
```

Giải thích: Thực hiện thao tác với đoạn $[-2, 1, -6]$ thu được dãy $[-3, 8, 4, -2, 12]$. Dãy này có độ đẹp là 22. Đây là độ đẹp lớn nhất có thể đạt được.

Ví dụ 2:

Input

```
8 -4
1 2 1 1 2 0 0 7
```

Output

```
14
```

Giải thích: Không cần thực hiện thao tác nào.

Ví dụ 3:

Input

5 10
-1 -2 -3 -4 -5

Output

0

Hướng dẫn giải:
Solution C++:

8 ĐỀ HSG 9 THCS TỈNH HẢI PHÒNG 2024 - 2025

Thời gian làm bài: 150 phút
Độ khó:

8.1 Bài 1: Tam giác vuông

Đề bài:
Cho một số nguyên dương A.
Yêu cầu:
Viết chương trình kiểm tra xem A có phải là diện tích của một tam giác vuông có các cạnh là số nguyên hay không. Nếu có in ra `YES`, nếu không in ra `NO`.
Dữ liệu đầu vào:
Gồm $T + 1$ dòng:

- Dòng đầu chứa số nguyên $T(T \leq 1000)$ là số lượng số A cần kiểm tra.
- T dòng tiếp theo, mỗi dòng ghi một số $A(A \leq 10^6)$.

Dữ liệu đầu ra:
Gồm T dòng, mỗi dòng là một chữ `YES` hoặc `NO` tương ứng với dữ liệu đề bài.
Ràng buộc dữ liệu:

- Subtask 1 (20%): $T = 2, A \leq 100$.
- Subtask 2 (30%): $2 < T \leq 100, A \leq 100$.
- Subtask 3 (50%): Không có ràng buộc gì thêm.

Ví dụ 1:

Input

3
6
24
50

Output

YES
YES
NO

Giải thích:

- Với $A = 6$ tam giác vuông thỏa mãn yêu cầu có các cạnh lần lượt là: $3; 4; 5$
- Với $A = 24$ tam giác vuông thỏa mãn yêu cầu đề bài có các cạnh lần lượt là: $6; 8; 10$.
- Với $A = 50$ không có tam giác vuông nào thỏa mãn yêu cầu đề bài.

Hướng dẫn giải:
Solution C++:

8.2 Bài 2: Số chính phương

Đề bài:

Cho một xâu S chỉ gồm các ký tự chữ cái trong bảng chữ cái tiếng Anh và các chữ số từ `0` đến `9`. Một số trong xâu S được định nghĩa là một ký tự chữ số hoặc là các kí tự số liên tiếp và không bao gồm các chữ số `0` không có nghĩa.

Ví dụ với xâu $S =$ `05aAb21b3956cDe488a`, các số có trong xâu là: $5; 21; 3956; 488$.

Yêu cầu:

Cho xâu S chỉ gồm các ký tự chữ cái tiếng Anh và các chữ số. Hãy viết chương trình tìm số chính phương lớn nhất trong xâu S. (Số chính phương là số bằng bình phương của một số nguyên, ví dụ 9 là số chính phương vì $9 = 3^2$).

Dữ liệu đầu vào:

Gồm xâu S chỉ chứa các ký tự chữ cái trong bảng chữ cái tiếng Anh và chữ số (dữ liệu đảm bảo S có không quá 18 chữ số có nghĩa liền nhau và độ dài xâu không quá 10^5 ký tự).

Dữ liệu đầu ra:

Gồm số chính phương lớn nhất tìm được hoặc số -1 nếu không tìm được số chính phương nào.

Ràng buộc dữ liệu:

- Subtask 1 (30%): $|S| \leq 250$.
- Subtask 2 (30%): $|S| \leq 10^3$.
- Subtask 3 (40%): Không có ràng buộc gì thêm.

Ví dụ:

Input

aBc2144gHf81Dgf09gf

Output

81

Giải thích:
Trong xâu có các số: $2144; 81; 9$. Số chính phương lớn nhất là 81.

Input

dGaf21eac056Ude00132aV

Output

-1

Giải thích: Trong xâu có các số: $21; 56; 132$. Không có số chính phương nào.
Hướng dẫn giải:
Solution C++:

8.3 Bài 3: Mượn sách

Đề bài: Bạn An có một bộ sách hay và muốn chia sẻ chúng với các bạn trong câu lạc bộ đọc sách của trường. Có N yêu cầu được mượn sách này từ các bạn trong câu lạc bộ, yêu cầu thứ $i(1 \leq i \leq N)$ cho biết thời điểm mượn sách là a_i và thời điểm trả sách là b_i. Bạn An có thể chấp nhận hoặc từ chối đối với một yêu cầu.

Yêu cầu: Hãy lập trình giúp bạn An chọn các yêu cầu mượn sách của các bạn sao cho đáp ứng được nhiều yêu cầu nhất. Đảm bảo khoảng thời gian sử dụng của hai yêu cầu không giao nhau.

Dữ liệu đầu vào:

- Dòng đầu tiên chứa số nguyên dương $N(1 \leq N \leq 10^4)$ là số lượng yêu cầu mượn sách.
- N dòng tiếp theo, dòng thứ $i(1 \leq i \leq N)$ chứa hai số nguyên dương $a_i, b_i(1 \leq a_i < b_i \leq 32000)(1 \leq i \leq N)$.

Dữ liệu đầu ra: Một số nguyên K là số các yêu cầu được chấp nhận.
Ràng buộc dữ liệu:

- Subtask 1 (30%): $N \leq 100; a_i < b_i \leq 10^3$.
- Subtask 2 (30%): $100 < N \leq 10^3; a_i < b_i \leq 10^3$.
- Subtask 3 (40%): Không có ràng buộc gì thêm.

Ví dụ:

Input

```
5
7 9
2 4
1 3
1 6
4 7
```

Output

3

Giải thích: Các yêu cầu được chấp thuận là $(1, 3), (4, 7), (7, 9)$.
Hướng dẫn giải:
Solution C++:

8.4 Bài 4: Số đặc biệt

Đề bài: Số X được gọi là số đặc biệt nếu tất cả các chữ số của X đều thuộc tập hợp $\{1; 3; 5; 7; 9\}$. Người ta tạo ra các số đặc biệt, sau đó sắp xếp chúng theo thứ thự tăng dần để được một dãy số A. Ví dụ 20 số đặc biệt đầu tiên:

$$1; 3; 5; 7; 9; 11; 13; 15; 17; 19; 31; 33; 35; 37; 39; 51; 53; 55; 57; 59.$$

Yêu cầu:: Cho số nguyên dương N, hãy tìm số đặc biệt thứ N trong dãy số A.
Dữ liệu đầu vào: Gồm một dòng duy nhất chứa số nguyên dương $N (1 \leq N \leq 10^{18})$.
Dữ liệu đầu ra: Gồm một dòng duy nhất chứa số đặc biệt thứ N trong dãy số A.
Ràng buộc dữ liệu:

- Subtask 1 (50%): $N \leq 10^6$.
- Subtask 2 (30%): $10^6 < N \leq 10^9$.
- Subtask 3 (20%): Không có ràng buộc gì thêm.

Ví dụ 1:

Input

8

Output

15

Giải thích: Số đặc biệt thứ 8 trong dãy số A là 15.
Ví dụ 2:

Input

29

Output

97

Giải thích: Số đặc biệt thứ 29 trong dãy số A là 97.

9 ĐỀ HSG 9 THCS TP. HỒ CHÍ MINH 2024 - 2025

Thời gian làm bài: 150 phút Độ khó:

9.1 Bài 1: Sắp xếp

Đề bài: Sắp xếp nổi bọt (Bubble Sort) là một trong những thuật toán đơn giản và dễ hiểu. Thuật toán sắp xếp nổi bọt thực hiện sắp xếp dãy phần tử bằng cách liên tục lặp lại việc so sánh hai phần tử liền kề và hoán đổi vị trí của chúng nếu chúng không theo thứ tự mong muốn. Quá trình này được lặp lại cho đến khi toàn bộ dãy được sắp xếp hoàn chỉnh.

Yêu cầu: Cho một dãy gồm n phần tử hãy viết chương trình đếm số lần hoán đổi vị trí các phần tử theo thuật toán sắp xếp nổi bọt để sắp xếp dãy tăng dần.

Dữ liệu đầu vào: Gồm hai dòng:

- Dòng 1: Số nguyên dương $n(1 \leq n \leq 2 \times 10^5)$ là số phần tử trong dãy.
- Dòng 2: n số nguyên $a_1, a_2, ..., a_n(1 \leq a_i \leq 10^9, \forall i = 1...n)$ là các phần tử trong dãy.

Dữ liệu đầu ra: Gồm một số nguyên là số lần hoán đổi vị trí các phần tử theo thuật toán sắp xếp nổi bọt để sắp xếp dãy tăng dần.

Ràng buộc dữ liệu:

- Subtask 1 (80%): $1 \leq n \leq 1000$.
- Subtask 2 (20%): $1 \leq n \leq 2 \times 10^5$.

Ví dụ:

Input

```
4
3 2 1 4
```

Output

```
3
```

Giải thích: Theo thuật toán sắp xếp nổi bọt có 3 lần hoán đổi vị trí các phần tử gồm:

- Lần 1: Hoán đổi 3 và 2 được dãy $[2, 3, 1, 4]$.
- Lần 2: Hoán đổi 3 và 1 được dãy $[2, 1, 3, 4]$.
- Lần 3: Hoán đổi 2 và 1 được dãy $[1, 2, 3, 4]$.

Hướng dẫn giải:
Solution C++:

9.2 Bài 2: Khu vực

Đề bài: Vùng đất thần tiên AlphaLand rộng lớn được chia thành nhiều khu vực khác nhau. Các khu vực được đánh số $1, 2, 3, ...$

Việc phân chia khu vực sinh sống, lao động, vui chơi cho người dân cũng khá kì lạ. Mỗi người dân được cấp một cái thẻ chứa một con số và họ chỉ được phép ra vào khu vực có số thứ tự là ước số của số thẻ. Các số trên thẻ của người dân được phép trùng nhau.

Mỗi dịp lễ hội thường niên, trưởng lão sẽ tập trung tất cả người dân về một khu vực để tổ chức tiệc mừng. Năm nay, ông quyết định mở tiệc tại khu vực mà tất cả người dân được phép ra vào khu vực đó có số thứ tự lớn nhất.

Nhận thấy rằng có một số thẻ đã cấp cho người dân làm ảnh hưởng đến việc chọn khu vực như trên, ông quyết định đổi cho một trong những số họ cái thẻ mới để chọn được khu vực tổ chức tiệc có số thứ tự lớn hơn.

Yêu cầu: Cho danh sách n thẻ với các số tương ứng. Hãy viết chương trình tìm khu vực mà tất cả người dân được phép ra vào và khu vực đó có số thứ tự lớn nhất. Lưu ý, việc xác định khu vực được thực hiện sau khi người dân được đổi thẻ.

Dữ liệu đầu vào: Gồm hai dòng:

- Dòng 1: Số nguyên dương $n(1 \leq n \leq 10^5)$ là số thẻ.
- Dòng 2: n số nguyên dương $a_1, a_2, ..., a_n(1 \leq a_i \leq 10^9, \forall i = 1...n)$ là các số trên thẻ.

Dữ liệu đầu ra: Gồm một số nguyên là số thứ tự khu vực mà tất cả người dân được phép ra vào và khu vực đó có số thứ tự lớn nhất.

Ràng buộc dữ liệu:

- Subtask 1 (40%): $1 \leq n \leq 100; 1 \leq a_i \leq 100, \forall i$.
- Subtask 2 (40%): $1 < n \leq 1000$.
- Subtask 3 (20%): Không có ràng buộc gì thêm.

Ví dụ:

Input

```
3
4 2 8
```

Output

```
4
```

Giải thích:

Thẻ số 4 đến được các khu vực $1, 2, 4$.

Thẻ số 2 đến được các khu vực $1, 2$.

Thẻ số 8 đến được các khu vực $1, 2, 4, 8$.

Ban đầu khu vực dự kiến chọn là 2. Có thể đổi thẻ số 2 thành số 4 với các thẻ $4, 4, 8$ để chọn khu vực 4.

Hướng dẫn giải:

Solution C++:

9.3 Bài 3: Giải đấu

Đề bài: Trong mùa hè năm nay, công ty game AlphaNet sẽ tổ chức giải đấu trực tuyến cho tất cả game thủ của mình. Danh sách tên các game thủ được đánh số thứ tự từ 1 đến n. Mỗi game thủ có một ranking (thứ hạng trong game) khác nhau. Trước khi bắt đầu giải, ban tổ chức sẽ cho các game thủ giao lưu với nhau thông qua q lượt đấu đồng đội bằng cách ghép đôi ngẫu nhiên ở mỗi lượt đấu, ban tổ chức thực hiện ghép đôi ngẫu nhiên như sau:

- Cho hệ thống sinh ngẫu nhiên hai số $u, v(u < v)$ để xác định nhóm các thành viên được chọn tham gia là các game thủ trong danh sách có số thứ tự từ u đến v.
- Tiếp theo ban tổ chức sẽ chia nhóm các thành viên được chọn thành 2 đội tương đối đều nhau về ranking.

Ranking của 1 đội là tổng ranking của các thành viên trong đội. Do đó, ban tổ chức phải tính toán để xác định 1 vị trí sao cho đội lệch ranking giữa 2 đội là nhỏ nhất và các thành viên trong đội phải có số thứ tự liên tiếp nhau trong danh sách (không quan trọng số lượng thành viên trong đội).

Yêu cầu: Cho một dãy n game thử với ranking tương ứng với q cặp số nguyên u, v được hệ thống sinh ngẫu nhiên, hãy viết chương trình cho biết độ lệch ranking nhỏ nhất của từng lượt sau khi ban tổ chức thực hiện ghép đội ngẫu nhiên.

Dữ liệu đầu vào: Gồm $q + 2$ dòng:

- Dòng 1: Hai số nguyên dương $n, q(1 \leq n, q \leq 10^5)$ là số game thủ và số lượt đấu.
- Dòng 2: n số nguyên dương $a_1, a_2, ..., a_n(1 \leq a_i \leq 10^9, \forall i = 1...n)$ là ranking của các game thủ.
- q dòng tiếp theo, dòng thứ $i(1 \leq i \leq q)$ chứa hai số nguyên dương $u_i, v_i(1 \leq u_i < v_i \leq n)$ là cặp số được hệ thống sinh ngẫu nhiên.

Dữ liệu đầu ra: Gồm q dòng, dòng thứ $i(1 \leq i \leq q)$ chứa độ lệch ranking nhỏ nhất của lượt đấu thứ i.

Ràng buộc dữ liệu:

- Subtask 1 (40%): $1 \leq n, q \leq 100$.
- Subtask 2 (40%): $1 \leq n \leq 10^3; 1 \leq q \leq 10^4$.
- Subtask 3 (20%): Không có ràng buộc gì thêm.

Ví dụ:

Input

```
6 2
4 2 2 1 5 6 1 4 2 5
```

Output

```
1
0
```

Giải thích:

Ở lượt thứ 1, ranking của các game thủ có số thứ tự từ 1 đến 4 là: 4221. Ta có thể chia được như sau:

- $\{4\}$ và $\{2, 2, 1\}$. Độ lệch ranking giữa 2 đội là 1.
- $\{4, 2\}$ và $\{2, 1\}$. Độ lệch ranking giữa 2 đội là 3.
- $\{4, 2, 2\}$ và $\{1\}$. Độ lệch ranking giữa 2 đội là 7.

Ban tổ chức sẽ chia đội theo cách chia đầu tiên với độ lệch ranking nhỏ nhất là 1.

Ở lượt thứ 2, ranking của các game thủ có số thứ tự từ 2 đến 5 là: 2215. Ta có thể chia được với độ lệch ranking giữa 2 đội là 0.

Hướng dẫn giải:

Solution C++:

10 ĐỀ HSG 9 THCS THÀNH PHỐ ĐÀ NẴNG 2024 - 2025

10.1 Bài 1: Kí Tự

Đề bài: Cho một xâu s chỉ chứa các kí tự in hoa trong bảng chữ cái $A...Z$.

Yêu cầu: Hãy kiểm tra xâu s còn thiếu những ký tự nào trong bảng chữ cái tiếng Anh $A...Z$.

Dữ liệu đầu vào: Một dòng duy nhất chứa xâu s có độ dài không quá 10^5 ký tự.

Dữ liệu đầu ra: In ra các ký tự chưa xuất hiện trong xâu s, các ký tự được viết liền nhau theo thứ tự tăng dần.

Ví dụ 1:

Input

EFGHJABCD

Output

IKLMNOPQRSTUVWXYZ

Hướng dẫn giải:

Với mỗi kí tự trong xâu, mình sẽ đánh dấu là kí tự đó đã xuất hiện trong xâu.

Sau đó duyệt lần lượt các kí tự trong bảng chữ cái từ 'A' đến 'Z', nếu kí tự đang duyệt chưa được đánh dấu thì in ra kí tự đó.

Độ phức tạp: $O(n)$.

Solution C++:

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
#define FASTIO ios_base::sync_with_stdio(NULL); cin.tie(NULL);
   cout.tie(NULL);

const ll M = 1e9 + 7;
const ll N = 1e7;

string s;
ll f[300];

int main() {
    FASTIO;
    cin >> s;
```

```
    for(ll i = 0; i < s.size(); i++) {
        f[s[i]] = 1;
    }
    for(ll i = 'A'; i <= 'Z'; i++) {
        if(f[i] == 0) {
            cout << (char)i;
        }
    }
}
```

10.2 Bài 2: Số tròn chục

Đề bài:

Số tròn chục là số có chữ số hàng đơn vị là chữ số 0.

Yêu cầu: Cho hai số tự nhiên L và R. Hãy đếm xem có bao nhiêu số tròn chục lớn hơn L và nhỏ hơn R.

Dữ liệu đầu vào: Nhập vào số tự nhiên L, R ($L < R \leq 10^{12}$). Mỗi số trên một dòng.

Dữ liệu đầu ra: Ghi ra kết quả của bài toán.

Ví dụ:

Input

```
5
31
```

Output

```
3
```

Giải thích: Có 3 số tròn chục lớn hơn 5 và nhỏ hơn 31 là: 10, 20, 30

Hướng dẫn giải:

Số lượng số tròn chục nhỏ hơn hoặc bằng L là $\frac{L}{10}$.

Số lượng số tròn chục nhỏ hơn R là $\frac{R-1}{10}$.

Suy ra số lượng số tròn chục lớn hơn L và nhỏ hơn R là $\frac{R-1}{10} - \frac{L}{10}$.

Độ phức tạp: $O(1)$.

Solution C++:

```
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
#define FASTIO ios_base::sync_with_stdio(NULL); cin.tie(NULL);
   cout.tie(NULL);

const ll M = 1e9 + 7;
const ll N = 1e7;

ll l, r;

int main() {
```

```
    FASTIO;
    cin >> l;
    cin >> r;
    cout << (r - 1) / 10 - l / 10;
}
```

10.3 Bài 3: Tổng liên tiếp

Đề bài: Trong cuộc thi "Học sinh tài năng"được tổ chức tại một trường học, ban tổ chức chuẩn bị một bảng điểm điện tử để hiển thị điểm số của từng thí sinh. Điểm số của n thí sinh được hiển thị theo thứ tự từ thí sinh 1 đến thí sinh n, sau đó lặp lại vô hạn lần. Cụ thể, sau khi hiển thị điểm của thí sinh n, bảng điểm sẽ quay lại hiển thị điểm của thí sinh 1 rồi thí sinh 2, và cứ thế không ngừng.

Yêu cầu: Hãy giúp ban tổ chức tính tổng k điểm số liên tiếp xuất hiện trên bảng điểm, bắt đầu từ vị trí p.

Dữ liệu đầu vào: Gồm hai dòng:

- Dòng 1: Chứa ba số nguyên dương n, k, p lần lượt là số thí sinh trong cuộc thi, số lượng điểm số cần tính và vị trí bắt đầu tính điểm trên bảng điện tử.
- Dòng 2: Ghi n số nguyên dương $(a_1, a_2, ..., a_n)$ $(1 \leq a_i \leq 10^9)$.

Dữ liệu đầu ra: In ra một số nguyên duy nhất là kết quả của bài toán lấy dư cho $10^9 + 7$.

Ràng buộc dữ liệu:

- Subtask 1 (40%): $n \leq 10^3; p = 1; k \leq n$.
- Subtask 2 (30%): $n \leq 10^3; p, k \leq 10^6$.
- Subtask 3 (30%): $n \leq 10^6; p, k \leq 10^{18}$.

Ví dụ 1:

Input

```
6 7 3
4 3 6 2 9 5
```

Output

```
35
```

Giải thích: 7 số nguyên liên tiếp xuất hiện trên màn hình bắt đầu từ số xuất hiện thứ 3 là 6 2 9 5 4 3 6.
Kết quả: $(6 + 2 + 9 + 5 + 4 + 3 + 6) \ \% \ (10^9 + 7) = 35$

Hướng dẫn giải:

- **Kiến thức cần có:** Mảng cộng dồn.

- **Subtask 1:**

  Vì $k \leq n$ và $p = 1$ nên đoạn cần tính ở vị trí từ 1 đến n.

  Đáp án bài toán là $a_1 + a_2 + ... + a_k$.

  Vì thế nên minh chỉ cần duyệt từ vị trí 1 đến k và cộng các giá trị vào kết quả.

- **Subtask 2:**

Solution C++:

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
#define FASTIO ios_base::sync_with_stdio(NULL); cin.tie(NULL);
   cout.tie(NULL);

const ll M = 1e9 + 7;
const ll N = 1e7;

ll n, k, p, a[1000006], f[1000006];

void sub1() {
    ll res = 0;
    for(ll i = 1; i <= k; i++) res += a[i];
    cout << res;
}

void sub2() {
    ll d = 0, res = 0;
    ll vt = p;
    while(d != k) {
        if(vt > n) vt = 1;
        res += a[vt] % M;
        res %= M;
        vt++;
        d++;
    }
    cout << res;
}

void sub3() {
    if(k <= (n - p + 1)) {
        cout << f[p + k - 1] - f[p - 1];
        return;
    }
    ll dau = (f[n] - f[p - 1]) % M;
    ll giua = ((f[n] % M) * (((k - (n - p + 1)) / n) % M)) % M;
    ll cuoi = f[(k - (n - p + 1)) % n] % M;
    cout << (dau % M + giua % M + cuoi % M + M) % M;
}

```

```cpp
int main() {
    FASTIO;
    cin >> n >> k >> p;
    for(ll i = 1; i <= n; i++) {
        cin >> a[i];
        f[i] = f[i - 1] + a[i];
    }
    if(p % n == 0) p = n;
    else p = p % n;

    if(n <= 1000 && p == 1 && k <= n) {
        sub1();
    } else if(n <= 1000 && p <= 1e6 && k <= 1e6) {
        sub2();
    } else sub3();
}
```

10.4 Bài 4: Chiến Binh

Đề bài: Trong một vương quốc xa xưa, một vị tướng huyền thoại đang tập hợp một đội quân bất bại để chuẩn bị cho một cuộc chiến vĩ đại. Đội quân này có một cơ chế huấn luyện đặc biệt theo quy luật sau:

- Ngày đầu tiên (ngày thứ 0): Đội quân có n chiến binh ở cấp độ 1.
- Mỗi ngày tiếp theo:
  - Mỗi chiến binh cấp i sẽ huấn luyện và chiêu mộ thêm i tân binh (tất cả đều có cấp 1). Những tân binh này sẽ bắt đầu huấn luyện và chiêu mộ binh lính từ ngày sau.
  - Đồng thời, chiến binh cấp i sẽ trở nên mạnh hơn và thăng lên cấp $i + 1$.

Yêu cầu: Hãy xác định sau k ngày, tổng số chiến binh trong quân đội là bao nhiêu. Kết quả in ra là số nguyên duy nhất, lấy modulo $10^9 + 7$.

Dữ liệu đầu vào: Gồm hai số nguyên n, k ($1 \leq n \leq 10^3, 1 \leq k \leq 10^5$).

Dữ liệu đầu ra: Một số nguyên duy nhất là tổng số chiến binh sau k ngày, lấy modulo $10^9 + 7$.

Ràng buộc dữ liệu:

- Subtask 1 (40%): $n \leq 10^2; k \leq 10^3$.
- Subtask 2 (60%): $n \leq 10^3; k \leq 10^5$.

Ví dụ 1:

Input

```
5
4
```

Output

```
170
```

Giải thích: Với 5 chiến binh ban đầu, sau 4 ngày tổng số chiến binh có trong quân đội là 170.

Hướng dẫn giải:

- **Subtask 2:**

  Sau ngày 1, đội quân đã huấn luyện được thêm n

Solution C++:

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
#define FASTIO ios_base::sync_with_stdio(NULL); cin.tie(NULL);
   cout.tie(NULL);

const ll M = 1e9 + 7;
const ll N = 1e7;

ll n, k, lol, last;

int main() {
    FASTIO;
    cin >> n;
    cin >> k;
    lol = n;
    for(ll i = 1; i <= k; i++) {
        last += lol % M;
        lol += last % M;
    }
    cout << lol % M;
}
```

11 ĐỀ HSG 9 THCS TỈNH QUẢNG NINH 2024 - 2025

11.1 Bài 1: Oẳn tù tì

Đề bài: Trò chơi oẳn tù tì là một trò chơi dân gian phổ biến, thường chơi khi hai người đối diện nhau. Mỗi người sẽ đưa ra một trong ba hình dạng của bàn tay như sau:

- Búa - thể hiện bằng cách cả bàn tay nắm chặt lại;
- Kéo - thể hiện bằng cách ngón trỏ và ngón giữa tạo thành hình chữ V;
- Bao - thể hiện bằng cách cả bàn tay xòe ra.

Người chơi sẽ đồng loạt đưa ra một trong ba lựa chọn và so sánh với đối thủ. Nếu hai người chọn giống nhau thì hòa, còn nếu khác nhau thì sẽ có người thắng và người thua: Búa thắng Kéo (vì búa đập kéo), Kéo thắng Bao (vì kéo cắt bao), Bao thắng Búa (vì bao bọc búa).

Hai bạn An và Bình sẽ chơi n lần. Tuy nhiên, trước đó An đã tìm thấy một ghi chú cho biết Bình sẽ lựa chọn gì trong các lần chơi. Bạn hãy xác định số lần chơi tối đa mà An có thể thắng, **nếu anh ta không được phép đưa ra cùng một lựa chọn trong hai lần liên tiếp**.

Dữ liệu đầu vào:

- Dòng 1: Chứa một số nguyên n ($1 \leq n \leq 10^5$) là số lần chơi.
- Dòng 2: Chứa một xâu s độ dài n, trong đó kí tự (viết hoa) thứ i là 'R' hoặc 'S' hoặc 'P' biểu thị trong lần chơi thứ i, Bình ra Búa hoặc Kéo hoặc Bao tương ứng.

Dữ liệu đầu ra: Một số nguyên là số lần chơi tối đa mà An có thể thắng, nếu anh ta không được phép đưa ra cùng một lựa chọn trong hai lần liên tiếp.

Ràng buộc dữ liệu:

- Subtask 1 (30%): Xâu s không chứa hai kí tự liên tiếp giống nhau;
- Subtask 2 (30%): $1 \leq n \leq 15$;
- Subtask 3 (40%): Không có thêm ràng buộc nào.

Ví dụ 1:

Input

```
5
RPRRS
```

Output

```
4
```

Giải thích: Trong ví dụ trên, An có thể đưa ra PSPSR và anh ta thắng ở lần chơi thứ nhất, thứ hai, thứ ba, thứ năm và thua ở lần chơi thứ tư.

Hướng dẫn giải:

Solution C++:

11.2 Bài 2: Khung tranh

Đề bài: An có n thanh gỗ có chiều dài 1 và m thanh gỗ có chiều dài 2. Các thanh gỗ có thể được nối với nhau bằng cách xếp chúng thẳng hàng hoặc vuông góc.

An muốn lắp ráp một khung hình chữ nhật từ các thanh gỗ này, để sau đó anh có thể đặt một tờ giấy vào khung nafy và vẽ một phong cảnh thật đẹp tặng cho mẹ mình nhân dịp Ngày Quốc tế Phụ nữ.

Hơn nữa An nghĩ rằng diện tích khung hình chữ nhật càng lớn thì món quà sẽ càng có ý nghĩa. Vì vậy, điều quan trọng là anh ta phải xác định diện tích tối đa của khung hình chữ nhật có thể được ghép từ các thanh gỗ có sẵn.

Dữ liệu đầu vào:

- Dòng 1: Chứa số nguyên n ($0 \leq n \leq 10^9$) là số thanh gỗ có chiều dài 1.
- Dòng 2: Chứa số nguyên m ($0 \leq m \leq 10^9$) là số thanh gỗ có chiều dài 2.

Dữ liệu đầu ra: Một số nguyên là diện tích tối đa của khung hình chữ nhật có thể được tạo thành từ các thanh gỗ có sẵn.

Nếu không thể tạo thành bất kỳ khung hình chữ nhật nào từ các thanh gỗ có sẵn thì in ra số 0.

Ràng buộc dữ liệu:

- Subtask 1 (10%): $n = 0$ hoặc $m = 0$;
- Subtask 2 (20%): $n, m \leq 20$;
- Subtask 3 (20%): $n, m \leq 1000$;
- Subtask 4 (20%): $n, m \leq 5 \times 10^5$;
- Subtask 5 (30%): Không có thêm ràng buộc nào.

Ví dụ 1:

Input

```
5
0
```

Output

```
1
```

Giải thích: Có 5 thanh gỗ chiều dài 1. Từ chúng, An có thể tạo một hình vuông có cạnh 1, diện tích của nó là 1 và sẽ còn lại 1 thanh gỗ.

Ví dụ 2:

Input

```
4
3
```

Output

```
0
```

Giải thích: Có 4 thanh gỗ chiều dài 1 và 3 thanh gỗ chiều dài 2. Từ chúng, An có thể tạo thành một hình chữ nhật có kích thước 2×3.

Ví dụ 3:

Input

```
3
0
```

Output

```
0
```

Giải thích: có 3 thanh gỗ chiều dài bằng 1. Từ chúng, An không thể tạo thành hình chữ nhật.

Hướng dẫn giải:

Solution C++:

11.3 Bài 3: Dãy số bitonic

Đề bài: Dãy số $b_1, b_2, ..., b_k$ được gọi là dãy **bitonic** nếu $b_1 < b_2 < ... < b_i > ... > b_k$ với $1 \leq i \leq k$. Chú ý rằng nếu $i = 1$ thì $b_1 > b_2 > ... > b_k$ và dãy b là dãy giảm, còn nếu $i = k$ thì $b_1 < b_2 < ... < b_k$ và dãy b là dãy tăng, còn nếu $1 < i < k$ thì các phần tử từ b_1 đến b_i tăng dần và các phần tử từ b_i đến b_k giảm dần.

Ví dụ các dãy sau là dãy **bitonic**:

- 1;
- $1, 2, 3, 2$;
- $1, 4, 10$;
- $3, 2$.

và các dãy sau không là dãy **bitonic**:

- $1, 1$;
- $2, 1, 3$.

Cho dãy số $a_1, a_2, ..., a_n$. Hãy đếm số cặp (l, r) sao cho $1 \leq l \leq r \leq n$ và dãy $a_l, a_{l+1}, ..., a_r$ là dãy **bitonic**.

Dữ liệu đầu vào:

- Dòng 1: Chứa số nguyên n $(1 \leq n \leq 3 \times 10^5)$.
- Dòng 2: Chứa n số nguyên $a_1, a_2, ..., a_n$ $(1 \leq a_i \leq n)$.

Dữ liệu đầu ra: Một số nguyên là số cặp (l, r) sao cho $1 \leq l \leq r \leq n$ và dãy $a_l, a_{l+1}, ..., a_r$ là dãy **bitonic**.

Ràng buộc dữ liệu:

- Subtask 1 (40%): $n \leq 500$;
- Subtask 2 (30%): $n \leq 5000$;
- Subtask 3 (30%): Không có thêm ràng buộc nào.

Ví dụ 1:

Input

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

Output

```
11
```

Giải thích: Trong ví dụ đầu tiên, các cặp sau là thỏa mãn:

1. $(1, 1)$ ứng với dãy 1;
2. $(2, 2)$ ứng với dãy 2;

3. $(2, 3)$ ứng với dãy $1, 2$;
4. $(2, 4)$ ứng với dãy $1, 2, 3$;
5. $(2, 5)$ ứng với dãy $1, 2, 3, 1$;
6. $(3, 3)$ ứng với dãy 2;
7. $(3, 4)$ ứng với dãy $2, 3$;
8. $(3, 5)$ ứng với dãy $2, 3, 1$;
9. $(4, 4)$ ứng với dãy 3;
10. $(4, 5)$ ứng với dãy $3, 1$;
11. $(5, 5)$ ứng với dãy 1.

Ví dụ 2:

Input

```
3
1 1 1
```

Output

```
3
```

Hướng dẫn giải:
Solution C++:

11.4 Bài 4: Sửa đường

Đề bài: Ở thành phố của An mọi thứ đều tốt, ngoại trừ một con đường. Con đường này có n cái hố xếp theo một hàng. Chúng ta đánh số các hố này từ 1 đến n theo thứ tự từ đầu đến cuối con đường.

An thực sự muốn giúp thành phố của mình. Vì vậy, anh ta muốn sửa chữa ít nhất k cái hố (có thể anh ta sửa chữa nhiều hơn) trên con đường này.

Thành phố có m công ty sửa đường, công ty thứ i cần c_i đơn vị tiền để sửa chữa một đoạn đường có chứa các hố với chỉ số nhỏ nhất là l_i và lớn nhất là r_i. Các công ty này rất tham lam, vì vậy nếu họ sửa chữa một đoạn đường có chứa một số hố đã được sửa, họ không giảm giá sửa chữa đoạn đường này.

Hãy xác định số tiền tối thiểu mà An sẽ cần để sửa chữa ít nhất k cái hố.

Yêu cầu: Hãy lập trình xác định chênh lệch điểm nhỏ nhất của hai bạn mà thầy Sơn chọn.

Dữ liệu đầu vào:

- Dòng 1: Chứa 3 số nguyên n, m, k $(1 \leq n \leq 300; 1 \leq m \leq 10^5; 1 \leq k \leq n)$.
- Dòng i trong m dòng tiếp theo: Chứa 3 số nguyên l_i, r_i, c_i $(1 \leq l_i \leq r_i \leq n; 1 \leq c_i \leq 10^9)$ mô tả công ty thứ i cần c_i đơn vị tiền để sửa chữa đoạn đường từ hố l_i đến r_i.

Dữ liệu đầu ra: Một số nguyên là số tiền tối thiểu mà An cần để sửa chữa ít nhất k cái hố. Trong trường hợp không thể sửa ít nhất k cái hố thì ghi ra số -1.

Ràng buộc dữ liệu:

- Subtask 1 (10%): $k = 1$;
- Subtask 2 (15%): $1 \leq l_i = r_i \leq n$ với mọi $i = 1, 2, ..., m$;
- Subtask 3 (20%): $1 \leq m \leq 20$;
- Subtask 4 (25%): $1 \leq n \leq 100$ và $1 \leq m \leq 1000$;
- Subtask 5 (30%): Không có thêm ràng buộc nào.

Ví dụ 1:

Input

```
10 4 6
7 9 11
6 9 13
7 7 7
3 5 6
```

Output

```
17
```

Giải thích: Phương án tối ưu là sử dụng công ty thức nhất và thứ tư để sửa đường. Tổng cộng có 6 hố được sửa chữa là $3, 4, 5, 7, 8, 9$ với tổng chi phí là $11 + 6 = 17$.

Ví dụ 2:

Input

```
10 7 1
3 4 15
8 9 8
5 6 8
9 10 6
1 4 2
1 4 10
8 10 13
```

Output

```
2
```

Giải thích: Phương án tối ưu là sử dụng công ty thứ năm sửa đường và có 4 hố được sửa chữa là $1, 2, 3, 4$ (thoả mãn tối thiểu 1 hố được sửa chữa) với chi phí là 2.

Ví dụ 3:

Input

10 1 9
5 10 14

Output

-1

Giải thích: Chỉ có duy nhất một công ty sửa đường và sửa được 6 hố. Vì vậy không có phương án nào để sửa được ít nhất 9 hố.

Hướng dẫn giải:

Solution C++:

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
int main() {
    int n;
    ll S;
    cin >> n >> S;
    vector<ll> a(n+1, 0), prefix(n+1, 0);
    for (int i = 1; i <= n; ++i) {
        cin >> a[i];
        prefix[i] = prefix[i-1] + a[i];
    }
    ll res = 0;
    for (int l = 1; l <= n; ++l) {
        for (int r = l; r <= n; ++r) {
            ll sum = prefix[r] - prefix[l-1];
            if (abs(sum) > S) res++;
        }
    }
    cout << res << endl;
    return 0;
}
```

12 ĐỀ HSG 9 THCS TỈNH BẮC NINH 2024 - 2025

12.1 Bài 1: Bánh kem

Đề bài: Nhân dịp Giáng sinh, An muốn tự làm một chiếc bánh kem thật đẹp để trang trí cho buổi lễ. An muốn để chiếc bánh kem trên mặt bàn hình tròn ở phòng khách. Tuy nhiên nhà An chỉ có khuôn làm được chiếc bánh kem hình chữ nhật có kích thước $a \times b$, An băn khoăn rằng với chiếc bánh kem kích thước $a \times b$ có thể đặt lên mặt bàn hình tròn bán kính r mà không có phần nào của bánh bị tràn ra ngoài biên của mặt bàn hay không.

Yêu cầu: Với 3 số r, a, b tương ứng là bán kính của mặt bàn và kích thước chiếc bánh kem. Bạn hãy cho biết có thể đặt chiếc bánh kem lên mặt bàn kính tròn hay không?

Dữ liệu đầu vào: Gồm 3 số r, a, b $(r, a, b \leq 10^5)$ các số cách nhau 1 dấu cách tương ứng là bán kính của mặt bàn, kích thước của chiếc bánh.

Dữ liệu đầu ra: Ghi **YES/NO** tương ứng với **CÓ/KHÔNG**.
Ví dụ 1:

Input

3 3 4

Output

YES

Ví dụ 2:

Input

1 3 4

Output

NO

Hướng dẫn giải:
Solution C++:

12.2 Bài 2: Số nguyên tố

Đề bài: Trong giờ Tin học của lớp 9A thầy Minh có đưa ra một trò chơi trúng thưởng như sau:

Thầy viết lên bảng một xâu S, bạn nào tìm ra số nguyên tố lớn nhất có trong xâu S sẽ nhận được phần thưởng.

Yêu cầu: Xâu S gồm n ký tự chỉ chứa chữ cái và chữ số, học sinh cần chọn 1 đoạn con liên tiếp chỉ gồm các chữ số để tạo thành 1 số không quá 5×10^6 và là số nguyên tố. Học sinh chọn được số nguyên tố lớn nhất sẽ được tặng thưởng số tiền bằng đúng số được chọn. Bạn hãy giúp các học sinh chọn ra số nguyên tố lớn nhất để nhận được nhiều tiền thưởng nhất có thể.

Ví dụ: S = 'cd0056aB45k1250cd19hk23' ta có thể tạo ra các số như: $0, 00, 056, 5, 56, 6, 4, 45, 5$..... *tuy nhiên chỉ có các số* $2, 3, 5, 19, 23$ *là các số nguyên tố được tạo ra là* 23.

Dữ liệu đầu vào: Chứa một xâu S có độ dài không quá 5×10^6 ký tự.

Dữ liệu đầu ra: Một số nguyên là số tiền mà học sinh nhận được. Nếu không có số nguyên tố nào thì số tiền nhận được là 0.

Ràng buộc dữ liệu:

- Subtask 1 (75%): $1 \leq N \leq 255$, số nguyên tố lớn nhất tạo được $\leq 10^6$;
- Subtask 2 (25%): $N \leq 5 \times 10^6$, số nguyên tố lớn nhất tạo được $\leq 5 \times 10^6$.

Ví dụ 1:

Input

cd0056aB45k1250cd19hk23

Output

23

Ví dụ 2:

Input

cA12cg42m

Output

0

Hướng dẫn giải:
Solution C++:

12.3 Bài 3: Điểm số

Đề bài: Ngày hội đọc sách được tổ chức định kỳ tại trường THCS A. Mỗi quyển sách trong thư viện trường có một "điểm số"đại diện cho độ phổ biến của nó. Có n quyển sách trong thư viện được đánh số thứ tự từ 1 đến n tương ứng với điểm số là các số nguyên $A_1, A_2, ..., A_n$.

Một đoạn con *[h; r]* là một dãy các điểm số liên tiếp $A_h, A_{h+1}, ..., A_r$ $(1 \leq h \leq r \leq n)$. Đoạn *[h; r]* được gọi là một đoạn điểm số đặc biệt nếu $A_h = A_r$ và tổng các điểm số của đoạn này là lớn nhất.

Yêu cầu: Hãy đưa ra tổng của đoạn điểm số đặc biệt.
Dữ liệu đầu vào:

- Dòng 1: Ghi số nguyên dương n là số lượng quyển sách.
- Dòng 2: Ghi n số nguyên $A_1, A_2, ..., A_n$ $(|A_i| \leq 10^3, 1 \leq i \leq n \leq 5 \times 10^5)$, mỗi số cách nhau bởi một khoảng trắng.

Dữ liệu đầu ra: Kết quả theo yêu cầu của bài toán.
Ràng buộc dữ liệu:

- Subtask 1 (30%) $1 \leq n \leq 10^2$;
- Subtask 2 (40%): $n \leq 5 \times 10^5; 0 < A_i \leq 10^3; \forall i \in [1, n]$;
- Subtask 3 (30%): Không có thêm ràng buộc nào.

Ví dụ 1:

Input

8
5 3 10 3 2 -1 2 9

Output

16

Ví dụ 2:

Input

```
6
5 20 6 1 2 6
```

Output

```
20
```

Hướng dẫn giải:
Solution C++:

12.4 Bài 4: Số đặc biệt

Đề bài: Hôm nya cô giáo B dạy học sinh về dãy con tăng dài nhất. Dãy con tăng dài nhất là dãy con nhận được từ dãy ban đầu bằng cách xoá đi một số số, giữ nguyên thứ tự ban đầu, sao cho dãy còn lại thoả mãn tính chất hai số cạnh nhau thì số đứng trước nhỏ hơn số đứng sau.

Sau khi dạy xong, cô giáo B thấy rằng một số số xuất hiện trong nhiều dãy con tăng dài nhất, cô gọi đó là số đặc biệt.

Cho dãy số nguyên $a_1, a_2, ..., a_n$ khác nhau từng đôi một ($n \leq 10^5, 1 \leq a_i \leq n$). Số a_i được gọi là một số đặc biệt đối với dãy số trên nếu như a_i thuộc ít nhất một dãy con tăng dài nhất của A.

Yêu cầu: Tìm các số đặc biệt của dãy A.

Dữ liệu đầu vào:

- Dòng 1: Ghi T ($1 \leq T \leq 10$) là số bộ test.
- T nhóm dòng tiếp theo, mỗi nhóm gồm hai dòng:
  - Dòng 1: Số n.
  - Dòng 2: n số nguyên có thứ tự từ 1 đến n.

Dữ liệu đầu ra: Gồm T dòng, mỗi dòng ghi các số đặc biệt của bộ test tương ứng theo giá trị tăng dần.

Ràng buộc dữ liệu:

- Subtask 1 (60%): $n \leq 10^2$;
- Subtask 2 (40%): $n \leq 10^5$.

Ví dụ 1:

Input

```
2
7
1 2 3 7 4 5 6
5
1 4 3 2 5
```

Output

```
1 2 3 4 5 6
1 2 3 4 5
```

Hướng dẫn giải:
Solution C++:

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
int main() {
    int n;
    ll S;
    cin >> n >> S;
    vector<ll> a(n+1, 0), prefix(n+1, 0);
    for (int i = 1; i <= n; ++i) {
        cin >> a[i];
        prefix[i] = prefix[i-1] + a[i];
    }
    ll res = 0;
    for (int l = 1; l <= n; ++l) {
        for (int r = l; r <= n; ++r) {
            ll sum = prefix[r] - prefix[l-1];
            if (abs(sum) > S) res++;
        }
    }
    cout << res << endl;
    return 0;
}
```

13 ĐỀ HSG 9 THCS TỈNH NINH BÌNH 2024 - 2025

Thời gian làm bài: 150 phút Độ khó:

13.1 Bài 1: Đếm số

Đề bài: Cho 4 số nguyên dương $a, b, x, y (2 \leq a, b \leq 10^9; 2 \leq x \leq y \leq 10^{12})$.
Yêu cầu:
Đếm số lượng số nguyên thuộc đoạn $[x, y]$ chia hết cho a nhưng không chia hết cho b.
Dữ liệu đầu vào:
Gồm một dòng chứa 4 số nguyên dương a, b, x, y. Các số cách nhau bởi một dấu cách.
Dữ liệu đầu ra:
Gồm một dòng chứa một số nguyên là kết quả của bài toán.
Ràng buộc dữ liệu:

- Subtask 1 (30%): $2 \leq x \leq y < 10^6$;
- Subtask 2 (40%): $10^6 \leq x \leq y \leq 10^9$;

- Subtask 3 (30%): Không có thêm ràng buộc nào.

Ví dụ 1:

Input

4 10 24 44

Output

5

Giải thích:
Trong đoạn $[24, 44]$ có 5 số nguyên thoa mãn điều kiện chia hết cho 4 nhưng không chia hết cho 10 là $24, 28, 32, 36, 44$.

Ví dụ 2:

Input

8 4 14 20

Output

0

Giải thích: Trong đoạn $[14, 20]$ không có số nguyên nào thoa mãn điều kiện chia hết cho 8 nhưng không chia hết cho 4.

Hướng dẫn giải:

Solution C++:

13.2 Bài 2: Cặp số

Đề bài: Cho dãy số gồm n số nguyên $a_1, a_2, ..., a_n$ và một số nguyên dương k. Số nguyên a_i, a_j là số nguyên lần lượt ở các vị trí thứ i và thứ j.

Yêu cầu:
Hãy cho biết có bao nhiêu cách chọn các cặp số i và j thỏa mãn: $i < j$ và $a_i + a_j$ chia hết cho k.

Dữ liệu đầu vào: Gồm hai dòng:

- Dòng 1: Chứa hai số nguyên dương n và $k (1 < n, k < 10^6)$ mỗi số cách nhau một khoảng trắng.
- Dòng 2: Chứa n số nguyên $a_1, a_2, ..., a_n (|a_i| < 10^9; 1 \leq i \leq n)$, mỗi số cách nhau một khoảng trắng.

Dữ liệu đầu ra: Gồm một dòng chứa một số nguyên là kết quả của bài toán.

Ràng buộc dữ liệu:

- Subtask 1 (60%): $n < 10^3$;
- Subtask 2 (40%): Không có thêm ràng buộc nào.

Ví dụ 1:

Input

```
4 6
2 4 8 -8
```

Output

```
4
```

Giải thích: Trong ví dụ trên, có 4 cặp số thỏa mãn là: $(1, 2), (1, 4), (2, 3), (3, 4)$.
Hướng dẫn giải:
Solution C++:

13.3 Bài 3: Xâu nguyên tố cùng nhau

Đề bài: Cho xâu S chỉ có các kí tự chữ cái và kí tự chữ số có độ dài không vượt quá 10^6 kí tự. Các số trong xâu S là một dãy các kí tự chữ số liên tiếp được phân tách bởi các kí tự chữ cái, xâu S bắt đầu bằng một kí tự chữ cái và kết thúc cũng bằng một kí tự chữ cái. Khi thực hiện lấy ra các số trong S, ta thu được một dãy số A gồm n số nguyên dương $a_1, a_2, ..., a_n (0 < a_i \leq 10^6)$.

Hai số nguyên dương x, y được gọi là nguyên tố cùng nhau nếu ước chung lớn nhất của chúng bằng 1. Một đoạn con liên tiếp trong dãy A được gọi là nguyên tố cùng nhau nếu mọi cặp số trong đoạn đó là nguyên tố cùng nhau.

Yêu cầu: Tìm đoạn con liên tiếp nguyên tố cùng nhau dài nhất.

Dữ liệu đầu vào: Gồm một dòng chứa xâu $S (|S| \leq 10^6)$.

Dữ liệu đầu ra: Gồm một dòng chứa một số duy nhất là độ dài của đoạn con liên tiếp nguyên tố cùng nhau dài nhất của dãy A.

Ràng buộc dữ liệu:

- Subtask 1 (30%): $n \leq 20$;
- Subtask 2 (40%): Các số nguyên xuất hiện trong xâu S đều là số nguyên tố
- Subtask 3 (30%): Không có thêm ràng buộc nào.

Ví dụ 1:

Input

```
a14a5ac7a6bb
```

Output

```
3
```

Giải thích: Trong ví dụ trên, ta có dãy số $A = [14, 5, 7, 6]$. Đoạn con liên tiếp nguyên tố cùng nhau dài nhất là $[5, 7, 6]$ có độ dài bằng 3.

Ví dụ 2:

Input

ac5b2c3a7b

Output

4

Giải thích: Trong ví dụ trên, ta có dãy số $A = [5, 2, 3, 7]$. Đoạn con liên tiếp nguyên tố cùng nhau dài nhất là $[5, 2, 3, 7]$ có độ dài bằng 4.

Hướng dẫn giải:

Solution C++:

13.4 Bài 4: Dãy số

Đề bài: Cho dãy số $a_1, a_2, ..., a_n$ các số $a_i (1 \leq i \leq n)$ không quá m và có giá trị đôi một khác nhau, trong đó có đúng một số có giá trị bằng 0.

Yêu cầu: Thay thế số 0 thành một giá trị bất kì không được trùng với các giá trị đã có để nhận được một dãy con có các giá trị liên tiếp dài nhất có thể.

Dữ liệu đầu vào: Gồm hai dòng:

- Dòng 1: Chứa hai số nguyên dương $n, m (1 \leq n < m \leq 10^6)$.
- Dòng 2: Chứa n số nguyên $a_1, a_2, ..., a_n (0 \leq a_i < m; 1 \leq i \leq n)$, mỗi số cách nhau một khoảng trắng.

Dữ liệu đầu ra: Gồm một dòng chứa một số nguyên là kết quả của bài toán.

Ràng buộc dữ liệu:

- Subtask 1 (40%): $1 \leq n \leq 100$;
- Subtask 2 (30%): $100 < n \leq 1000$;
- Subtask 3 (30%): $1000 < n < 10^6$

Ví dụ:

Input

8 5
8 2 0 5 7

Output

4

Giải thích: Trong ví dụ trên, ta có thể thay số 0 thành số 6 để nhận được dãy con có các giá trị liên tiếp dài nhất là $5, 6, 7, 8$ có độ dài bằng 4.

Hướng dẫn giải:

Solution C++:

14 ĐỀ HSG 9 THCS TP CẦN THƠ 2024 -2025

Thời gian làm bài: 150 phút Độ khó:

14.1 Bài 1: Luận văn

Đề bài: Trong một đợt tổng kết khóa học, trường có n bài luận văn để chấm. Luận văn thứ 1 có số lỗi là a_i, trường chọn ra đúng k luận văn có số lỗi nhỏ nhất trong số n luận văn trên để khen thưởng.

Yêu cầu: Hãy lập trình xác định tổng số lỗi nhỏ nhất của k luận văn được chọn.

Dữ liệu đầu vào: Gồm hai dòng:

- Dòng 1: Chứa hai số nguyên dương $n, k (1 \leq k \leq n \leq 10^6)$.
- Dòng 2: Chứa n số nguyên dương $a_1, a_2, ..., a_n (1 \leq a_i \leq 10^4; 1 \leq i \leq n)$, mỗi số cách nhau một khoảng trắng.

Dữ liệu đầu ra: Gồm một dòng chứa một số nguyên là kết quả của bài toán.

Ràng buộc dữ liệu: Bài này không có chia Subtask.

Ví dụ:

Input

```
5 3
1 2 3 1 4
```

Output

```
4
```

Giải thích: Có 5 luận văn, số lỗi lần lượt theo thứ tự là $(1, 2, 3, 1, 4)$. Chọn 3 luận văn $(1 + 2 + 1 = 4)$.

Hướng dẫn giải:

Solution C++:

14.2 Bài 2: Vị trí

Đề bài: Một xâu được gọi là xâu chuẩn hóa nếu nó tuân theo quy tắc sau: ký tự đầu tiên được viết in hoa, các ký tự còn lại là hai ký tự thường và giữa các ký tự không có khoảng trắng.

Cho một danh sách gồm n dòng, mỗi dòng chứa một xâu và mỗi câu chỉ chứa các ký tự chữ cái tiếng Amh, không có ký tự đặc biệt.

Yêu cầu: Hãy lập trình xác định vị trí các xâu chưa chuẩn hóa trong danh sách.

Dữ liệu đầu vào: Gồm $n + 1$ dòng:

- Dòng 1: Chứa số nguyên dương $n (1 \leq n \leq 10^3)$.
- n dòng tiếp theo: mỗi dòng ghi một xâu (độ dài của xâu không vượt quá 10^2 ký tự).

Dữ liệu đầu ra: Gồm một dòng chứa các vị trí của các xâu chưa chuẩn hóa trong danh sách, các vị trí cách nhau một dấu cách. Nếu không tìm được kết quả thì chỉ ghi duy nhất số 0.

Ràng buộc dữ liệu: Bài này không có chia Subtask.

Ví dụ 1:

Input

```
5
Anh
NgOc
Phuong
tRan
TunG
```

Output

```
2 4 5
```

Giải thích: Các xâu chưa chuẩn hóa là: "NgOc", "tRan", "Tun"ở vị trí $2, 4, 5$.

Ví dụ 2:

Input

```
3
Binh
Toan
Sang
```

Output

```
0
```

Giải thích: Tất cả các xâu đều chuẩn hóa.

Hướng dẫn giải:

Solution C++:

14.3 Bài 3: Gặp nhau

Đề bài: Một công ty du lịch A có hai xe vận chuyển khách hàng đi đến các điểm tham quan trong khu vực. Công ty đã sắp xếp lịch trình sao cho mỗi xe luôn xuất phát đúng theo lịch trình. Xe thứ nhất cứ x giờ thì về bến công ty, xe thứ hai cứ y giờ thì về bến công ty.

Yêu cầu: Hãy lập trình cho biết số lần gặp nhau của hai xe tại công ty sau n giờ.

Dữ liệu đầu vào: Gồm hai dòng:

- Dòng 1: Chứa số nguyên dương $n (10 \leq n \leq 10^6)$.
- Dòng 2: Chứa hai số nguyên dương $x, y (1 \leq x \leq y \leq 10^2)$ cách nhau một ký tự trắng.

Dữ liệu đầu ra: Gồm một dòng chứa một số nguyên là kết quả của bài toán.
Ràng buộc dữ liệu:

- Subtask 1 (60%): $10 \leq n \leq 10^3$;
- Subtask 2 (40%): $10^3 < n \leq 10^6$.

Ví dụ:

Input

```
40
6 4
```

Output

```
3
```

Giải thích: Hai xe xuất phát cùng thời điểm. Xe thứ nhất cứ 6 giờ ghé vào công ty rồi đi, xe thứ hai cứ 4 giờ ghé vào công ty rồi đi.
Trong 40 giờ, hai xe gặp nhau 3 lần tại công ty sau khi xuất phát.
Hướng dẫn giải:
Solution C++:

14.4 Bài 4: Đội tuyển

Đề bài: Một trường học tổ chức kỳ thi học sinh giỏi môn tin học vào đội tuyển trường. Cuộc thi có nhiều vòng thi,, mỗi vòng thi được chấm theo một thang điểm. Giáo viên A có một danh sách điểm số của học sinh và muốn xác định có bao nhiêu học sinh được vào đội tuyển trường. Một học sinh được vào đội tuyển trường nếu tổng điểm của học sinh đó không nhỏ hơn 75% của bạn có tổng điểm cao nhất trong danh sách.
Yêu cầu: Cho trước tổng điểm của từng học sinh từ các vòng thi. Hãy tính xem số bạn học sinh được vào đội tuyển trường.
Dữ liệu đầu vào: Gồm hai dòng:

- Dòng 1: Chứa số nguyên dương $n(1 \leq n \leq 10^3)$.
- Dòng 2: Chứa n số nguyên dương $a_1, a_2, ..., a_n(1 \leq a_i \leq 10^2; 1 \leq i \leq n)$, mỗi số cách nhau một khoảng trắng là tổng điểm số các vòng thi của từng thí sinh.

Dữ liệu đầu ra: Gồm một dòng chứa một số nguyên là số lượng học sinh được vào đội tuyển trường.
Ràng buộc dữ liệu: Bài này không có chia Subtask.
Ví dụ:

Input

```
4
40 45 15 45
```

Output

```
3
```

Giải thích: Có 4 học sinh có điểm lần lượt là $40, 45, 15, 45$. Trong đó điểm cao nhất của thí sinh đạt được là 45. Có 3 học sinh được vào đội tuyển trường với số điểm lần lượt là $(40, 45, 45)$.

Hướng dẫn giải:

Solution C++:

14.5 Bài 5: Đơn hàng

Đề bài: Một công ty công nghệ chuyên giao hàng theo đơn bằng đội nhân viên của công ty. Trong công ty này có n nhân viên. Nhân viên thứ i có thời gian hoàn thành một đơn hàng trong a_i giờ. Các nhân viên này giao hàng một cách độc lập.

Yêu cầu: Hãy lập trình xác định thời gian nhỏ nhất công ty hoàn thành giao được đơn hàng.

Dữ liệu đầu vào: Gồm hai dòng:

- Dòng 1: Chứa hai số nguyên dương $n, k (1 \leq n, k \leq 10^5)$ cách nhau một ký tự trắng, là số lượng nhân viên và số lượng đơn hàng cần hoàn thành.
- Dòng 2: Chứa n số nguyên dương $a_1, a_2, ..., a_n (1 \leq a_i \leq 10^3; 1 \leq i \leq n)$ là thời gian hoàn thành một đơn hàng của nhân viên thứ i, các số kề nhau cách nhau một ký tự trắng.

Dữ liệu đầu ra: Gồm một dòng chứa một số nguyên là kết quả của bài toán.

Ràng buộc dữ liệu:

- Subtask 1 (60%): $1 \leq n, k \leq 10^2$;
- Subtask 2 (40%): $1 \leq n, k \leq 10^5$.

Ví dụ 1:

Input

```
4 7
1 4 2 5
```

Output

```
4
```

Giải thích: Có 4 nhân viên thời gian hoàn thành của mỗi nhân viên lần lượt là $1, 4, 2, 5$ giờ. Trong 4 giờ: Nhân viên 1 giao 4 đơn, nhân viên 2 giao 1 đơn, nhân viên 3 giao 2 đơn. Tổng cộng 7 đơn hàng được giao.

Ví dụ 2:

Input

```
5 12
2 4 4 4 5
```

Output

```
10
```

Giải thích: Có 5 nhân viên thời gian hoàn thành của mỗi nhân viên lần lượt là $2, 4, 4, 4, 5$ giờ. Trong 10 giờ: Nhân viên 1 giao 5 đơn, nhân viên 2 giao 2 đơn, nhân viên 3 giao 2 đơn, nhân viên 4 giao 2 đơn, nhân viên 5 giao 1 đơn. Tổng cộng 12 đơn hàng được giao.

Hướng dẫn giải:

Solution C++:

15 ĐỀ HSG 9 THCS TP. HÀ NỘI 2024 - 2025

Thời gian làm bài: 150 phút Độ khó:

15.1 Bài 1: Cắt hình

Đề bài: Cho một tờ giấy hình chữ nhật kích thước $M(cm) \times N(cm)$ và một số tự nhiên K.

Yêu cầu: Nếu cắt những hình vuông có kích thước $K(cm) \times K(cm)$ từ tờ giấy này thì diện tích còn lại nhỏ nhất là bao nhiêu cm^2.

Dữ liệu đầu vào: Gồm một dòng chứa ba số nguyên dương $M, N, K (1 \leq M, N, K \leq 10^9)$ cách nhau một khoảng trắng.

Dữ liệu đầu ra: Gồm một dòng chứa một số nguyên là kết quả của bài toán.

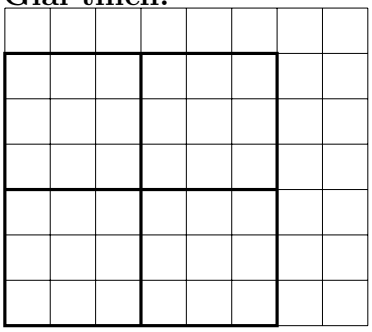
Ràng buộc dữ liệu: Bài này không có chia Subtask.

Ví dụ:

Input
8 7 3

Output
20

Giải thích:



Hướng dẫn giải:

Solution C++:

15.2 Bài 2: Mạch DNA

Đề bài: Cho mạch mã gốc DNA bốn loại nucleotide A, T, G, C. Để tiết kiệm bộ nhớ, mạch mã gốc đã được nén lại thành một chuỗi S gồm các cặp là **số lần xuất hiện liên tiếp nucleotide** và loại nucleotide tương ứng.

Ví dụ: Mạch mã gốc $AAACAATGGGGA$ nén thành chuỗi $3A1C2A4G1A$.

Các nucleotide ở hai mạch của phân tử DNA liên kết với nhau theo nguyên tắc bổ sung, trong đó A liên kết với T, G liên kết với C. Do vậy, nếu biết trình tự nucleotide trên một mạch có thể suy ra trình tự của mạch còn lại.

Ví dụ: Một đoạn phân tử DNA ở sinh vật nhân thực có trình tự nucleotide trên mạch mã gốc là $AAACAATGGGGA$. Trình tự nucleotide trên mạch bổ sung của đoạn DNA này là : $TTTGTTACCCCT$.

Yêu cầu: Cho một chuỗi ký tự S mô tả mạch mã gốc DNA sau khi đã nén. Hãy lập trình xác định mạch bổ sung của mạch mã gốc sau khi giải nén.

Dữ liệu đầu vào: Gồm một dòng chứa chuỗi ký tự S có độ dài không vượt quá 1000. Dữu liệu đảm bảo chuỗi sau khi giả nén có độ dài không vượt quá 10^5.

Dữ liệu đầu ra: Chuỗi ký tự là mạch bổ sung của mạch mã gốc sau khi giải nén.

Ràng buộc dữ liệu:

- Subtask 1 (20%): Độ dài chuỗi S là 2, trong đó ký tự đầu tiên là chữ số, ký tự thứ hai là một trong 4 chữ cái A, T, G, C.
- Subtask 2 (20%): Có duy nhất một loại nucleotide.
- Subtask 3 (40%): Số lần xuất hiện liên tiếp nucleotide A, T, G, C nhỏ hơn 10.
- Subtask 4 (20%): Không có thêm ràng buộc nào.

Ví dụ:

Input

```
5A2G1A11T1C
```

Output

```
TTTTTCCTAAAAAAAAAAAG
```

Giải thích: Mạch mã gốc sau khi giải nén là $AAAAAGGATTTTTTTTTTC$. Mạch bổ sung của mạch mã gốc này là $TTTTTCCTAAAAAAAAAAAG$.

Hướng dẫn giải:

Solution C++:

15.3 Bài 3: Dãy đèn

Đề bài: Để trang trí Tết, Nam treo một dây đèn gồm N bóng đèn, được đánh số từ 1 đến N, từ trái sang phải. Mỗi bóng đèn khi bật sẽ có hai màu vàng hoặc đỏ. Dây đèn được nhúng một mã lệnh cho phép nhận một số tự nhiên X. Khi đó, màu của bóng đèn thứ X và các bóng đèn cách bóng đèn thứ X không quá K bóng đèn sẽ đều đổi từ vàng thành đỏ và ngược lại.

Ban đầu các bóng đèn đều có màu vàng. Để dayay đèn trông đẹp nhất, Nam đã lập trình để điều khiển màu của các bóng đèn. Chương trình của Nam có M dòng lệnh, mỗi dòng lệnh tương ứng với một lần gọi mã lệnh của dãy đèn. Vì số lượng bóng đèn quá lớn, sau khi lập trình xong, Nam muốn kiểm tra ngẫu nhiên màu của một số bóng đèn xem có đúng như ý tưởng ban đầu không.

Yêu cầu: Cho các số tự nhiên X là tham số của M dòng lệnh trong chương trình của Nam. Hãy lập trình để trả lời Q câu hỏi tương ứng với các lần kiểm tra của Nam. Biết

rằng mỗi câu hỏi chứa một số nguyên dương P để xác định xem bóng đèn thứ P trong dãy đèn có màu vàng hay đỏ.

Dữ liệu đầu vào: Gồm ba dòng:

- Dòng 1: Chứa bốn số nguyên dương $N, K, M, Q (1 \leq N \leq 10^9; 1 \leq M \leq 10^5; 1 \leq Q \leq 10^5; 0 \leq K \leq N)$ cách nhau một ký tự trắng.
- Dòng 2: Chứa M số nguyên dương $X_1, X_2, ..., X_M (1 \leq X_i \leq N; 1 \leq i \leq M)$ cách nhau một ký tự trắng.
- Dòng 3: Chứa Q số nguyên dương $P_1, P_2, ..., P_Q (1 \leq P_i \leq N; 1 \leq i \leq Q)$ cách nhau một ký tự trắng.

Dữ liệu đầu ra: Gồm Q dòng, dòng thứ i trả lời câu hỏi thứ i. Nếu bóng đèn tại vị trí P_i đang có màu vàng thì ghi ra ký tự **"V"**, ngược lại ghi ra ký tự **"D"**.

Ràng buộc dữ liệu:

- Subtask 1 (60%): $1 \leq N, M, Q \leq 10^3$;
- Subtask 2 (20%): $N, M \leq 10^5$;
- Subtask 3 (20%): Không có thêm ràng buộc nào.

Ví dụ:

Input

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

Output

```
D
V
V
D
```

Giải thích:

- Sau lần gọi mã lệnh thứ nhất, các bóng trong dãy đèn có màu là: V, D, D, D, V, V, V;
- Sau lần gọi mã lệnh thứ hai, các bóng trong dãy đèn có màu là: V, D, D, V, D, D, V;
- Kết quả trả lời các câu hỏi lần lượt là: D, V, V, D.

Hướng dẫn giải:

Solution C++:

15.4 Bài 4: Trò chơi

Đề bài: Cho một bảng hình vuông kích thước là $N \times N, N$ là số lẻ. Các hàng của bảng được đánh số từ 1 tới N, từ trên xuống dưới, các cột của bảng được đánh số từ 1 tới N, từ trái sang phải. Ban đầu, các số từ 1 đến N^2 được ghi vào bảng này lần lượt từ trái sang phải, từ trên xuống dưới. Khi $N = 5$ thì bảng vuông sẽ có dạng như hình sau:

	1	**2**	**3**	**4**	**5**
1	1	2	3	4	5
2	6	7	8	9	10
3	11	12	13	14	15
4	16	17	18	19	20
5	21	22	23	24	25

Luật chơi: Có Q luật chơi, mỗi lượt chơi quản trò sẽ cấp cho người chơi thông tin là ba số nguyên $P, X, Y (1 \leq P \leq N^2; 1 \leq X, Y \leq N)$. Người chơi cần đưa số nguyên P đến vị trí hàng X cột Y với số lần dịch bảng nhỏ nhất bằng cách sau:

- Dịch các số trên hàng chứa số P sang phải hoặc sang trái một ô theo vòng tròn cho đến khi số P nằm trên cột Y;
- Dịch các số trên cột Y lên trên hoặc xuống dưới một ô theo vòng tròn cho đến khi số P nằm trên hàng X.
- Mỗi theo tác dịch hàng hoặc cột như trên được tính là một lần dịch bảng. Bảng đầu tiên của lượt chơi sau chính là bảng kết thúc của lượt chơi trước.

Yêu cầu: Cho thông tin của Q lượt chơi. Hãy lập trình xác định số lần dịch bảng nhỏ nhất để đưa số P đến vị trí hàng X cột Y trong mỗi lượt chơi.

Dữ liệu đầu vào: Gồm $Q + 1$ dòng:

- Dòng 1: Chứa hai số nguyên dương $N, Q (1 \leq N \leq 30000; 1 \leq Q \leq 2000)$ cách nhau một ký tự trắng.
- Dòng $i + 1 (1 \leq i \leq Q)$: Chứa ba số nguyên dương $P_i, X_i, Y_i (1 \leq P_i \leq N^2; 1 \leq X_i, Y_i \leq N)$ cách nhau một ký tự trắng.

Dữ liệu đầu ra: Gồm Q dòng, dòng thứ i chứa một số nguyên là kết quả của lượt chơi thứ i.

Ràng buộc dữ liệu:

- Subtask 1 (40%): $1 \leq N < 100; 1 \leq Q \leq 100$;
- Subtask 2 (40%): $1 \leq N < 1500; 1 \leq Q \leq 1500$;
- Subtask 3 (20%): Không có thêm ràng buộc nào.

Ví dụ:

Input

```
5 3
17 2 5
5 4 2
18 1 1
```

Output

4
2
4

Giải thích: - Lượt chơi thứ nhất: Bảng ban đầu có dạng như hình trên. Để đưa số 17 đến vị trí hàng 2 cột 5, ta có thể thực hiện các bước như hình sau:

....

Hướng dẫn giải:

Solution C++:

15.5 Bài 5: Mua hàng

Đề bài: An đi mua M sản phẩm khác nhau, các sản phẩm được đánh số từ 1 đến M. Ở chợ có N quầy hàng được xếp thành hàng ngang được đánh số từ 1 đến N, từ trái sang phải. Quầy hàng thứ i chỉ bán một loại sản phẩm duy nhất là $A_i(1 \leq A_i \leq M)$ và với mỗi sản phẩm trong M sản phẩm luôn tồn tại ít nhất một quầy hàng bán sản phẩm loại đó. Thời gian để An mua sản phậm tại quầy hàng thứ i là T_i phút. Thời gian để di chuyển giữa hai quầy hàng liền kề là 1 phút.

Yêu cầu: Tìm cách mua hàng sao cho:

- An mua đủ M sản phẩm theo đúng thứ tự từ 1 đến M. Có thể bắt đầu từ một quầy hàng bất kì bán sản phẩm 1;
- Thời gian tính từ lúc bắt đầu mua sản phẩm 1 đến khi mua xong sản phẩm M là nhỏ nhất;

Dữ liệu đầu vào: Gồm ba dòng:

- Dòng 1: Chứa hai số nguyên dương $N, M(1 \leq M \leq N \leq 10^5)$ cách nhau một ký tự trắng.
- Dòng 2: Chứa N số nguyên dương $A_1, A_2, ..., A_N(1 \leq A_i \leq M; 1 \leq i \leq N)$ cách nhau một ký tự trắng.
- Dòng 3: Chứa N số nguyên dương $T_1, T_2, ..., T_N(1 \leq T_i \leq 10^9; 1 \leq i \leq N)$ cách nhau một ký tự trắng.

Dữ liệu đầu ra: Gồm một dòng chứa một số nguyên là số phút nhỏ nhất để An mua M sản phẩm.

Ràng buộc dữ liệu:

- Subtask 1 (10%): $M = 1$;
- Subtask 2 (30%): $M = N$;
- Subtask 3 (30%): $N \leq 2000$;
- Subtask 4 (30%): Không có thêm ràng buộc nào.

Ví dụ:

Input

```
5 2
1 2 1 1 2
5 10 6 8 3
```

Output

```
11
```

Giải thích: Cách mua sao cho tổng số phút nhỏ nhất là:

- Mua sản phẩm 1 tại quầy hàng 3 (tốn 6 phút);
- Di chuyển từ quầy hàng 3 đến quầy hàng 5 (tốn 2 phút);
- Mua sản phẩm 2 tại quầy hàng 5 (tốn 3 phút).
- Tổng thời gian là $6 + 2 + 3 = 11$ phút.

Hướng dẫn giải:
Solution C++: