

Bài 9: Đếm số dãy tương tự dãy a mà có tích các số là chẵn.

- Dãy tương tự khi $|b[i] - a[i]| \leq 1$ với mọi i .
- $b[i]$ có thể nhận một trong 3 giá trị: $a[i] - 1, a[i], a[i] + 1$.
- Sử dụng duyệt đệ quy để thử tất cả các trường hợp.

```
int n, a[11], b[11], ans = 0;

void check() {
    int rem = 1;
    for (int i = 0; i < n; ++i)
        rem = (rem * b[i]) % 2;
    ans += rem == 0;
}

void backtrack(int i) {
    if (i == n) {
        check();
        return;
    }
    // Duyệt các khả năng của b[i].
    for (int j = a[i] - 1; j <= a[i] + 1; ++j) {
        b[i] = j;
        backtrack(i + 1);
    }
}
```

Bài 10: Tìm dãy con liên tiếp dài nhất có đúng K số nguyên tố phân biệt

Sử dụng 2 con trỏ để duyệt dãy.

Hai con trỏ là gì?

```
a1 a2 a3 a4 a5 a6 a7 a8 a9 a10
^-----^
|           |
left        right
```

Khi dịch sang phải, ta thấy rằng:

```
a1 a2 a3 a4 a5 a6 a7 a8 a9 a10
^-----^
```

So với trường hợp trước, ta thấy rằng: - Mất đi a1 và thêm vào a6. - Chỉ cần kiểm tra lại a1 và a6 $\Rightarrow O(1)$, (không cần duyệt lại toàn bộ dãy).

Quay lại bài toán:

B1: Đánh dấu các vị trí không phải số nguyên tố trong mảng là 0.

Cửa sổ trượt cần quản lý một đoạn dài nhất có đúng K số nguyên tố phân biệt. - Khi dịch sang phải một ô mới, thì có hai trường hợp: + Nếu $a[right + 1]$ là số nguyên tố mới thì số lượng số nguyên tố phân biệt tăng lên. + Nếu số lượng số nguyên tố phân biệt vượt quá K thì cần cắt bỏ phần đầu dãy để giảm số lượng số nguyên tố phân biệt về K. + Nếu $a[right + 1]$ là số nguyên tố cũ hoặc không phải số nguyên tố thì không cần làm gì cả. \Rightarrow Cần mảng thống kê số lần xuất hiện của các số nguyên tố trong cửa sổ trượt. - Nếu số lượng số nguyên tố phân biệt bằng K thì cập nhật kết quả.

```
void solve() {
    int n, k;
    cin >> n >> k;
    vector<int> a(n);
    for (int &x : a) cin >> x;
```

```

vector<int> is_prime(1000001, 1);
is_prime[0] = is_prime[1] = 0;
for (int i = 2; i <= 1000000; ++i) {
    if (is_prime[i]) {
        for (int j = i * 2; j <= 1000000; j += i)
            is_prime[j] = 0;
    }
}

// B1: Đánh dấu các vị trí không phải số nguyên tố trong mảng là 0.
for (int &x: a)
    if (!is_prime[x]) x = 0;

// B2: Cửa sổ trượt.
int ans = -1, L = 0, R = -1;
vector<int> cnt(1000001, 0); // Thống kê số lần xuất hiện của các số nguyên tố trong cửa sổ trượt.
int primes = 0; // Số lượng số nguyên tố phân biệt trong cửa sổ trượt.
while (1) {
    // Dịch sang phải một ô.
    R++;
    if (R == n) break;
    if (a[R] != 0 and cnt[a[R]] == 0) primes++;
    cnt[a[R]]++;

    // Nếu số lượng số nguyên tố phân biệt vượt quá K thì cần cắt bỏ phần đầu dãy để giảm số lượng số nguyên tố.
    while (primes > k) {
        if (a[L] != 0) {
            cnt[a[L]]--;
            if (cnt[a[L]] == 0) primes--;
        }
        L++;
    }

    // Phải đảm bảo đoạn [L, R] không vi phạm điều kiện.

    // Nếu số lượng số nguyên tố phân biệt bằng K thì cập nhật kết quả.
    if (primes == k) ans = max(ans, R - L + 1);
}
cout << ans << '\n';
}

```

Bài 11: Số lượng số hạn chế

- Số hạn chế là số có đúng 4 ước trong đó có 2 ước nguyên tố.
- Đếm trong đoạn $[L, R]$ có bao nhiêu số hạn chế. \Leftrightarrow Quy về đếm số hạn chế trong đoạn $[1, R]$ - đếm số hạn chế trong đoạn $[1, L - 1]$.
- Gọi $cnt[x]$ là số lượng số hạn chế trong đoạn $[1, x]$.
- $cnt[x] = cnt[x - 1] + (x \text{ có đúng 4 ước trong đó có 2 ước nguyên tố})$.

Cách kiểm tra x có đúng 4 ước trong đó có 2 ước nguyên tố:

x có 4 ước thì phải thuộc một trong hai dạng: - $x = p^3 \Rightarrow 1, p, p^2, x$ - $x = p \cdot q \Rightarrow 1, p, q, x$

x phải là tích 2 số nguyên tố phân biệt, tức là $x = p \cdot q$ với p, q là số nguyên tố phân biệt. \Rightarrow Theo công thức đếm ước, số ước của $x = (a + 1) \cdot (b + 1)$ với a, b là số mũ của p, q .

- Cách 1: Sàng ước \Rightarrow Tính được mảng số ước của các số từ 1 đến $10^6 \Rightarrow uoc[x]$ = số ước của x .
- Nếu $uoc[x] == 4$ và x không phải số lập phương thì x là số hạn chế.
- Cách 2: Tự nghĩ.

```

int uoc[200005], cnt[200005];

bool hanche(int x) {
    if (uoc[x] != 4) return false;
    int k = cbrt1(x);
    return k * k * k != x;
}

void solve() {
    // Sàng nguyên tố =>  $O(n * \log(\log(n)))$ .  $\sim 10^7$ .
    // Sàng ước => Rất quan trọng =>  $O(n * \log(n))$ .  $\sim 10^6$ .
    for (int i = 1; i <= 200000; ++i)
        for (int j = i; j <= 200000; j += i)
            uoc[j]++;
    // Tính cnt.
    for (int i = 1; i <= 200000; ++i)
        cnt[i] = cnt[i - 1] + hanche(i);
    int t;
    cin >> t;
    while (t--) {
        int l, r;
        cin >> l >> r;
        cout << cnt[r] - cnt[l - 1] << '\n';
    }
}

```

Bài 12: Nguyên tố tương đương

A, B là nguyên tố tương đương khi A và B cùng tập ước nguyên tố \Leftrightarrow Tích các ước nguyên tố của A = Tích các ước nguyên tố của B .

\Rightarrow Sử dụng sàng nguyên tố + sàng ước để tính tích cách ước nguyên tố của các số từ 1 đến 10^6 .

Quy về bài toán đếm số cặp $(A < B)$ có $product[A] == product[B] \Rightarrow$ Bài toán thống kê cơ bản.

```

int product[1000005], cnt[1000005];
bool is_prime[1000005];

void solve() {
    // Sàng nguyên tố.
    fill(is_prime, is_prime + 1000001, 1);
    is_prime[0] = is_prime[1] = 0;
    for (int i = 2; i <= 1000; ++i) {
        if (is_prime[i]) {
            for (int j = i * i; j <= 1000000; j += i)
                is_prime[j] = 0;
        }
    }
    // Sàng ước => tính product.
    fill(product, product + 1000001, 1);
    for (int i = 1; i <= 1000000; ++i)
        if (is_prime[i])
            for (int j = i; j <= 1000000; j += i)
                product[j] *= i;
    int t;
    cin >> t;
    while (t--) {
        int a, b;
        cin >> a >> b;
        memset(cnt, 0, sizeof(cnt));
    }
}

```

```

    long long ans = 0;
    for (int i = a; i <= b; ++i) {
        ans += cnt[product[i]];
        cnt[product[i]]++;
    }
    cout << ans << '\n';
}
}

```

Bài 15: Dãy tương tự hard

- Dãy tương tự khi $|b[i] - a[i]| \leq 1$ với mọi i .
- $b[i]$ có thể nhận một trong 3 giá trị: $a[i] - 1, a[i], a[i] + 1$.
- Chỉ cần có một giá trị $b[i]$ chẵn thì tích các số trong dãy b là chẵn. \Rightarrow Nếu không có giá trị $b[i]$ chẵn thì tích các số trong dãy b là lẻ.
- Đếm bằng cách bù trừ \Rightarrow Lấy tổng số dãy tương tự - số dãy tương tự không có giá trị chẵn.
- Tổng số dãy tương tự $= 3^n$.
- Số dãy tương tự không có giá trị chẵn là: 2^k với k là số lượng vị trí $a[i]$ là chẵn. (Tự chứng minh)
- Đáp án là $3^n - 2^k$.