Brute Force: Multiple Queries

It makes all the difference

UTEC - Competitive Programming

Challenge

Given a number n, determine if it is prime.

- Design an algorithm for $n \leq 10^8$

Challenge

Given a number n, determine if it is prime.

- Design an algorithm for $n \leq 10^8$
- Now make one for $n \leq 10^{18}$

Challenge

Given a number n, determine if it is prime.

- Design an algorithm for $n \leq 10^8$
- Now make one for $n \leq 10^{18}$
- What if $n \leq 10^7$, but there are q ($q \leq 10^7$) queries.

Sieve of Eratosthenes

- The following is a very famous sieve for calculating what numbers are prime.
- This sieve has a complexity of $O(nlg(lgn))$. *Why?*

```
bool notPrime[N];

void sieve() {
    for (int i = 2; i <= N; i++) {
        if (notPrime[i]) continue;
        for (int j = 2 * i; j <= N; j++) {
            isPrime[j] = false;
        }
    }
}
```

Preprocessing

- When we have multiple queries, we can sometimes compute some values that will help us answer queries.
- This is called preprocessing and the idea is that it reduces the time per query without being
- Lets say $T_p(n)$ is the complexity of preprocessing and $T_q(n)$ is the complexity per query after said preprocessing, the complexity of our program would be $O(max(T_p(n), QT_q(n)))$.

Challenge - Prefix Sums

You are given an array a of size n. You will receive q queries that will consist of to numbers l ans r. For each you need to print $\sum_{i=l}^{r} a_i$.

- Design an algorithm to solve this for $n, q \leq 10^4$.

Challenge - Prefix Sums

You are given an array a of size n. You will receive q queries that will consist of to numbers l ans r. For each you need to print $\sum_{i=l}^{r} a_i$.

- Design an algorithm to solve this for $n, q \leq 10^4$.
- Can we solve this problem for $n, q \leq 10^6$?

Solution - Prefix Sums

- By preprocessing the array and calculating the sum for any prefix we can answer queries with a complexity of O(1).
- The final complexity for our solution would be O(min(n, q)).

```
int main() {
    int n, q;
    cin >> n >> q;

    int a[n], pref[n];
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
        pref[i] = pref[i - 1] + a[i];
    }

    int l, r;
    for (int i = 0; i < q; i++) {
        cin >> l >> r;
        cout << pref[r] - pref[l - 1] << endl;
    }

    return 0;
}
```

Challenge - Offline Processing

You are given q queries. Each consists of 2 numbers n ans p. For each query output $|\{x : x \leq n \wedge p|x\}|$. $q \leq 10^6$ and $p \leq n \leq 10^6$

- Any ideas?

Offline Processing

- In many cases the order in which we answer the queries doesn't affect their answers.
- Offline processing is changing the order of queries in order to improve overall speed.
- **IMPORTANT:** In order to be able to process queries offline, all queries must be independent (The result of a query is not affected by previous queries).

Solution - Offline Processing

- We can use a sieve to find all the divisors of a number.
- Order the queries in ascending order according to their n.
- Initialize counter `cnt`. `cnt[i]` stores how many values are divisible by i.
- Iterate x through all numbers from 1 to n_{max} and increase the counter of all divisors of x by one.
- Answer all queries where $n_i = x$.

Thanks for Listening!