Team notebook

October 20, 2023

Contents

1 Codes

1.1 Data Structures

1.1.1 segment$_t$*ree*

```
const int N = 100000 + 5;
const long long inf = 1e18 + 10;

struct node {
	long long sum;
	long long maxi;
	node(){
		sum = 0;
		maxi = -inf;
	}

	node(long long x) {
		sum = maxi = x;
	}

	node operator + (const node &rhs) const {
		node q;
		q.sum = sum + rhs.sum;
		q.maxi = max(maxi, rhs.maxi);
		return q;
	}
};

int n;
int q;
node NIL;
long long a[N];
node st[4 * N];

void build(int pos = 1, int l = 1, int r = n) {
	if(l == r) {
		st[pos] = node(a[l]);
		return;
	}
	int mi = (l + r) / 2;
	build(2 * pos, l, mi);
	build(2 * pos + 1, mi + 1, r);
	st[pos] = st[2 * pos] + st[2 * pos + 1];
}

void update(int x, int y, int pos = 1, int l = 1, int r = n) {
	if(st[pos].maxi <= 1) return; // Funcion Potencial sqrt(1) = 1
	if(y < l or r < x) return;
	if(l == r) {
	// to change
		st[pos].sum = sqrt(st[pos].sum);
		st[pos].maxi = st[pos].sum;
		return;
	}
	int mi = (l + r) / 2;
```

```
        update(x, y, 2 * pos, l, mi);
        update(x, y, 2 * pos + 1, mi + 1, r);
        st[pos] = st[2 * pos] + st[2 * pos + 1];
}

node query(int x, int y, int pos = 1, int l = 1, int r = n) {
        if(y < l or r < x) return NIL;
        if(x <= l and r <= y) return st[pos];
        int mi = (l + r) / 2;
        return query(x, y, 2 * pos, l, mi) + query(x, y, 2 * pos + 1, mi +
            1, r);
}

int main() {
        build();
    update(l, r);
    query(l, r).sum;
    query(l, r).maxi;
}
```

1.2 Number theory

1.2.1 natural$_s$*ieve*

```
const int N = 100000000 + 5;
vector<int> primes;
bitset<N> composite;

void lineal(int n){
    for(int i = 2; i <= n; i++) {
                if(not composite[i]) primes.emplace_back(i);
                for(int p : primes) {
                        if(i * p > n) break;
                        composite[i * p] = true;
                        if(i % p == 0) break;
                }
        }
}
```

1.2.2 sieve

```
const int N = 100000000 + 5;
bitset<N> composite;

void natural(int n){
    // (WARNING) Todos los pares son primos
    for(int i = 3; i * i <= n; i += 2) {
                if(not composite[i]) {
                        for(int j = i * i; j <= n; j += i) composite[j] =
                            true;
                }
        }
}
```