

# Competitive Programming Reference

ngmh

December 2025

## Contents

<b>1</b>	<b>Data Structures</b>	<b>2</b>
1.1	Prefix Sums . . . . .	2
1.1.1	1D . . . . .	2
<b>2</b>	<b>Graph Theory</b>	<b>2</b>
<b>3</b>	<b>Dynamic Programming</b>	<b>2</b>
<b>4</b>	<b>Math</b>	<b>2</b>
<b>5</b>	<b>Algorithms</b>	<b>2</b>
5.1	Binary Search . . . . .	2
5.2	Binary Search using Lifting . . . . .	2
<b>6</b>	<b>Miscellaneous</b>	<b>3</b>
6.1	Fast I/O . . . . .	3
6.2	Superfast I/O . . . . .	3

# 1 Data Structures

Data Structure	Precomputation / Update	Query	Memory	Notes
Prefix Sum	$O(N) / X$	$O(1)$	$O(N)$	Associative Functions (+, XOR)
Sparse Table	$O(N \log N) / X$	$O(1)$	$O(N \log N)$	Non-Associative Functions (max, gcd)
Fenwick Tree	$X / O(\log N)$	$O(\log N)$	$O(N)$	Prefix Sum with Updates
Segment Tree	$X / O(\log N)$	$O(\log N)$	$O(4N)$	Allows more Information

Table 1: Quick Summary of Data Structures

## 1.1 Prefix Sums

### 1.1.1 1D

---

```
//Query - 1-Indexed
int query(int s, int e){
    return ps[e]-ps[s-1];
}

//Precomputation
for(int i = 1; i <= n; i++) ps[i] = ps[i-1]+a[i];
```

---

## 2 Graph Theory

## 3 Dynamic Programming

## 4 Math

## 5 Algorithms

### 5.1 Binary Search

Find the cuberoot of  $n$ .

---

```
long long n; cin >> n;
long long mini = 0, maxi = 1e6, medi;
while (mini < maxi) {
    medi = mini+(maxi-mini)/2;
    if (medi * medi * medi >= n) maxi = medi;
    else mini = medi+1;
}
cout << mini << "\n";
```

---

### 5.2 Binary Search using Lifting

Find the cuberoot of  $n$ .

---

```
long long n; cin >> n;
long long cur = 0, gap = 1e6, next;
while (gap > 0) {
    while (next = cur + gap, next * next * next < n) {
        cur = next;
    }
    gap >>= 1;
}
cout << cur+1 << "\n";
```

---

## 6 Miscellaneous

### 6.1 Fast I/O

Cannot use with `scanf`, `printf`.

---

```
ios_base::sync_with_stdio(false);  
cin.tie(0);
```

---

### 6.2 Superfast I/O

Only for non-negative integer input!

---

```
inline ll ri () {  
    ll x = 0;  
    char ch = getchar_unlocked();  
    while (ch < '0' || ch > '9') ch = getchar_unlocked();  
    while (ch >= '0' && ch <= '9') {  
        x = (x << 3) + (x << 1) + ch - '0';  
        ch = getchar_unlocked();  
    }  
    return x;  
}
```

---