# TIME COMPLEXITY AND ARRAYS

CPU / Memory

X → $10^9 \times X$

Y → $10^9 \times Y$

200,000 → 2000$

100,000 × 2000$

200M $

---

N → prime or not

2, N-1

+ - / * % = → 1 unit CPU time.

2 → 1 unit

3

⋮

N-1

```
for i = 2 to N-1 :
    if (N % i == 0):
        N is not pr.
        return false
return true
```

O(1)

(N-1) - 2 + 1 = N-2 unit operations

```
int s = sqrt(N)  → 20 operat
for i=1 to sqrt(N):
    if N % i == 0:
        return false
return true
```

a<b

N = a * b

a ⇒ sqrt(N)

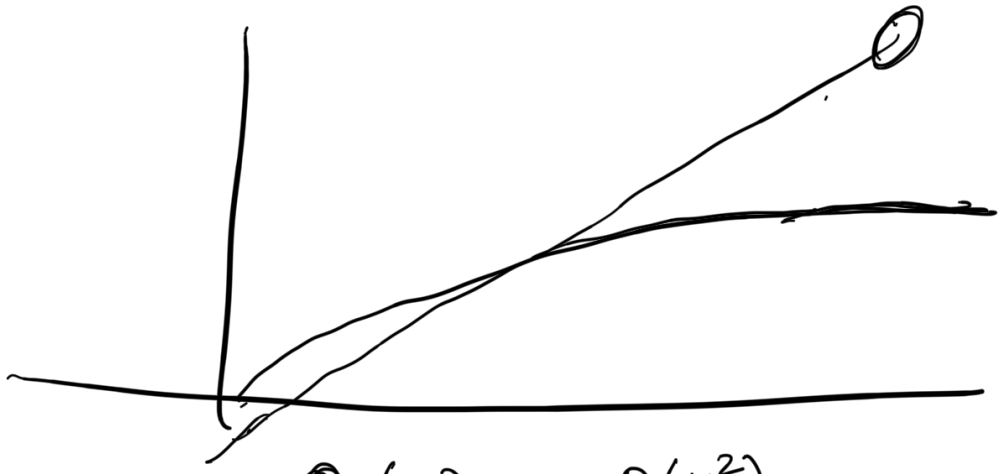
(sqrt(N) - 2)

(√N + 20)

N = 10,000

100

9998



$O(N)$ $O(N^2)$

$O(x) = O(y)$ if $c * x$ $y$

Constant $*$ $O(2N)$ $O(N)$ $O(2N) = O(N)$

$\frac{1}{2} * 2N$ $N$ $O(4N) = O(N)$

$O(N^3)$ $O(N^4)$

$(2N^2 + 4N + 5) = O(N^2)$

Constant $* N^2 > 2N^2 + 4N + 5$ for large value of N

$1000 * N^2 > 2N^2 + 4N + 5 < N^2$

$a_i * N^i + a_{i-1} * N^{i-1} + \ldots$

$N * \log(N)$

$O(N * N)$ $O(N \log N)$

$O(N * \sqrt{N})$

```
for (i = 1; i < N; i *= 2) {        -> log(N)
    ans += i;
}
```

$O(\log_2 N)$

1, 2, 4, 8, 16, ... 64

$2^x > N$

(X) $2^x$ $2^x \geq N$

$x \geq \log(N)$

```
for (i = 1; i*i < N; i++) {        -> sqrt(N)
}
```

$i * i > N$

$i^2 > N$

$i > \sqrt{N}$

$O(\sqrt{N})$

$O(N^2)$

$O(N)$

```
int j = 0;
for (int i = 0; i < n; i++) {
    while (j < n && arr[i] < arr[j]) j++;
}
```

$j = n$

N

$i = 0 \rightarrow j = n$

$i = 1 \rightarrow$

$i \rightarrow$

$N + 1 + 1 + 1 \cdots (N-1)$

$= (2N-1) \rightarrow O(N)$

---

## Space complexity :

CPU

Memory

Hashset H ← N

for i = 0 ; to size (A) ; → string of variable size

if H contains A[i] :

return true

O(N)

O(N*M)

H.insert (A[i])

N

200

10,000

2

M

max size of str

---

For i = 2; i < sqrt(N); i++)

if N % i == 0 ;

ans.insert(i)

ans.insert(N/i)

~~2^N~~ O( number of factors)

O(sqrt(N))

---


~~O(N)~~

O(N * size of arg)

Time vs Space

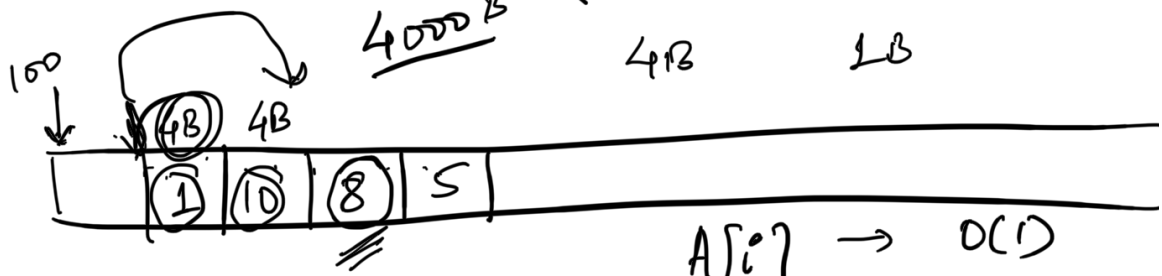
Contiguous set of homogenous elements.

(same data type)

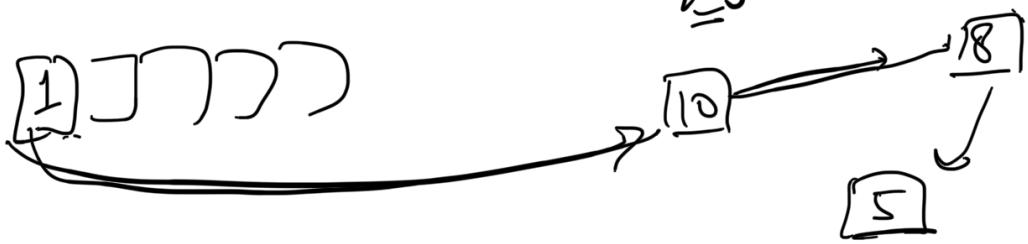
1000

4000 B

4B 1B




A[i] → O(1)




["abcd", 'a', 13] → [0 → "abcd", 1 → "a", 2 → 13]

---

A → [1, 3, 5, 13, 10, 2] ← Static

0 1 2 3 4 5

A[5] → 2

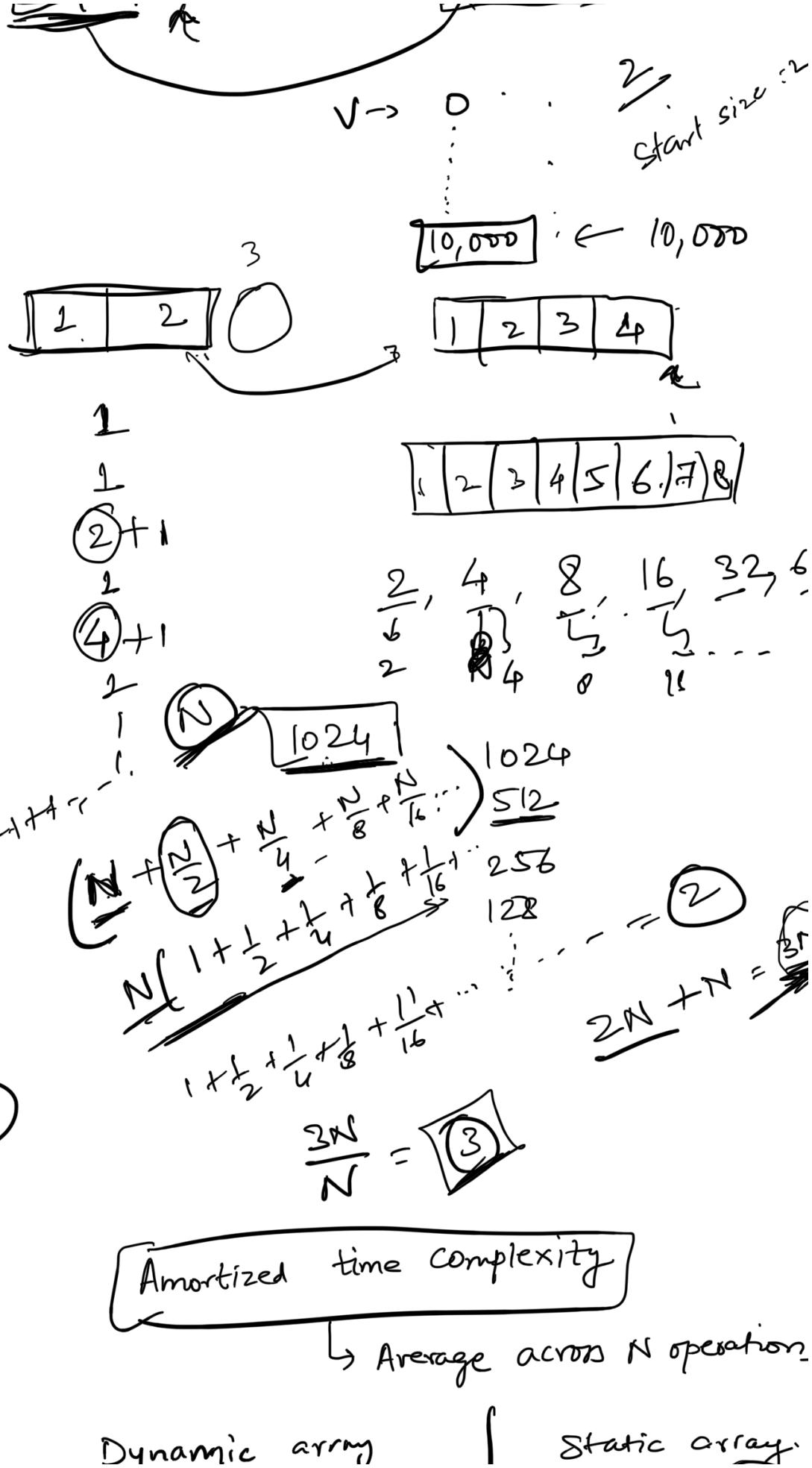
int A[100];

dynamic array → vector in C++ / Arraylist in Java

vector<int> V;

2, 10, 15

re-sizing factor → 1.6, 1.8, 2




Amortized time complexity

↳ Average across N operations

Dynamic array | Static array

vector

- slower
- Flexible

vector v (1000)

int arr[100]

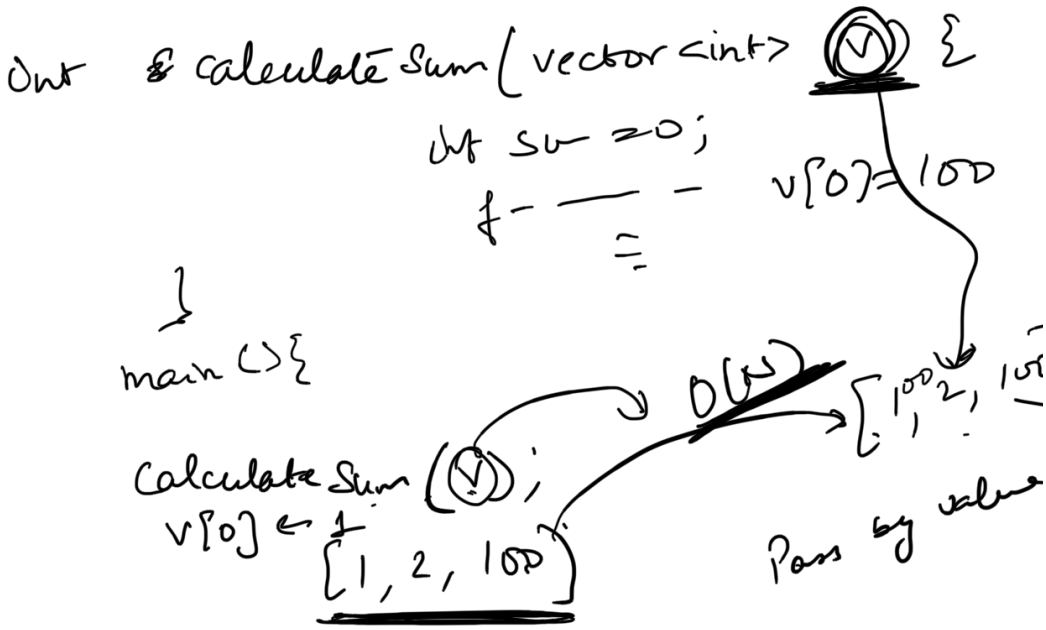
- Faster
- Not flexible
  - ↳ production

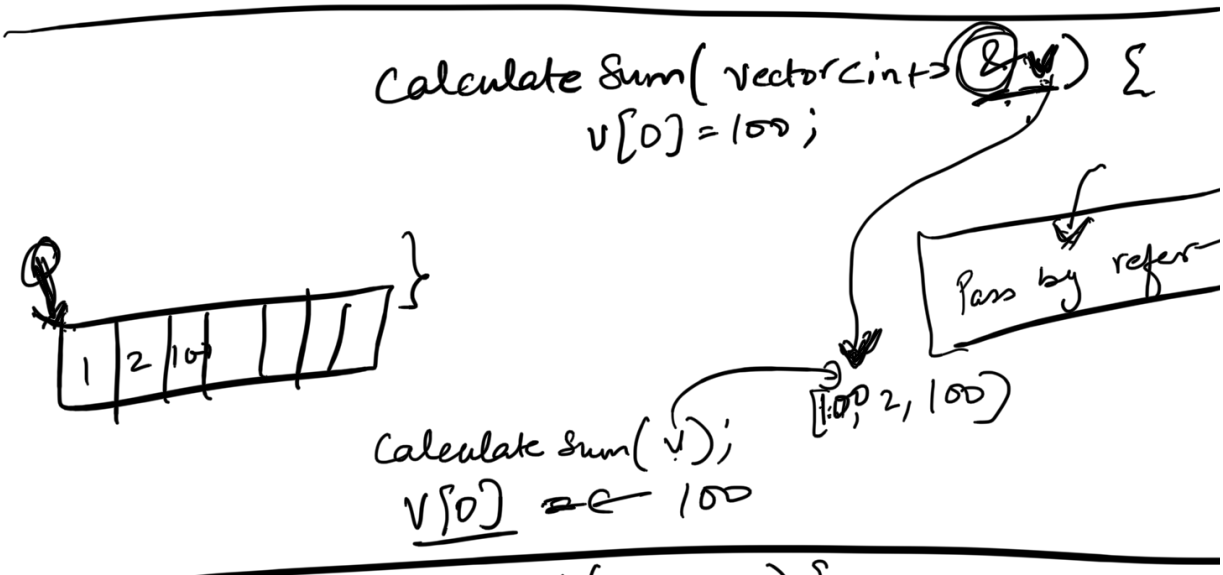
10,000

100,000

200,000

## Pass by value vs. pass by reference



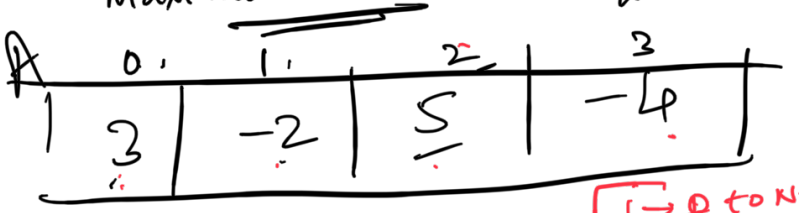



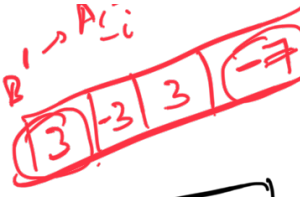

f( ) {
= obj.setVar( )

List<Integer> sampleList;

f(sampleList);

# Maximum Absolute Difference




$O(N^2)$

$|a-b|$

$a=2$ $b=5$ $|2-5| = 3$

$i \neq j$

$i \to 0$ to $N-1$
$j \to 0$ to $N-1$

$N \log N$

$3-(-7) = 10$

$i, j$

$$|A_i - A_j| + |i-j|$$

maximum

$|A_i - A_j| + |i-j|$

$|A_i - A_j|$

$\hookrightarrow A_i - A_j$

$\hookrightarrow A_j - A_i$

**Case 1:** $A_i > A_j$ and $i > j$

$$(A_i - A_j) + (i-j)$$

$$(A_i + i) - (A_j + j)$$

max, min

B


$B[i] = A[i] + i$

$B_i - B_j$

max, min

$7-(-1) = 8$

$$|A_i - A_j| + |i-j|$$

**Case 2:** $A_i > A_j$ and $i < j$

$$(A_i - A_j) + (j - i)$$

$$= (A_i - i) - (A_j - j)$$

B'


max, min

$|A_i - A_j|$

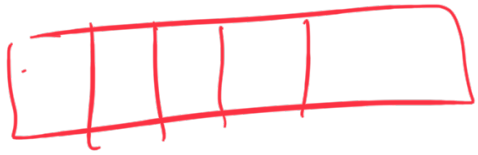
$i < j$

Case 2 $A_i < A_j$

$\hookrightarrow (A_j - A_i) + (j - i)$

$= (A_j + j) - (A_i + i)$

(10) $|A_i - A_j|$



Time complexity :

$O(N^2)$

```
for (i=0; i<N; i++){
    for (j=0; j<N; j++){
        if (i==j) continue;
        int num = abs(A[i] - A[j]) + abs(i-j);
        if num > max:
            max = num
```

Efficient

$B \hookrightarrow A_i + i$ — max

$B' \hookrightarrow A_i - i$ — min

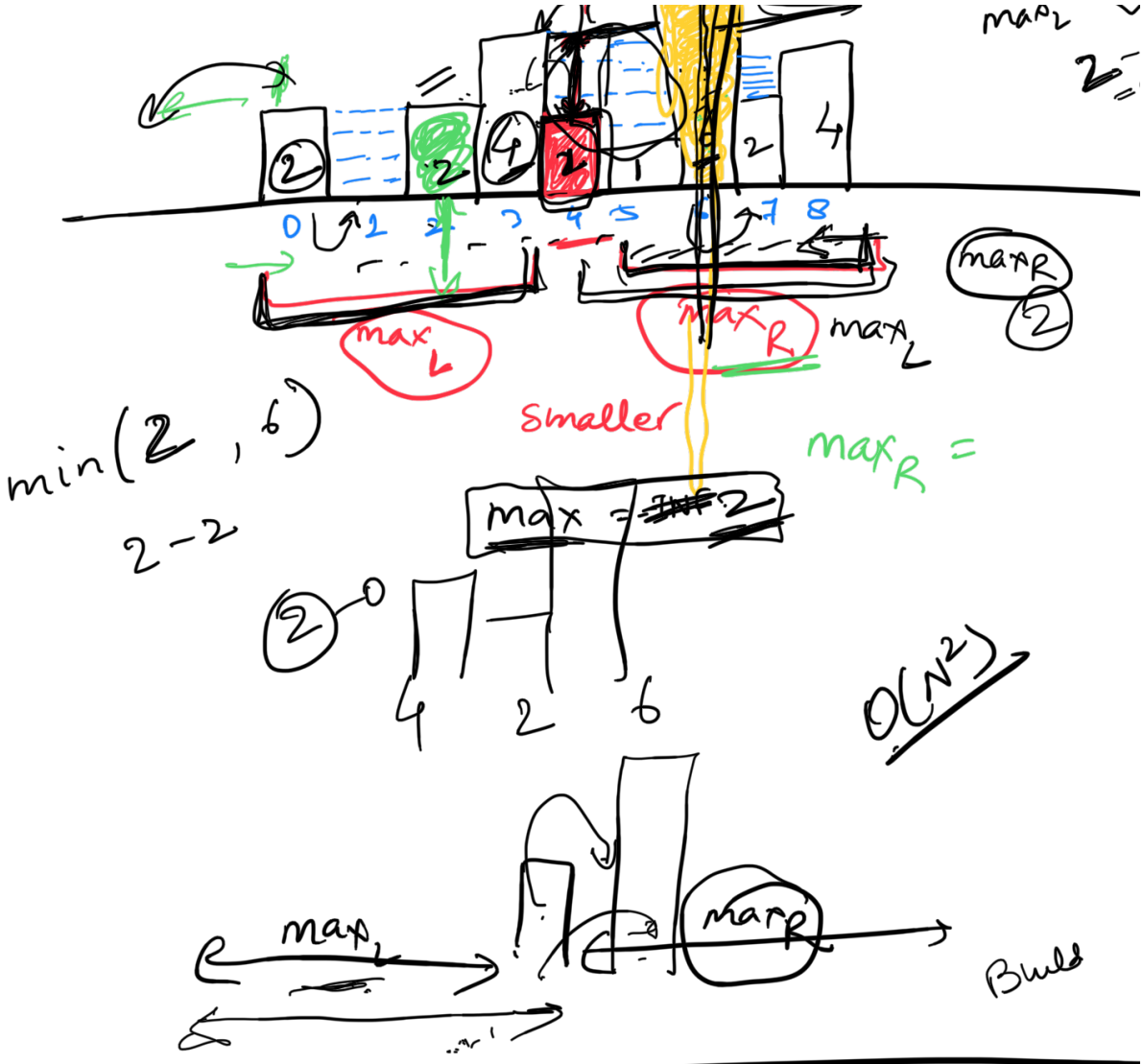
$O(N)$ $A_i + i$

```
int max = -INF, min = INF,
for (int i=0; i<N; i++){
    if ((A[i]+i) > max)
        max = A[i]+i;
    if ((A[i]+i) < min)
        min = A[i]+i;
}
```

$O(N) \leftarrow$ max − min ← case 1.

---

## RAIN WATER TRAPPED




for (int i=0; i<N; i++) {

$max_{left}$

$max_{right}$

$O(N^2)$ ←

0 — i-1 $max_{left} = INF$

for (j=0; j<=i; j++

$max_{left} = max($

$max_L, B[j]$

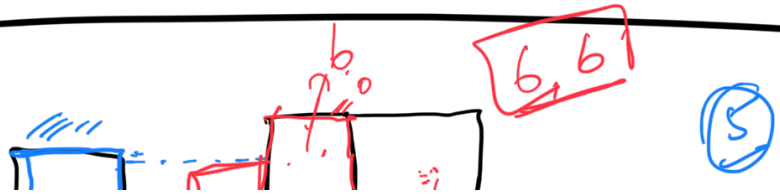
}

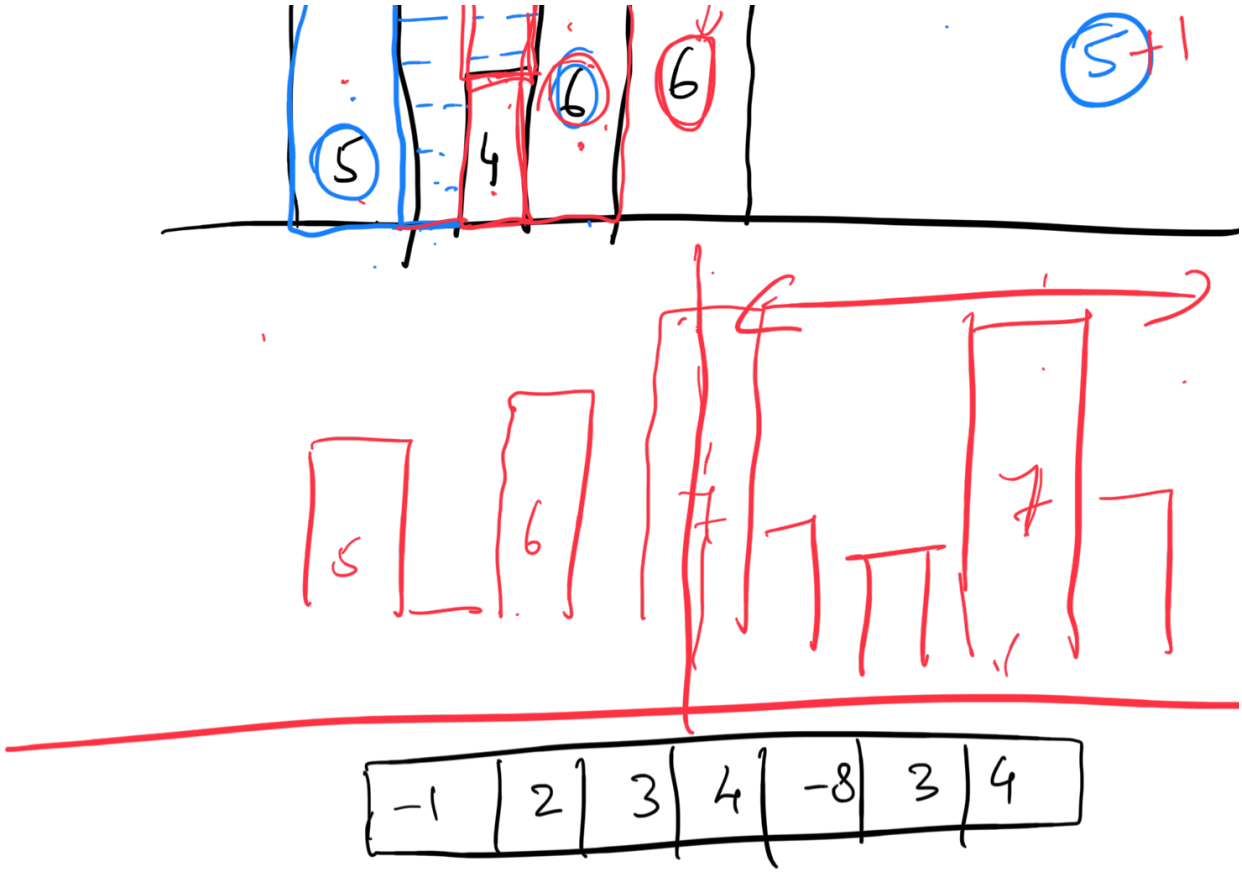
for (j=i+1; j<N; j++)

$max_{right} = max(max_R, A[$

$\min(max_L, max_R) - A[i]$

}






| -1 | 2 | 3 | 4 | -8 | 3 | 4 |
|---|---|---|---|---|---|---|