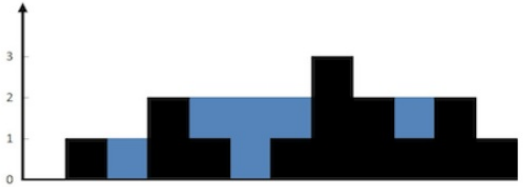
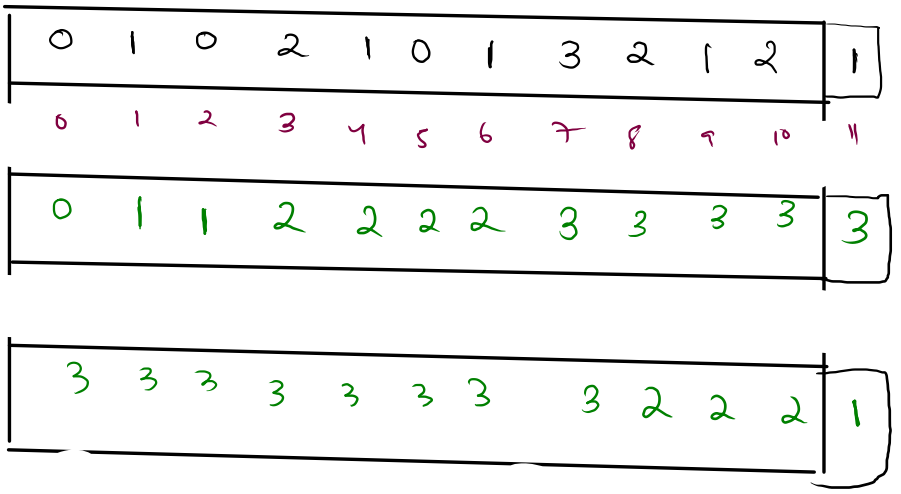
Store Maximum



Input: height = [0,1,0,2,1,0,1,3,2,1,2,1]

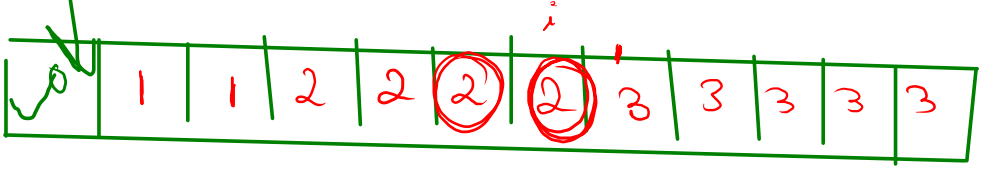
Sample Input 0

```
12
0 1 0 2 1 0 1 3 2 1 2 1
```



n=12

l2r (n)

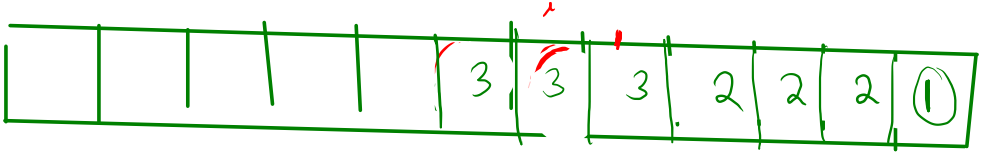


→

[0 ---- 11]

l2r[0] = A[0]

⇝ l2r[i] = Max(l2r[i-1], A[i])



```java
public class Solution {
    public static int trap(int[] height) {
        int n = height.length;
        int [] l2r = new int[n];
        l2r[0] = height[0];
        for(int i = 1; i < n; i++){
            l2r[i] = Math.max(l2r[i-1], height[i]);
        }

        int [] r2l = new int[n];
        r2l[n-1] = height[n-1];

        for(int i = n-2; i >= 0; i--){
            r2l[i] = Math.max(height[i], r2l[i+1]);
        }

        int ans = 0;
        for(int i = 0; i < n; i++){
            ans += Math.min(l2r[i], r2l[i]) - height[i];
        }
        return ans;

    }
```

```java
import java.io.*;
import java.util.*;

public class Solution {
    public static int trap(int[] height) {
        int n = height.length;
        int [] l2r = new int[n];
        l2r[0] = height[0];
        for(int i = 1; i < n; i++){
            l2r[i] = Math.max(l2r[i-1], height[i]);
        }
        int [] r2l = new int[n];
        r2l[n-1] = height[n-1];

        for(int i = n-2; i >= 0; i--){
            r2l[i] = Math.max(height[i], r2l[i+1]);
        }
        int ans = 0;
        for(int i = 0; i < n; i++){
            ans += Math.min(l2r[i], r2l[i]) - height[i];
        }
        return ans;

    }
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int [] A = new int[n];
        for(int i = 0; i < n; i++){
            A[i] = scn.nextInt();
        }
        int ans = trap(A);
        System.out.println(ans);
    }
}
```

```java
import java.util.*;
public class Main
{
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        for(int i = 0; i < n; i++){
            System.out.println(i);
        }
    }
}
```

i/p	o/p
5	5
500	500
1000	1000
10^5	10^5

depends on i/p

```java
import java.util.*;
public class Main
{
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        for(int i = 0; i < 10; i++){
            System.out.println(i);
        }
    }
}
```

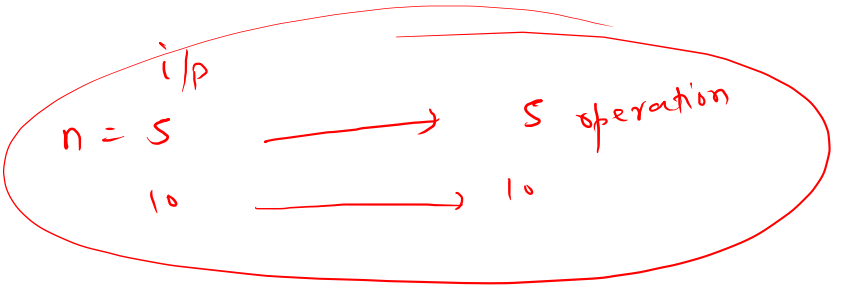
i/p	o/p
5	10
500	10
1000	10
10^5	10

constant itr

Time Complexity (TC)

* TC ≠ time to run your program.

* TC: any maths func^{c} to represent relationship between no. of i/p & operations.



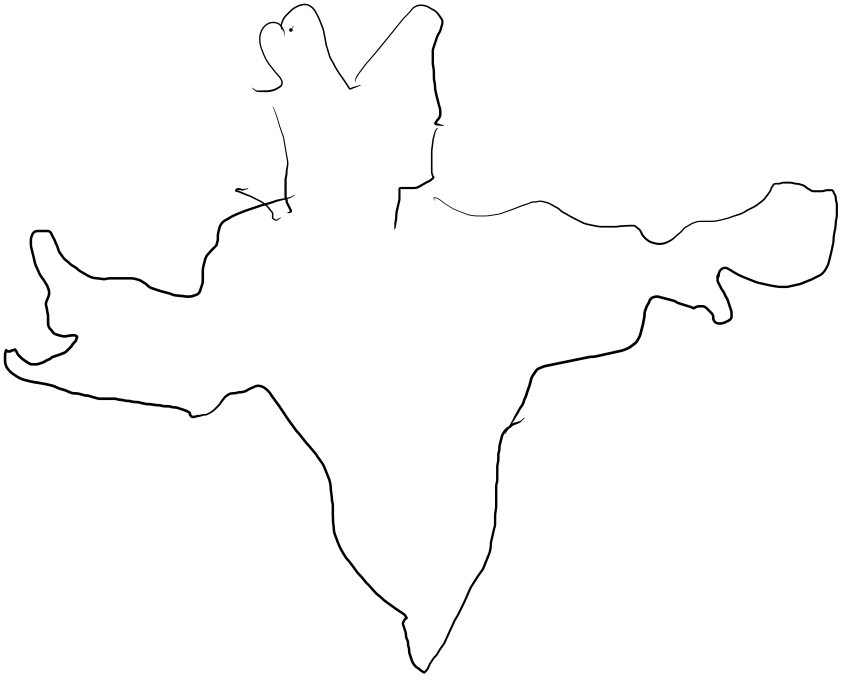
TC ?

Why ??

A	B
TC ↓	TC ↓
SC ↓	TC

Notations

O(1)



worst → Sagar

O

↳ atleast 1 state

avg → 5 states.

Θ

best → 29 states

Ω

omega

Constant

O(1)

O(k)

O(1 * k)

k(O(1))

maths (+ − * / %)

int age = 52 ;

Syso("Aman")

i/p

$y = 2x + k$

any.

1

10

6

50

$2x + k_1$

$2x + k_2$

$4x + k$

n = 5

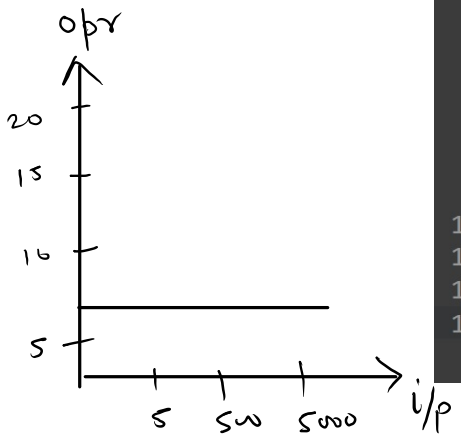
{ 4 unit

n = 500

{ 4 unit

n = 500000

{ 4 unit




```java
import java.util.*;
public class Main
{
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int a = 10;
        a = 17;
        int b = 20;
        System.out.println(a + b);
    }
}
```

O(1)

y = (K) } maths

O(K) ⇒ K.O(1)

= O(1)

$$y = x^2 + 2x + 5$$

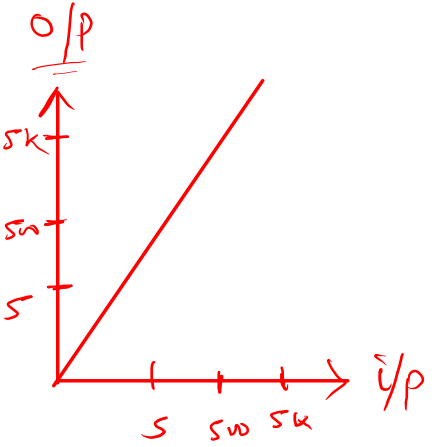
highest power = 2 quadratic

$$\Rightarrow O(n^3 + 2n + 5)$$

$$O(n^3)$$

O(n) → linear TC

i/p	o/p
5	5
50	50
500	500




```java
import java.util.*;
public class Main
{
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        for(int i = 0; i < n; i++){
            System.out.println(i);
        }
    }
}
```

7
52
502

n + 2

n + 2

O(n + 2)

O(n)

```java
public static int search(int [] A, int x){
    for(int i = 0 ;i < A.length; i++){
        if(A[i] == x){
            return i;
        }
    }
    return -1;
}
```

$k_1 x + k_2$

$2x + 3$ ← degree = 1

$TC = k_1 + n \times k_2$

$= O(n k_2)$

$= k_2 O(n)$

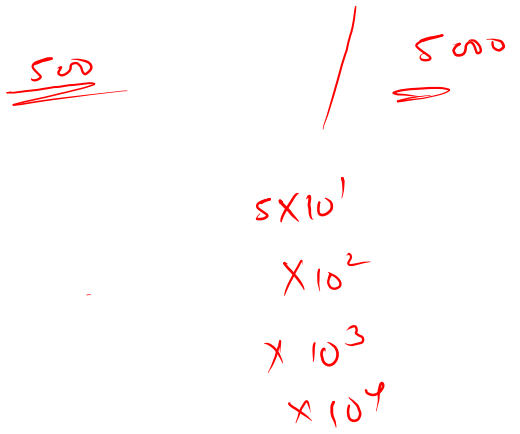
$= O(n)$

```java
import java.util.*;
public class Main
{
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        for(int i = 0; i < n; i++){
            int x = scn.nextInt();
            System.out.println(x);
        }
    }
}
```

k_1

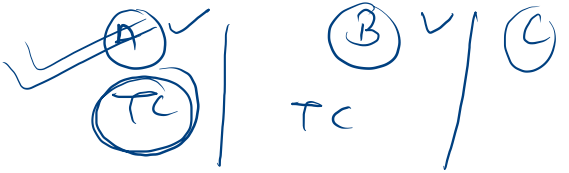
k_2

```java
import java.util.*;
public class Main
{
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        for(int i = 0; i < 10; i++){
            int x = scn.nextInt();
            System.out.println(x);
        }
    }
}
```



```java
import java.util.*;
public class Main
{
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        for(int i = 0; i < n; i++){
            System.out.println(i);
        }

        for(int i = 0; i < n; i++){
            System.out.println(i);
        }
    }
}

```

```java
import java.io.*;
import java.util.*;

public class Solution {

    public static int search(int [] A, int x){
        for(int i = 0 ;i < A.length; i++){
            if(A[i] == x){
                return i;
            }
        }
        return -1;
    }
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int [] A = new int[n];
        for(int i = 0; i < n; i++){
            A[i] = scn.nextInt();
        }
        int x = scn.nextInt();      //tar to be matched
        int ans = search(A, x);
        if(ans == -1){
            System.out.println("False");
        }else{
            System.out.println("True");
        }
    }
}
```



$k_1 + n \quad + n \quad + k_2$

$k_2 + k_1 + n + n \quad 2n + k_1 + k_2$

$2n$

$O(n)$ $\quad 2n + k$

$O(n)$