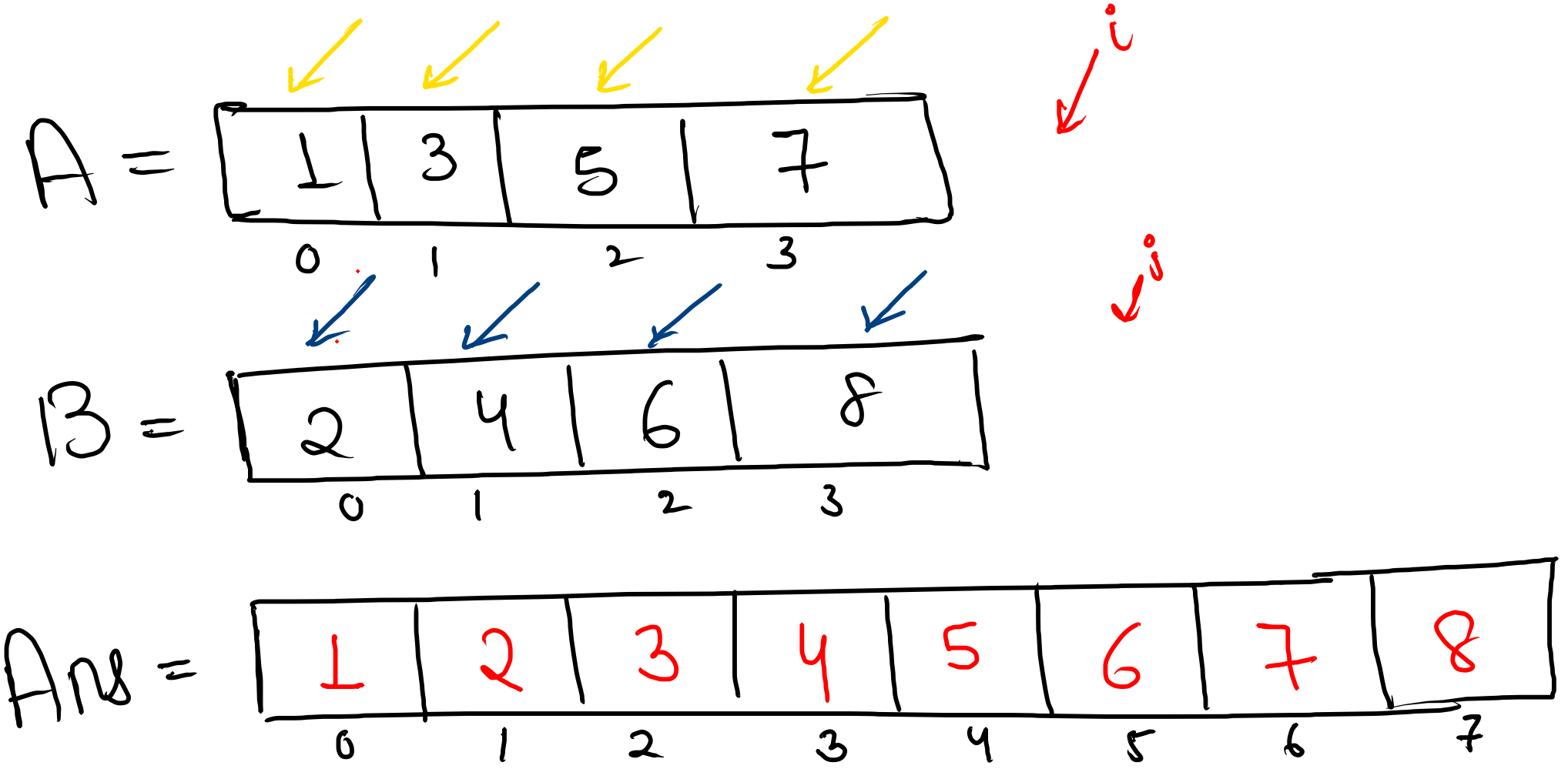
Duel merge 2 sorted arrays



⇒ Time Complexity

↳ It is a relation b/w Input size and running time

```
main() {
    Syso("Hello");   // O(1) → constant
    Syso("Hi");      // O(1) → constant
}
```

T.C :- $O(1+1) = O(2) \cong O(1)$

Note :- 10^9 operations in 1 sec

Constraints :- $n = 10^5$

$O(n^2) \Rightarrow O((10^5)^2)$

$\Rightarrow O(10^{10})$

Note :- ★★★

how to calculate time :-

how many operations we are performing in a program

Types of Operations

- linear operations ⇒ input n ⟶ output operations n
- quadratic operations ⇒ n ⟶ n^2
- cubic operations ⇒ n ⟶ n^3
- logerithmic operations ⇒ n ⟶ $\log(n)$
- constant operations ⇒ n ⟶ 1

Ex 1

```
P.S  void main() {
       // input
       int n = scn.nextInt();
       for (int i = 0; i<n; i++) {   // N
       }
}
```

T.C = O(N) where n is input size

relation:- input $\propto$ n

(linear)

↳ Time Complexity

1) Best case :- when least no. of operations has to be done
2) Average case :- when average no. of operations has to be done
3) Worst case :- when most no. of operations has to be done

ex:- find 1 in an array :-

input = (1) 2 5 7 3 4 :- best case // 1
input = 4 2 5 (1) 3 4 :- avg. case // 4
input = 7 2 5 7 3 (1) :- worst case // 6

ex 2

```
for(int i=0; i<n; i++) {      // n
    for(int j=0; j<m; j++) {  // m
        // Syso("Hi");
    }
}
```

T.C = O(N * M)

Note:- when we add complexities (time) then smaller value will get neglected because larger value have more significance.

ex 3

```
for(int i=0; i<n; i++) {      // n
    Syso("Hi");
}
for(int j=0; j<m; j++) {      // m
    Syso("Hi2");
}
```

T.C = O(n + m)

---

n = 1, m = 2 } → 3 O(m)

n = 200000000000, m = 20000 } 200002 O(m) O(n)

eXy

```
for(int i=0 ; i<n ; i+=2){
    Syso("Hi");
}
```

operation ⇒ n/2

T.C = $O(n/2) \cong O(n)$

$O(2*n) \cong O(n)$

$O(n-2) \cong O(n)$

$O(n+10) \cong O(n)$

Compare	$O(n)$	$O(n^2)$	$O(n^3)$
$n = 1$	1	1	1
$n = 2$	2	4	8
$n = 3$	3	9	27
⋮	⋮	⋮	⋮
$n = 10^5$	10^5	10^{10}	10^{15}

⇒ Space Complexity

↳ no. of variable / spaces you are using

P.S

```
void main() {
    Scanner scn = new Scanner();
    int n = scn.nextInt();
    int[] arr = new int[n]; // n
    for (int i=0; i<n; i++) {
        sysO(_);
    }
}
```

→ 1

S.C = O(1)

constant

S.C = O(n)

1) Constant T.C — $O(1)$ v. fast

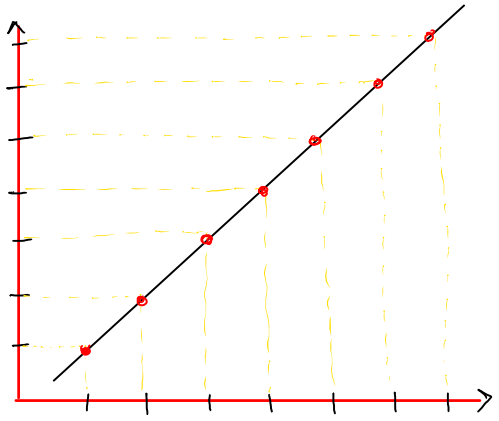
2) Lograthmic T.C — $O(\log N)$ fast

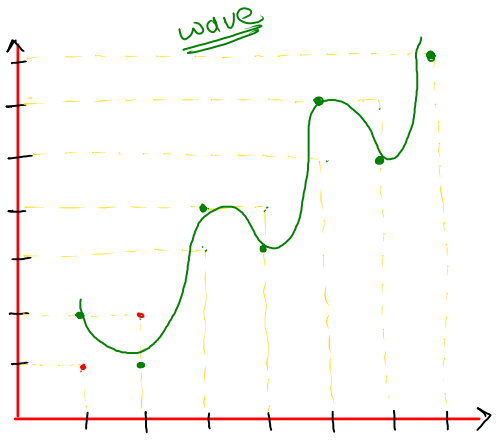
3) Linear T.C — $O(N)$ avg.

4) $O(n \log n)$ T.C slow

5) Quadratic T.C — $O(N^2)$ v. slow

Sort an array in wave form 1





Steps

1) Sort the array (to make a line)

2) Swap every consecutive element (to make a wave)

dry run

	10		90		49		2		1	5	23
step1	1		2		5		10		23	49	90
step2	2	>=	1	<=	10	>=	5	<=	49	>= 23	<= 90

code

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

    waveForm(arr, n);

    for (int i = 0; i < n; i++) {
        System.out.print(arr[i] + " ");
    }
}

public static void waveForm(int[] arr, int n) {
    // step1
    Arrays.sort(arr);

    // step2
    for (int i = 0; i < n - 1; i += 2) {
        swap(arr, i, i + 1);
    }
}

public static void swap(int[] arr, int x, int y) {
    int temp = arr[x];
    arr[x] = arr[y];
    arr[y] = temp;
}
```