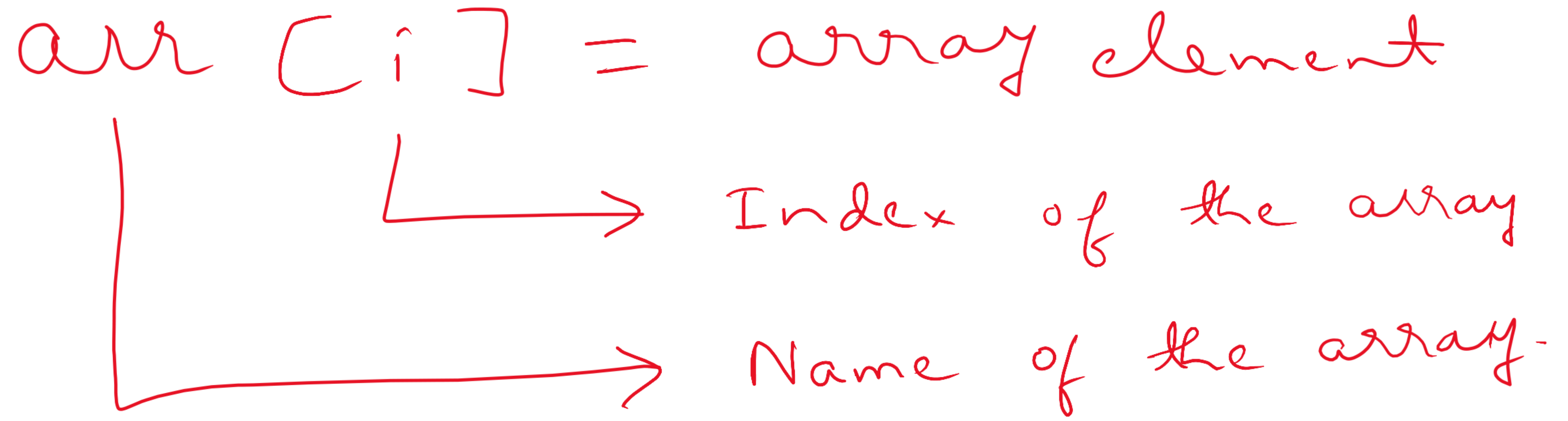
⊙ **Array** :- It is a collection of items (numbers, characters, etc) which are stored at contiguous memory location. It always starts with index 0.

eg:-
arr =

0	1	2	3	4	5
56	21	72	89	30	5
2000	2004	2008	2012	2016	2020

⊙ arr[i] = array element



Index of the array

Name of the array.

⊙ arr[0] = 56

- $$\text{arr} = \left[\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ \underline{20}, & 5, & 7, & 18, & 201, & 19 \\ \uparrow & & & & & \uparrow \\ \boxed{i} & & & & & \boxed{j} \end{array}\right]$$

- Sorted Array = [5, 7, 18, 19, 20, 201]

- { <u>Heap Sort</u>, <u>Merge Sort</u> }

  T.C = O(n·log n)

① Two Pointer Approach :-

1. In this technique, you will have two pointer to store index locations of first and last element.

2. Depending on the logic you choose, either you have to increment first pointer or decrement last pointer or do both.

3. Continue the step No 2, until the first pointer >= last pointer index location.

0	1	2	3	4	5	6
10	20	30	40	50	60	70

```
int i=0, j=6; while (i < j)
{ if (       )
        i++;
    else if (          )
   {
        j--;
}   else  i++; j--;
```

$$\text{① arr} = \left[\begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 26, & 5, & 31, & 81, & 3, & 10, & 15 \end{array}\right]$$

$$\boxed{\text{target} = 25}$$

(31, 3)

$$i = 0,\ j = 6,\ \underbrace{\text{arr}[i] + \text{arr}[j]} = 25 <$$

25

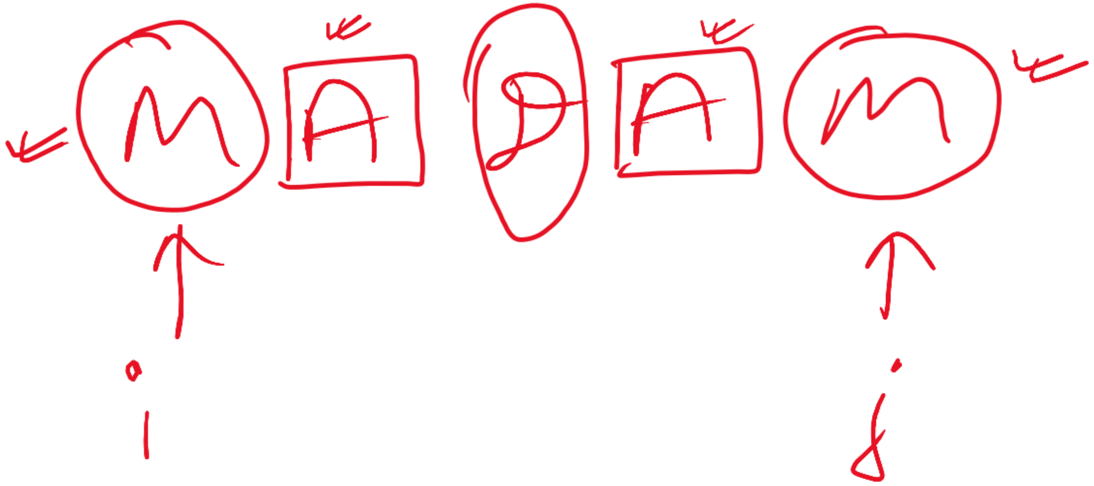
$$\text{⓪ arr} = \left[\begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 3, & 5, & 10, & 15, & 26, & 31, & 81 \\ 0 & & i & j & & & \end{array}\right] \ ⓝ$$

i = 3
j = 4

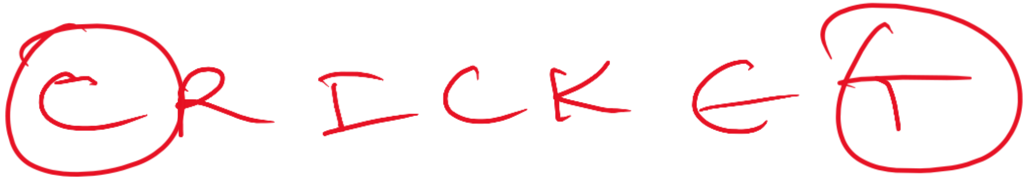
i++ (Greater Value)

j-- (Lesser Value)

①



①



① Index Manipulation Technique :-

It will be used in the problems where you need to restructure or reorder the input array elements.

```
         0  1  2  3  4  5  6  7  8
arr =  [ 1, 1, 1, 1, 2, 2, 2, 3, 3 ]

arr = [ 1, 2, 3 ]
        0  1  2

int k = 1;
for (int i = 1; i < n; i++)
{       if ( a[i] != a[i-1] )
           { a[k] = a[i];
                 k++;      // k = 3
           }
}
```