

Q. Sort an array in ascending order by the no. of factors. If the no. of factors are same then sort by value.

A: { 9, 3, 10, 6, 4 }
of factors \Rightarrow 3 2 4 4 3

A: { 3, 4, 9, 6, 10 }

Comparator function :-

* In any sorting algo, at a time we only need to compare 2 elements.

* No. of arguments = 2

* Based on the parameters & rules it should tell us which arg. should come first.

```
bool comp (int a, int b) {
```

```
    f1 = countfactors(a);
```

$\frac{a-b}{b-a}$ $\begin{matrix} \text{T} \\ \text{F} \end{matrix}$

```
    f2 = countfactors(b);
```

```
    if (f1 < f2) return true;
```

```
    if (f1 > f2) return false;
```

```
    if (a <= b) return true;
```

```
    return false;
```

3

C++

```
sort (v.begin(), v.end(), comp);
```

```
sort (a, a+n, comp);
```

array- size

Java

```
Arrays.sort ( Arr, new Comparator<Integer>() {  
    public int comp (inta, intb) {  
        // logic  
    }  
});
```

Q: Largest Number - Amazon / MakeMyTrip /
Paytm / Zoho / MS.

=> Given an Array of size N of non negative integers, Arrange them s.t they form the largest no.

* Result may be very large so return string instead of an integers.

A: [3, 30, 34, 5, 9]

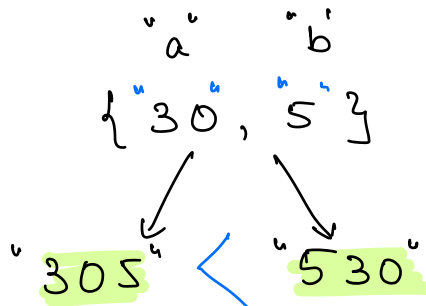
[9534330] ✓

$$1 \leq N \leq 10^5$$

$$0 \leq A[i] \leq 2 \times 10^9$$

Logic :-

ex



$$^a + ^b = ab$$

if (ab > ba) : (a) comes before (b)

$$^b + ^a = ba$$

else : (b) comes before (a).

```

bool comp ( string a , string b ) {
    if ( a+b > b+a ) {
        return true;
    }
    return false;
}

```

```

3
string largestNumber ( int A[], int N ) {
    string B[N];
    for ( i=0; i<N; i++ ) {
        B[i] = to_string(A[i]);
    }
    sort ( B , B+N , comp );
    string ans = ""
    for ( i=0; i<N; i++ )
        ans += B[i];
    return ans;
}

```

3

Comparison b/w sorting Algo's

	Best case	Worst case	Avg case
1) Selection sort	$O(N^2)$	$O(N^2)$	$O(N^2)$
2) Bubble sort	$O(N)$	$O(N^2)$	$O(N^2)$
3) Merge sort	$O(N \log N)$	$O(N \log N)$	$O(N \log N)$
4) Insertion sort	$O(N)$	$O(N^2)$	$O(N^2)$
5) Quick sort.	$O(N \log N)$	$O(N^2)$	$O(N \log N)$

When the
Array is getting
divided into half
every time.

