Largest Element in an Array

Given an array **arr** of size **N**, the task is to find the largest element in the given array.

Example:

> **Input:** arr[] = {10, 20, 4}
>
> **Output:** 20
>
> **Input :**
>
> **Output :**

Approach 1 - Linear Traversal: One of the most simplest and basic approach to solve this problem is to simply traverse the whole list and find the maximum among them.

Follow the steps below to implement this idea:

- Create a local variable **max** to store the maximum among the list
- **Initialize max with the first element** initially, to start the comparison.
- Then **traverse the given array** from second element till end, and for each element:
  - **Compare the current element with max**
  - If the current element is greater than max, then **replace** the value of **max** with the current element.
- At the end, **return** and print the value of the largest element of array stored in **max**.

Below is the implementation of the above approach:

```
// C++ program to find maximum
// in arr[] of size n
#include <bits/stdc++.h>
using namespace std;

int largest(int arr[], int n)
{
    int i;
```

```
    // Initialize maximum element
    int max = arr[0];

    // Traverse array elements
    // from second and compare
    // every element with current max
    for (i = 1; i < n; i++)
        if (arr[i] > max)
            max = arr[i];

    return max;
}

// Driver Code
int main()
{
    int arr[] = {10, 324, 45, 90, 9808};
    int n = sizeof(arr) / sizeof(arr[0]);
    cout << "Largest in given array is "
         << largest(arr, n);
    return 0;
}
```

Output

```
Largest in given array is 9808
```

Time complexity: O(N), to traverse the Array completely.

Auxiliary Space: O(1), as only an extra variable is created, which will take O(1) space.

Approach 2: Using Library Function: Most of the languages have a relevant max() type in-built function to find the maximum element, such as std: : max_element in C++. We can use this function to directly find the maximum element.

Below is the implementation of the above approach:

```
// C++ program to find maximum in arr[] of size n
#include <bits/stdc++.h>
using namespace std;
```

```
// returns maximum in arr[] of size n
int largest(int arr[], int n)
{
    return *max_element(arr, arr+n);
}

int main()
{
    int arr[] = {10, 324, 45, 90, 9808};
    int n = sizeof(arr)/sizeof(arr[0]);
    cout << largest(arr, n);
    return 0;
}
```

Output

```
9808
```

Time complexity: O(N), since the in-built max_element() function takes O(N) time. The Java Solution will take O(N logN) time complexity because of Arrays.sort(arr) function.

Auxiliary Space: O(1), as only an extra variable is created, which will take O(1) space.