How to check if a given array represents a Binary Heap?

Given an array, how to check if the given array represents a Binary Max-Heap.

Examples:

```
Input:  arr[] = {90, 15, 10, 7, 12, 2}
Output: True
The given array represents below tree
       90
     /    \
   15      10
  /  \     /
 7    12  2
The tree follows max-heap property as every
node is greater than all of its descendants.

Input:  arr[] = {9, 15, 10, 7, 12, 11}
Output: False
The given array represents below tree
       9
     /    \
   15      10
  /  \     /
 7    12  11
The tree doesn't follows max-heap property 9 is
smaller than 15 and 10, and 10 is smaller than 11.
```

We strongly recommend you to minimize your browser and try this yourself first.

A **Simple Solution** is to first check root, if it's greater than all of its descendants. Then check for children of root. Time complexity of this solution is $O(n^2)$

An **Efficient Solution** is to compare root only with its children (not all descendants), if root is greater than its children and same is true for for all nodes, then tree is max-heap (This conclusion is based on transitive property of > operator, i.e., if x > y and y > z, then x > z).

The last internal node is present at index (2n-2)/2 assuming that indexing begins with 0.

Below is C++ implementation of this solution.

```
// C program to check whether a given array
// represents a max-heap or not
#include <stdio.h>
#include <limits.h>

// Returns true if arr[i..n-1] represents a
// max-heap
bool isHeap(int arr[], int i, int n)
{
   // If a leaf node
   if (i > (n - 2)/2)
       return true;

   // If an internal node and is greater than its children, and
   // same is recursively true for the children
   if (arr[i] >= arr[2*i + 1]  &&  arr[i] >= arr[2*i + 2] &&
       isHeap(arr, 2*i + 1, n) && isHeap(arr, 2*i + 2, n))
       return true;

   return false;
}

// Driver program
int main()
{
    int arr[] = {90, 15, 10, 7, 12, 2, 7, 3};
    int n = sizeof(arr) / sizeof(int);

    isHeap(arr, 0, n)? printf("Yes"): printf("No");

    return 0;
}
```

Output:

```
Yes
```

Time complexity of this solution is O(n). The solution is similar to preorder traversal of Binary Tree.

Thanks to Utkarsh Trivedi for suggesting the above solution.

An **Iterative Solution** is to traverse all internal nodes and check id node is greater than its children or not.

```
// C program to check whether a given array
// represents a max-heap or not
#include <stdio.h>
#include <limits.h>

// Returns true if arr[i..n-1] represents a
// max-heap
bool isHeap(int arr[],  int n)
{
    // Start from root and go till the last internal
    // node
    for (int i=0; i<=(n-2)/2; i++)
    {
        // If left child is greater, return false
        if (arr[2*i +1] > arr[i])
                return false;

        // If right child is greater, return false
        if (arr[2*i+2] > arr[i])
                return false;
    }
    return true;
}

// Driver program
int main()
{
    int arr[] = {90, 15, 10, 7, 12, 2, 7, 3};
    int n = sizeof(arr) / sizeof(int);

    isHeap(arr, n)? printf("Yes"): printf("No");

    return 0;
}
```

Output:

```
Yes
```