**my\_len(A) T(n) =1 + n \* 1 + 1 = n+2**

**//Array A is passed**

**count ← 0 1**

**for each element in A n**

**count ← count + 1 1**

**return count 1**

**Algorithm HeapSort (A) (T(n)+6)\*(n/2)+2n+T(i)+8**

**//Array A is passed**

**build\_max\_heap(arr) (T(i)+6)\*(n/2)**

n my\_len(A) **n+2**

for i from n - 1 to 0 step -1: **n-1**

swap arr[i] with arr[0] **1**

**MaxHeap(arr, i, 0) T(i)+6**

**MaxHeap(arr, n, i )**

**//Array arr of size n and root node of index i is passed T(n)=T(n)+6**

largest i 1

left 2 \* i + 1 1

right 2 \* i + 2 1

if left < n and arr[largest] < arr[left]:

largest left 1

if right < n and arr[largest] < arr[right]:

largest right 1

if largest ! i:

swap arr[i] with arr[largest] 1

MaxHeap(arr, n, largest) T

**Build\_Max\_Heap(arr) (T(n)+6)\*(n/2)**

**//Array arr is passed**

**n length of arr 1**

for i from (n/2) - 1 to 0 step -1: (n/2)-1

**max\_heap(arr, n, I) T(n)+6**

**Algorithm isTriangular(A,n) (T(n)+6)\*(n/2)+3n+T(i)+10**

**HeapSort(nums) (T(n)+6)\*(n/2)+2n+T(i)+8**

for i from 0 to length of nums – 3: **n**

if nums[i] + nums[i + 1] > nums[i + 2] and nums[i + 1] + nums[i + 2] > nums[i] and nums[i] + nums[i + 2] > nums[i + 1]:

return 1 1

return 0 **1**