

Implementation and Analysis of Heap Sort

```
def heapify(arr, n, i):
    largest = i
    l = 2 * i + 1
    r = 2 * i + 2
    if l < n and arr[l] > arr[largest]:
        largest = l
    if r < n and arr[r] > arr[largest]:
        largest = r
    if largest != i:
        arr[i], arr[largest] = arr[largest], arr[i]
        heapify(arr, n, largest)

def heap_sort(arr):
    n = len(arr)
    for i in range(n // 2 - 1, -1, -1):
        heapify(arr, n, i)
    for i in range(n - 1, 0, -1):
        arr[i], arr[0] = arr[0], arr[i]
        heapify(arr, i, 0)

print("Enter an array")
a = list(map(int, input().split()))
heap_sort(a)
print("Sorted array is")
for i in range(len(a)):
    print(a[i], end=' ')
```

Input:

10 50 20 30 40 60

Output:

10 20 30 40 50 60

Time Complexity:

- $O(\log n)$

Program:

```
Heap Sort.py
1 def heapify(arr, n, i):
2     largest = i
3     l = 2 * i + 1
4     r = 2 * i + 2
5     if l < n and arr[l] > arr[largest]:
6         largest = l
7     if r < n and arr[r] > arr[largest]:
8         largest = r
9     if largest != i:
10        arr[i], arr[largest] = arr[largest], arr[i]
11        heapify(arr, n, largest)
12
13
14 def heap_sort(arr):
15     n = len(arr)
16     for i in range(n // 2 - 1, -1, -1):
17         heapify(arr, n, i)
18     for i in range(n - 1, 0, -1):
19         arr[i], arr[0] = arr[0], arr[i]
20         heapify(arr, i, 0)
21
22
23     print("Enter an array")
24     a = list(map(int, input().split()))
25     heap_sort(a)
26     print("Sorted array is")
27     for i in range(len(a)):
28         print(a[i], end=' ')
29
```

Input/Output:

```
Heap Sort
C:\Anaconda\envs\MLProject\python.exe "C:\MLProject\Heap_Sort.py"
Enter an array
10 20 30 40 50 60
Sorted array is
10 20 30 40 50 60
Process finished with exit code 0
```