Assignment 7

Sayak Ghorai || BT21GCS004 || B2 || Design and Analysis of Algorithm

Q: Write a code to sort an array using Heap Sort and analysis for different cases

Answer:

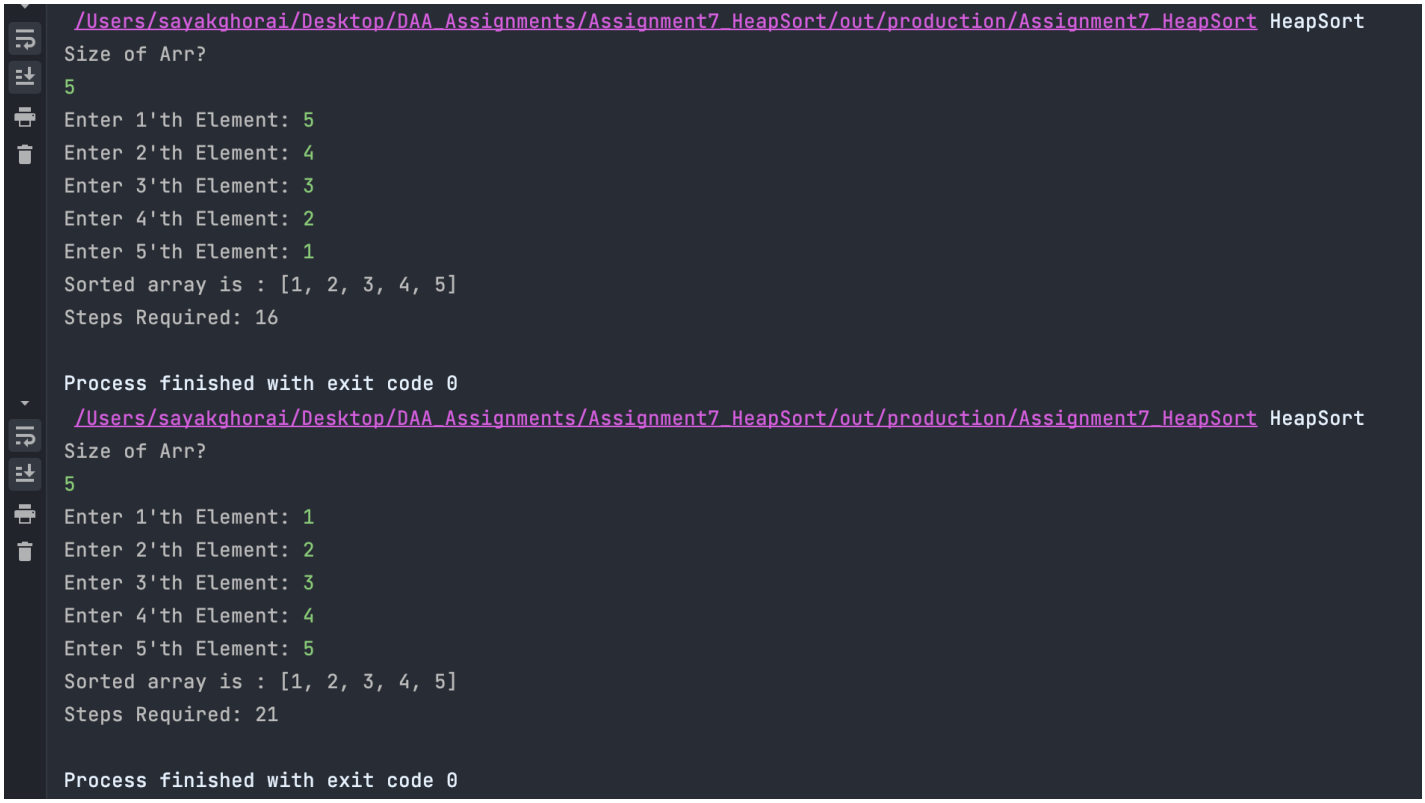
Heapsort Code:

HeapSort.java

```java
import java.util.*;

class HeapSort {
    6 usages
    static int Step;
    1 usage
    static int[] CreateArr() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Size of Arr?");
        int N = sc.nextInt();
        int[] Arr = new int[N];
        for (int i = 0; i < N; i++) {
            System.out.print("Enter " + (i+1) + "'th Element: ");
            Arr[i] = sc.nextInt();
        }
        return Arr;
    }
    3 usages
    static void Heapify(int[] arr, int n, int i){
        int largest=i;
        int left=2*i+1;
        int right=2*i+2;
        if(left<n && arr[left]>arr[largest]) {
            largest = left;
            Step++;
        }
        if (right<n && arr[right]>arr[largest]) {
            largest = right;
            Step++;
        }
        if (largest != i) {
            int swap=arr[i];
            arr[i]=arr[largest];
            arr[largest]=swap;
            Step++;
            Heapify(arr, n, largest);
        }
    }
    1 usage
    static void Heap_Sort(int[] arr){
        int n=arr.length;
        for(int i=n/2-1;i>=0;i--) {
            Step++;
            Heapify(arr, n, i);
        }
        for(int i=n-1;i>=0;i--){
            int temp=arr[0];
            arr[0]=arr[i];
            arr[i]=temp;
            Step++;
            Heapify(arr,i, i: 0);
        }
    }
    public static void main(String[] args){
        int[] arr=CreateArr();
        Heap_Sort(arr);
        System.out.println("Sorted array is : "+ Arrays.toString(arr));
        System.out.println("Steps Required: "+Step);
    }
}
```

Output:



Analysis: This sorting technique is using heapify( ) method that creates a Binary Almost complete tree and then converts the tree into Max Heap. So technically, for ascending order sorted array, it becames more complex to convert from min heap to max heap. But for descendening order sorted array, it takes effort only to check for max heap properties. It dosent have to swap any values. So heapify( ) consumes less cost for descending sorted sequence.

Time Complexity: Best case == Average Case == Worst Case == O(NxlogN) or logarithmic time complexity.
Space Complexity: O(1) or constant space complexity

Pros:

- Better optimised,efficient and accurate in terms of performance.
- Additional memory use is not there like Merge sort or Quick sort that usages recursion for execution.
- Can instantly return the smallest or largest element of a sequence

Cons:

- Less stable than other sorting techniques
- When it comes to highly complex data, it’s less efficient