# California State University, Fresno

# DEPARTMENT OF COMPUTER SCIENCE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Class: | **Algorithms & Data Structures** | | | Semester: | **Fall 2021** |
|  | | | | | |
| Points |  | Document author: | **Ojas Deshmukh** | | |
|  | Author’s email: | **ojas@mail.fresnostate.edu email** | | |
| Laboratory number: | **Lab 4** | | |
|  | | | | | |

**1. Statement of Objectives**

We are going to see the working and at the time complexity of the heap sort algorithm. We will build it as a class solution. We will give it input to add an element in the heap and ask the algorithm to sort and give the array including the newly added element. We will the also print min and max values in the heap and we will give the user the option to select between descending or ascending order of heap.

**2. Experimental Procedure**

A binary heap is a binary tree which is complete and has all the last nodes or levels filled in order from first to last. So, for example, we cannot fill the right parent in level two of any heap unless we fill the left parent completely. We will build heap as a class. This includes creating constructor, making the function to perform action in the class. We will need to fill all the functions mentioned in the templated design given to us for this experiment, for example, build heap, max heap, min heap, etc. Since we are building min and max heap let us take an overview of the working of both the styles.

Max heap- We implement max heap with the basic principle- we fill the root node with the maximum value in the array. Then on we make sure that every parent is filled with larger value than its child node. And first we fill the two child nodes of root then we fill node of left parent node and then the right, and so on. Then we create different functions to fill the heap with an extra element, delete an element, find max value, etc.

Min heap- We must fill the heap of tree based on, checking if the root node and parent nodes are smaller than their child nodes. If this does not follow, then it becomes a violation of the binary heap in min case. To remove a value, we must use decrease key and decrease the value at the index, Then, remove the root or shift and then heapify it back using the extract minimum function by following the main principle of min heap as mentioned above. Other operations include getting the min value and other functions.

For both the algorithms we will follow –

a[0] will return the root node of our binary heap, a[(i-1)/2] Returns the parent node, a[(2\*i)+1] Returns the left child node, a[(2\*i)+2] Returns the right child node. These are just formulae to implement the root parent and child nodes in the heap. Sorting is being carried out by both our heap algorithms in themselves.

**3. Analysis**

The average time complexity for the algorithms tends to be of order O(NlogN).

**4. Encountered Problems**

I had not understood the logic behind the max heap, it took some time to understand the concept and implement

**5. Conclusions**

It is a good algorithm with average time being of order NlogN. Coding for this tedious, I would personally try to avoid it.

**6. References**

List the references used in this report.