



Analysis of Algorithms I

Homework2

Student Name: Medina Zaganjori

Student Number: 150160908

Due date: 12.12.2019

Algorithm Explanation.

In this homework we were required to implement Min HeapSort Algorithm. Heap is considered as a binary tree which always has a heap structure where the levels are filled up, from left to right. Secondly, in our case it is ordered as a min heap. The idea is finding the minimum element and storing it at the end of an unsorted collection. The main steps are as follows:

1. We read some element and we immediately have a random unsorted array. Next we transform it into a heap. Our unsorted data will be taken as a parameter for the first function of buildMinHeap.
2. In our case the minimum value will be saved in the root of the heap. Every parent node will be smaller than its children. So we move the minimum value which is already located in the root node to the end of the heap by swapping it with last element.
3. Last node corresponds to minimum element. We will be moving down the root node item down to its correct place using our next function called heapify(). The algorithm will keep repeating until heap is down to just a single node. So basically it knows that all the elements in the unsorted array are in sorted positions and that last element node remaining will end up being first element in sorted array.

When we want to sort a list in ascending order we create Min Heap from that array and we pick the first element (like it is the smallest) and we keep repeating for the rest of elements. Building minHeap takes $O(n)$ time. We must then heapify all items in the heap except for the root node. This will take $O(\log n)$ time with $n-1$ calls to heapify. Since every single item must be added to heap and if we will be having a big amount of numbers means that our heap will be large. In order not to forget heap sort is a binary sort and based on the lecture slides the logarithmic time is $O(\log n)$ which means that acts very fast.

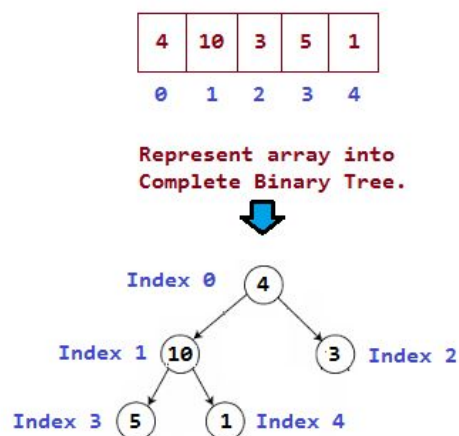
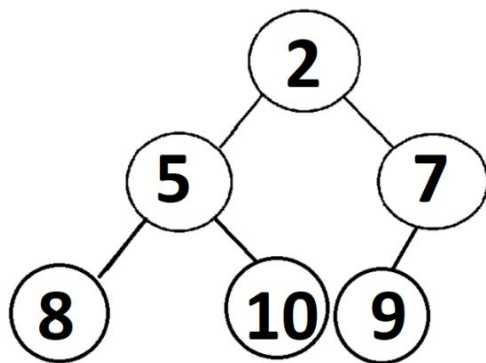


Figure1: Heapify Process

Heap sort will be running in linear time or in Big (O) notation is $O(\log n)$. Algorithm runs in an iteration mode and using comparison it swaps and uses heapify function. The disadvantage of this algorithm might be if we will be having elements which are duplicated and we can not rely on elements if the order they possess will be respected.



Tree Attributes:

- Root of the tree : first element in the array, corresponding to $i=1$
- $\text{parent}(i) = i/2$: returns index of parent node.
- $\text{left}(i) = 2i$: returns the index of node left child.
- $\text{right}(i) = 2i + 1$: returns index of node right child.

Figure2: Min Heap Tree

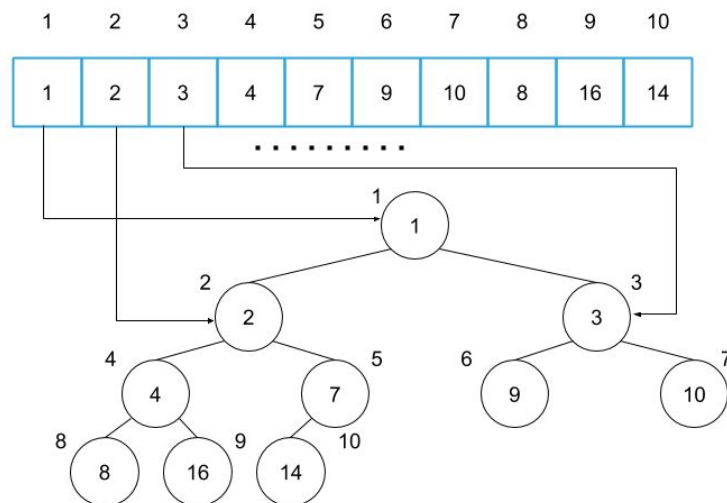
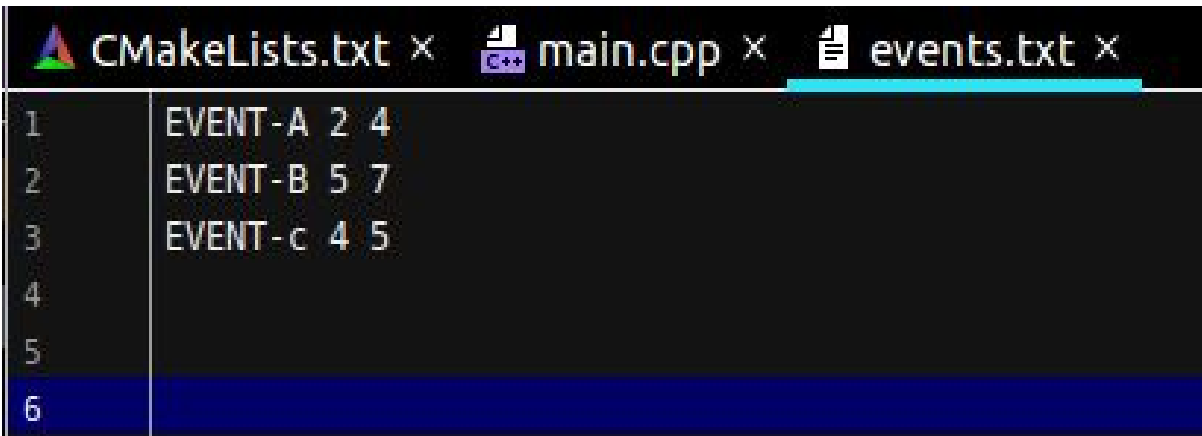


Figure3: Min HeapProcess.

In this homework i used a class called Event which will be holding the event type (EVENT-A, EVENT-B, EVENT-C...), start time which correspond to the values that second column holds or end time which corresponds to the third column in our reading file. In one object i saved 3 attributes, event name, event time and limit of time which is either END or START respectively, 2nd and 3rd column with integers. All these objects will be saved in a vector list and will be pushed iteratively. In our normal heap sort algorithm we have integers but in our case we will be sorting objects based on their time (start or end time) using the techniques explained in details above.

Next step is implementing a “virtual timer” or a counter that will be increasing iteratively and every time it will reach the value of one of end or start values of our objects will be printing a message as explained in the pdf.



1	EVENT-A 2 4
2	EVENT-B 5 7
3	EVENT-C 4 5
4	
5	
6	

```
medina@medina-Latitude-E7250:~/CLionProjects/homework2$ g++ main.cpp -o main
medina@medina-Latitude-E7250:~/CLionProjects/homework2$ ./main events.txt
TIME 1: NO EVENT
TIME 2: EVENT-A STARTED
TIME 3: NO EVENT
TIME 4: EVENT-A ENDED
TIME 5: EVENT-B STARTED
TIME 6: NO EVENT
TIME 7: EVENT-B ENDED
TIME 7: NO MORE EVENTS, SCHEDULER EXITS
medina@medina-Latitude-E7250:~/CLionProjects/homework2$
```

Figure4: Output Results.