Name: Afredo Perez

COSC 40403 - Analysis of Algorithms: Fall 2020: Homework 4

Due: 23:30 on 9/14

Some of the questions below require you to draw a heap or to trace through a heap algorithm on a specific array. You may do so using LaTeXpackages that you can research and learn how to use. You may also use a computer drawing program (i.e. PowerPoint or similar), create your diagrams, and export them as a PDF file (I find that it also helps to trim the PDF file using graphics software.) Then you can import the PDF file into your solution.

1. (5 points) What are the $n/2$ minimum and maximum number of elements in a heap of height h?

   At most, it is a perfect binary tree, which is $2^(h+1)1; atleast, thereisonlyoneatthelastlevel, whichis 2^h$.

2. (5 points) Show that an n-element heap has height lgn.

3. (5 points) Is the array with values $\langle 23, 17, 14, 6, 13, 10, 1, 5, 7, 12\rangle$ a max-heap? Show your work by using computer software to draw the heap.

   In a max-heap every node should be greater than its child, node with value of 6 is less than its child which is node with a value of 7. Hence it is not a max heap.

```
def extractMax(self):
  popped = self.Heap[self.FRONT]
  self.Heap[self.FRONT] = self.Heap[self.size]
  self.size -= 1
  self.maxHeapify(self.FRONT)
  return popped
```

4. (10 points) Illustrate (using computer software) the operation of $Max - Heapify(A, 3)$ on the array $A = \langle 27, 17, 3, 16, 13, 10, 1, 5, 7, 12, 4, 8, 9, 0\rangle$. Show how a heap element will trickle down the heap level-by-level.

   Illustration attached in file.

5. (5 points) Starting with the procedure $Max - Heapify(A, i)$, write pseudocode for the procedure $Min - Heapify(A, i)$, which performs the corresponding manipulation on a min-heap. How does the running time of $Min - Heapify$ compare to that of $Max - Heapify$?

```
MIN-HEAPIFY(A, i):
   l <- LEFT(i)
   r <- RIGHT(i)
   smallest <- i
   if l    heap-size[A] and A[l] < A[i]:
      then smallest <- l
   if r    heap-size[A] and A[r] < A[smallest]:
```

```
   then smallest <- r
if smallest =  i:
   then swap(A[i], A[smallest])
      MIN-HEAPIFY(A, smallest)
```

The running time is the same. Actually, the algorithm is the same with the exceptions of two comparisons and some names.

6. (10 points) Illustrate (using computer software) the operation of $Build - Max - Heap$ on the array $A = \langle 5, 3, 17, 10, 84, 19, 6, 22, 9 \rangle$. Each diagram should show the placement of an array element into its final position in the heap.

   Illustration attached in file

7. (5 points) Why do we want the loop index i in line 3 of $Build - Max - Heap$ to decrease from $A.length/2$ to 1 rather than increase from 1 to $A.length/2$?

   If we start from 1, because its subtree is not a maximum heap, we can't follow this order.

8. (10 points) Illustrate (using computer software) the operation of $Heapsort$ on the array $A = \langle 5, 13, 2, 25, 7, 17, 20, 8, 4 \rangle$. Show the original input array, the heap after $Build - Heap$, and then after each call to $Build - Max - Heap$.

   Illustration attached in file

9. (5 points) What is the running time of $Heapsort$ on an array A of length n that is already sorted in increasing order? What about decreasing order?

   For both ascending and descending order we have a run time of O(n lg n).