Name: Sakina Ghafoor
COSC 40403 - Analysis of Algorithms: Homework 3
Due: 11:59:59 on September

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, Visio, 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.

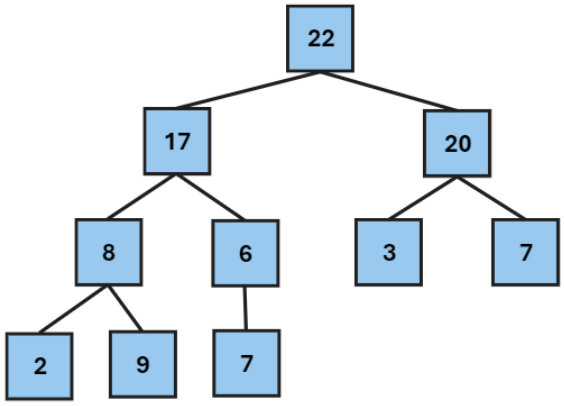
For all problems, you must show your work to receive partial credit.

1. (5 points) For the following array of values, use computer software to draw this array as a heap. Is this a max-heap? Explain perhaps by drawing a picture of this heap.

$$\langle 22, 17, 20, 8, 6, 3, 7, 2, 9, 7 \rangle$$

Answer:

No this is not a max-heap. This is because of the number 8, it is not larger than its child 9. Additionally, the value 6 is not larger than its child 7.



2. (10 points) Given the following heap, $A = \langle 99, 3, 10, 4, 5, 8, 7, 2, 1, 4 \rangle$, the value 3 is violating the max-heap property. Illustrate (by use of listing the elements of array A) the operation of MAX-HEAPIFY$(A, 3)$ on the array. Show how a heap element will trickle down the heap level-by-level. Note: for this problem just list the array/heap values. Do not show this as a tree. I am looking for several rows of array values.

Answer:

The MAX-HEAPIFY$(A, 3)$ starts with $A = \langle 99, 3, 10, 4, 5, 8, 7, 2, 1, 4 \rangle$.
Then, the function looks at value 3 and compares it with its children 4 and 5. It will use swap to move the largest of the three values to the parent position. $A = \langle 99, 5, 10, 4, 3, 8, 7, 2, 1, 4 \rangle$
Then, the function moves to the children of 3 which is 4. The child is larger than the parent so the two values swap positions. $A = \langle 99, 5, 10, 4, 4, 8, 7, 2, 1, 3 \rangle$
There is no more children left so the function terminates and we are left with a Max Heap. The value 3 no longer violates the max heap property.
We are left with the final array sequence: $A = \langle 99, 5, 10, 4, 4, 8, 7, 2, 1, 3 \rangle$

$A = \langle 99, 3, 10, 4, 5, 8, 7, 2, 1, 4 \rangle$
$A = \langle 99, 5, 10, 4, 3, 8, 7, 2, 1, 4 \rangle$
$A = \langle 99, 5, 10, 4, 4, 8, 7, 2, 1, 3 \rangle$

3. (10 points) Illustrate (by use of listing the elements of array A) the operation of BUILD-MAX-HEAP on the array $A = \langle 77, 48, 8, 8, 63, 31, 95, 53, 91, 52 \rangle$. Each listing of A should show the placement of an array element into its final position in the heap.

Answer:

The array is of length 10 and we start at value n/2 or value 63 at index 5 in the array $A = \langle 77, 48, 8, 8, 63, 31, 95, 53, 91, 52 \rangle$.
Then, for the children of 63 we have just 52 and there is no need for a swap since the parent is the largest. $A = \langle 77, 48, 8, 8, 63, 31, 95, 53, 91, 52 \rangle$
Then, the next value is the next index up which is value 8 at index 4. The children of value 8 would be 53 and 91. A swap is needed between 91 and 8. Now 8 at index 9 has no children so there is no need to go forward from here. $A = \langle 77, 48, 8, 91, 63, 31, 95, 53, 8, 52 \rangle$
The next value is 8 at index 3. Its children are 31 and 95, so a swap is needed between 8 and 95. Now 8 at index 7 has no children so there is no need to go forward from here. $A = \langle 77, 48, 95, 91, 63, 31, 8, 53, 8, 52 \rangle$
The next value is 48 at index 2 where the children are 91 and 63. A swap is needed between 48 and 91. $A = \langle 77, 91, 95, 48, 63, 31, 8, 53, 8, 52 \rangle$
The children of 48 at index 4 are now 53 and 8, so a swap is needed between 53 and 48. Now 48 at index 8 has no children so no need to go forward from here. $A = \langle 77, 91, 95, 53, 63, 31, 8, 48, 8, 52 \rangle$
The next value is 77 at index 1 where the children are 91 and 95. A swap is done between 95 and 77. The value 77 is now at index 3 with children 31 and 8, so no further action is needed here. $A = \langle 95, 91, 77, 53, 63, 31, 8, 48, 8, 52 \rangle$
The operation on the array is now complete and the array follows the max heap property.

$A = \langle 77, 48, 8, 8, 63, 31, 95, 53, 91, 52 \rangle$
$A = \langle 77, 48, 8, 8, 63, 31, 95, 53, 91, 52 \rangle$
$A = \langle 77, 48, 8, 91, 63, 31, 95, 53, 8, 52 \rangle$
$A = \langle 77, 48, 95, 91, 63, 31, 8, 53, 8, 52 \rangle$
$A = \langle 77, 91, 95, 48, 63, 31, 8, 53, 8, 52 \rangle$
$A = \langle 77, 91, 95, 53, 63, 31, 8, 48, 8, 52 \rangle$
$A = \langle 95, 91, 77, 53, 63, 31, 8, 48, 8, 52 \rangle$

4. (5 points) Consider an instance of an array in non-decreasing order and then passed to HEAPSORT. Does this instance have an effect on the running time? Explain. What if the array is in decreasing order? Explain.

Answer:

The order of the array has no effect on running time, the running time of Heap Sort is always $O(nlogn)$. This is because each element in the array is passed through the algorithm to ensure that it follows the heap property. Specifically, in the Build Max Heap method of Heap Sort, each input is sorted based on the value of its children making it $O(n)$. To make the heap, each element in the array is sorted from the last to first element making the running time for this portion linear. Then, Heap Sort swaps the root and the last element for a removal that follows the heap property with the Max Heapify method running in $O(logn)$ time. This adds up to a running time of $O(nlogn)$. The algorithm always looks for the largest value to ensure that the parent is at the right position in regards to its children, the max heap property. A certain order doesn't make the Heap Sort algorithm sort fewer elements, the same amount of elements are sorted for both types of arrays for validation.

5. (20 points) You are tasked with helping a botanist analyze the heights of trees in a forest. Given a list of tree heights (in feet), the botanist is interested in identifying the tallest trees. Specifically, they want to know the top k tallest trees in the forest. You may not sort the data by any means. You must use the `heapq` python library. Demonstrate your function works correctly using the two examples listed below.

   Write a function `top_k_tallest_trees(heights, k)` that takes the following inputs:

   - `heights`: A list of integers, where each integer represents the height of a tree in the forest.
   - `k`: An integer, representing the number of tallest trees to return.

   The function should return a list of the k tallest *unique* heights. If k is larger than the number of unique tree heights, return all the heights in descending order.

   Input:

   - The list `heights` will contain between 1 and 10,000 integers.
   - Each integer in the list will be between 1 and 100 (representing tree heights in feet).
   - `k` will be a positive integer and will not exceed the number of heights in the list.

   Output:

   - The function should return a list of integers representing the top k unique tallest tree heights in descending order.

   Example 1:

```
heights = [120, 130, 150, 160, 130, 140, 150, 170]
k = 3
top_k_tallest_trees(heights, k)  # Output: [170, 160, 150]
```

   Example 2:

```
heights = [120, 130, 120, 130]
k = 5
top_k_tallest_trees(heights, k)  # Output: [130, 120]
```

   Explanation:

   - In Example 1, the tallest unique tree heights are `[170, 160, 150, 140, 130, 120]`. The top 3 tallest heights are `[170, 160, 150]`.
   - In Example 2, there are only two unique tree heights `[130, 120]`. Since $k = 5$ is larger than the number of unique heights, the function returns all the unique heights in descending order.

   Additional Requirements:

   - You must implement the system using a heap (min-heap or max-heap). Use the `heapq` library.

- You may not sort the data at any stage of your solution.
- Submit your solution to this question as a python notebook (hw3q5.ipynb). Upload this file to TCU Online.

6. (20 points) You are tasked with building a basic emergency room triage system for a hospital. Patients arrive at the emergency room with varying levels of severity, and the hospital staff must treat patients in order of urgency. Patients with higher severity should be treated before those with lower severity, but patients with the same severity should be treated in the order they arrived.

   Write a python class `EmergencyRoom` that simulates the triage system. It should support the following operations:

   - `add_patient(name: str, severity: int)`: Adds a patient to the queue. Each patient has a unique name and a severity level. The severity is represented as an integer, where a higher number means more severe.
   - `treat_next() -> str`: Treats and returns the name of the patient with the highest severity. If multiple patients have the same severity, treat the one who arrived earlier.
   - `is_empty() -> bool`: Returns `True` if no patients are waiting to be treated, otherwise `False`.

   Input Constraints:

   - The severity level is an integer between 1 and 100, with 100 being the most severe.
   - Up to 100,000 patients may arrive.
   - Each patient has a unique name of up to 50 characters.

   Output:

   - The `treat_next` method should return the name of the next patient to be treated, according to the severity and arrival order.

   Example:

```python
# Example interaction
er = EmergencyRoom()

er.add_patient("Alice", 5)
er.add_patient("Bob", 8)
er.add_patient("Charlie", 8)
er.add_patient("David", 3)

print(er.treat_next())  # Output: "Bob" (highest severity)
print(er.treat_next())  # Output: "Charlie" (same severity as Bob, but arrived late
print(er.is_empty())    # Output: False
print(er.treat_next())  # Output: "Alice"
print(er.treat_next())  # Output: "David"
print(er.is_empty())    # Output: True
```

   Additional Requirements:

   - You must implement the system using a **priority queue** (min-heap or max-heap).

- The patients must be treated in order of severity, and if multiple patients have the same severity, they should be treated in the order they arrived.
- You may assume no two patients have the same name.
- Submit your solution to this question as a python notebook (hw3q6.ipynb). Upload this file to TCU Online.