Questions and/or Exercises to work out and turn in:

Grading Guidelines (See Appendix):

A right answer will get full credit when:

1. It is right (worth 25%)
2. It is right **AND** neatly presented making it easy and pleasant to read. (worth an **extra** 15%)
3. There is an **obvious and clear link** between 1) the information provided in the exercise and in class and 2) the final answer. A clear link is built by properly writing, justifying, and documenting an answer (worth an **extra** 60%).
4. Calculation mistakes will be minimally penalized (2 to 5% of full credit) while errors on units will be more heavily penalized.

You are welcome/encouraged to discuss exercises with other students or the instructor. But, ultimately, **personal** writing is expected.

* USE THIS FILE AS THE STARTING DOCUMENT YOU WILL TURN IN. **DO NOT DELETE ANYTHING FROM THIS FILE:** JUST **INSERT** YOUR ANSWERS.
* IF USING HAND WRITING (STRONGLY DISCOURAGED), **USE THIS FILE** BY CREATING SUFFICIENT SPACE AND WRITE IN YOUR ANSWERS.
* FAILING TO FOLLOW TURN IN DIRECTIONS /GUIDELINES WILL COST **A 30% PENALTY.**

Objectives of this assignment:

* to use and manipulate the concepts presented in this module
* to use and manipulate min-heap and max-heap concepts
* to reinforce the understanding of the heap structure
* to apply the heap concept to sorting

What you need to do:

Answer the questions and/or solve the exercises described below.

The objective of this assignment is to reinforce the understanding of the heap data structure.

Problem (100 points) Design Heapsort Using a Min-Heap

The objective of this exercise is to use a Min-Heap to implement Heapsort to sort an array A.

1. (10 points) Define what a **min-heap** is.

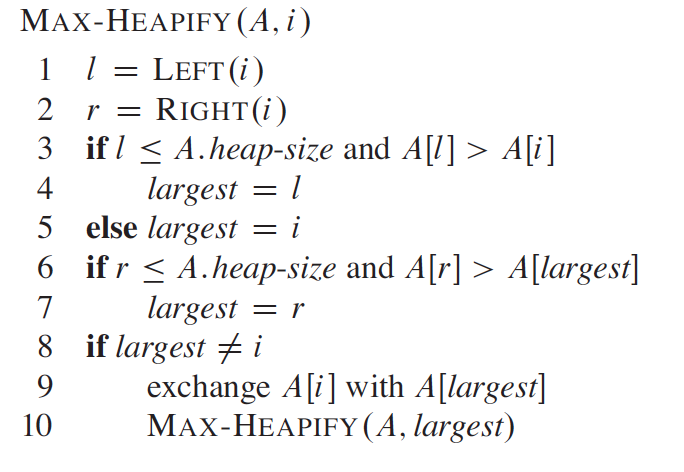
This will be defined as the opposite of what we had described in the video lectures about a max-heap. This means that a min-heap will still be a complete binary tree where the value of each node is less than or equal to the values of its children. This property must again be shown throughout the heap. This makes the smallest element of the heap at the root.

1. (10 points) Consider the array A = <42; 20; 28; 23; 25; 31; 36; 46; 28; 41>. **Draw** this as a heap and **explain** whether it is a min-heap or not. For your convenience, you may draw by hand the resulting heap, take a picture of your drawing, and insert it in this file. Just insure your drawing is neat and pleasant.

A diagram of a graph

Description automatically generated

From the graph I made above you can easily/immediately see from the root to it’s first children that this can not be a min-heap. The value 42 is the root of our array A and the values 20 and 29 are both less than the root, which should be the “minimum” value. This continues into the next level as well. Similarly, node 23 is smaller than its parent 20, which also violates the min-heap property. For this to be a min-heap, the elements would need to be rearranged so that the smallest element is at the root, and every parent node is smaller than its children.

1. (15 points) The following is the Max-Heapify(A,i) procedure. 

**Rewrite** Max-Heapify(A,i) into **Min-Heapify(A,i)** to help building a min-heap.

**Here is what the min-heap pseudocode would look using the above pseudocode for Max-Heapify:**

**Min-Heapify(A, i)**

**l = LEFT(i)**

**r = RIGHT(i)**

**smallest = i**

**if l ≤ A.heap-size and A[l] < A[i]**

**smallest = l**

**if r ≤ A.heap-size and A[r] < A[smallest]**

**smallest = r**

**if smallest ≠ i**

**exchange A[i] with A[smallest]**

**Min-Heapify(A, smallest)**

1. (25 points) **Execute** your procedure Min-Heapify(A,1) on the array A = <42; 20; 28; 23; 25; 31; 36; 46; 28; 41>. **Provide** what the array A becomes after the execution of each recursive Min-Heapify(A,i) and draw the resulting heap. You must provide the representation as an array **and** a heap. You may draw by hand the resulting heap, take a picture of your drawing, and insert it in this file. Just insure your drawing is neat and pleasant.

A=<42,20,28,23,25,31,36,46,28,41>

A=<20,42,28,23,25,31,36,46,28,41>

A=<20,23,28,42,25,31,36,46,28,41>

A=<20,23,28,28,25,31,36,46,42,41>

A= <20,23,28,28,25,31,36,46,42,41>

A diagram of a tree

Description automatically generated

1. (**40** points) **Analyze** the time complexity of your Min-Heapify(A,i) procedure.
2. (25 points) Let us **assume** that when a heap rooted at Node contains n nodes, then the subtrees rooted at or contain at most nodes where . Based on this assumption, derive the recurrence relation for Min-Heapify(A,i). Be as precise as Student 3 (see Appendix).

A binary heap is a complete binary tree, which means all levels are fully filled except possibly the last, which is filled from left to right. The min-heap property states that the value of each node is less than or equal to the values of its children. The total number of nodes n in a heap rooted at 𝑠 can be split among its child nodes. Typically, the left child will have roughly half of n, minus one node for the root itself. Since each call does a constant amount of work in terms of big O(1) and then recurses down the subtree. The relation can be represented as:

A mathematical equation with a number of letters

Description automatically generated with medium confidence

This recurrence will show how the time complexity propagates throughout the heap. This relation captures the idea that each recursive call works on a smaller subtree, which progressively reduces the problem size by a factor of a/b, plus some constant work for the comparison and then the potential swap involving each level.

1. (15 points) Assume that a = 2 and b = 3. Solve the recurrence relation you found in the previous question.



The Master Theorem helps in solving recurrences of the form:



a = 1 (the function makes a single recursive call)

b = 3/2

f(n) = Big O(1)



The complexity of Min-Heapify is dominated by the height of the tree because we potentially make one recursive call per level:



**Appendix**: Grading: What is an OBVIOUS and CLEAR LINK?

Here is an example to explain what an **obvious and clear link** is and how we grade your work.

Consider the following problem:

"(100 points) John travels from Auburn to Atlanta in his car at a speed of 60 mph. Leaving at 8am, at what time will John reach Atlanta".

Here are the answers of three students and their scores:

* **Student 1** answers: "9:48am". Student 1 will get 25 points.
* **Student 2**answers : "John will reach Atlanta at 9:48am". Student 2 will get 25+15 = 40 points
* **Student 3** answers: "The time t to travel a distance d at speed v is equal to d/v = d/60mph. The problem does not provide the distance d from Auburn to Atlanta. Based on GoogleMaps, the distance from Auburn to Atlanta is approximately 108 miles (**document is attached**).



Therefore, the time t = 108 miles/60mph \* 60 minutes/hour= 108 minutes. Since John left at 8am, he will then reach Atlanta at 8am + 108 minutes = 8 am + 60 minutes + 48 minutes = 9:48".

**Student 3** will get 25 + 15 + 60 = 100 points

Do you see the **direct** **link** going from the data provided in the question to the final answer, using general knowledge/formula and documents?.... Can you now solve the following problem and get 100 points?

"(100 points) Alice travels from Auburn to Atlanta in her car at a speed of 60 mph. Leaving at 8am, at what time will Alice reach Atlanta assuming that she had a flat tire that delayed her 30 minutes".