Northeastern University
CS5800 – Algorithms
Fall 2023 Semester, Jonathan Mwaura

Problem Set #5

This Problem Set is due at **11:30PM on Monday Oct** 23^{rd}, and will be submitted on GRADESCOPE.

This Problem Set will be marked out of 35. The first one is worth 15 points and the second one is worth 20 points

Please type (or neatly handwrite) your solutions on standard 8.5×11 paper, with your name at the top of each solution. Ensure that you submit you solutions in one file PDF file on Gradescope. **each problem sets solution should be on in its own individual page, Gradescope will help ensure you submit each solution under its correct problem number**

While a solution must be absolutely perfect to receive full marks, I will be generous in awarding partial marks for incomplete solutions that demonstrate progress.

So that there is no ambiguity, there are two non-negotiable rules. A violation of either rule constitutes plagiarism and will result in you receiving an F for this course.

(a) If you meet with a classmate to discuss any of the Individual Problems, your submission must be an individual activity, done in your own words, away from others. The process of finding a solution might take 3 - 5 iterations or even more BUT you learn from all these attempts and your confidence grows with each iteration.

(b) These problem sets might seem hard on a first look. They are designed to be so. We learn by attempting problems, struggling through them and coming on top. I encourage you to make this learning exercise worth your while. What do I mean? Open the problem sets as early as you get them, then do not look at hints or answers any where (including on the internet and consulting other students for direct answers), give it the best shot you can. If you get stuck come to Professor or TA's office hour and we shall be glad to listen to your rationale and work with you till you are able to tackle the problem sets.

Problem #1

A complete binary tree of height h has "no holes": i.e reading from top-bottom and left-to-right, every node exists. An almost complete binary tree has every node until the last row, which is allowed to stop early.

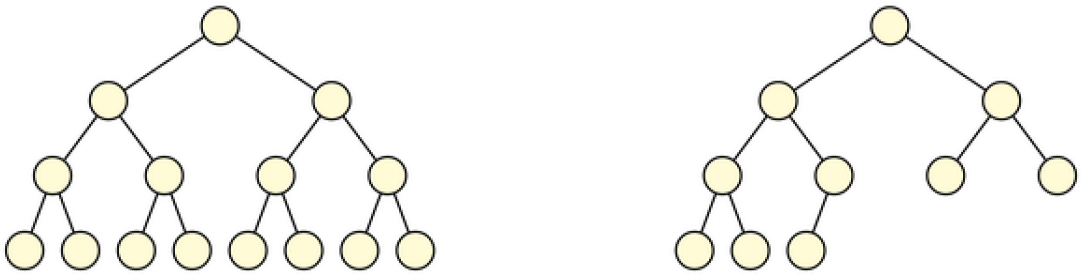

Figure 1: complete (perfect) binary tree (left) and almost complete binary tree (right)

(a) Prove by mathematical induction that a complete binary tree with height, h, contains precisely $2^{h+1} - 1$ nodes

(b) How many leaves does an almost complete binary tree of height, h, have? Give the smallest and largest possible values, and explain. Note, by definition every complete binary tree is almost complete tree.

(c) The diameter of a tree or a graph is the maximum distance (length of the longest path) between nodes. Whats the diameter of an almost complete binary tree of height, h. Give the smallest and largest possible values and explain

(d) Suppose that we "reroot" a complete binary tree of height, h, by designating one of the erstwhile leaves as the root. What is the height of the rerooted tree?

(e) What is the diameter of a complete binary tree rerooted at one of its leaves?

Problem #2

A binary heap is called a *max heap* if it has the property that for every node i other than the root, the value of the node is at most the value of its parent.

Below is an example of a max heap with 10 nodes (i.e., 10 elements), presented both as a binary tree and as an array.

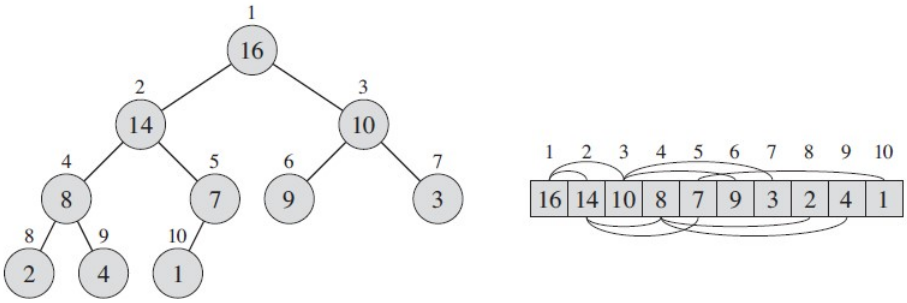

Figure 2: Max Heap with 10 nodes

(a) The *height* of a heap is defined to the number of edges on the longest downward path from the root node to a leaf node. Thus, in the example above, the height of the heap is $h = 3$.

If a (binary) heap has height $h = 6$, determine the minimum number and maximum number of elements that can be in this heap. Clearly justify your answer.

(b) Consider an unsorted array of n elements. Recall that the Heapsort Algorithm consists of two parts: first we run BUILD-MAX-HEAP to convert our input array into a max heap, and then we run MAX-HEAPIFY n times to generate the n elements of our sorted array.

Demonstrate the Heapsort Algorithm on the input array $[5, 2, 1, 7, 6, 3, 4]$. Clearly show your steps.

(c) In part (b) above, you should have noticed that after the $i = 2$ iteration of MAX-HEAPIFY, your heap was $[5, 3, 4, 2, 1, \mathbf{6}, \mathbf{7}]$. Notice how the first $n - i$ elements form a max heap, and the last i elements are sorted and are the i *largest* elements of the array.

Show that this property holds for *any* max heap with n elements. Specifically, prove that for all $1 \leq i \leq n$, after the i^{th} iteration of MAX-HEAPIFY, the first $n - i$ elements form a max heap, and the last i elements are sorted and are the i *largest* elements of the array.

(d) Let P be a permutation of the first 7 positive integers. Sometimes this permutation is a max heap; examples include $[7, 6, 5, 4, 3, 2, 1]$, $[7, 6, 4, 2, 5, 1, 3]$, $[7, 5, 6, 2, 4, 3, 1]$, and $[7, 3, 6, 2, 1, 4, 5]$.

If P is a randomly-chosen permutation of $[1, 2, 3, 4, 5, 6, 7]$, determine the probability that it is a max heap. Clearly and carefully justify your answer.