Northeastern University
CS5800 – Algorithms
Spring 2024 Semester, Jonathan Mwaura

Problem Set #4

This Problem Set is due at **11:30PM on Monday Feb** 26th, and will be submitted on GRADESCOPE. Note, any source codes required should be submitted in Canvas.

This Problem Set will be marked out of 40. There are three problems.

Suggested working schedule:

- Problem 1 - Monday 2/19/24
- Problem 2 - Wed 2/21/24
- Problem 3 - Friday 2/23/24
- Review and Submission - 2/24/24

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.

This work must be submitted on Gradescope on or before the due date. The late due date can be used with no penalty. Note that the system wont allow for work to be submitted after the late due date. Any work, not submitted will be awarded zero points. The instructor does not accept emailed work.

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 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.

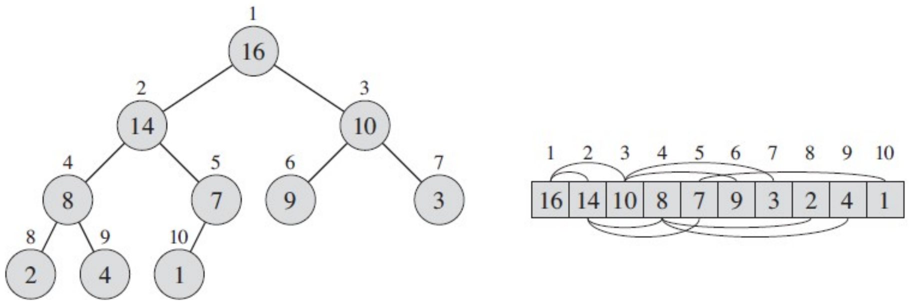


Figure 1: Max Heap with 10 nodes

(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 $[7, 3, 6, 2, 1, 4, 5]$.

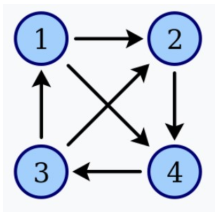
If P is a randomly-chosen permutation of $[1, 2, 3, 4, 5, 6, 7]$, determine the probability that it is a max heap. Clearly and carefully justify your answer.

Problem #2

Some CS5800 students are very competitive. So one day after class, n players decided to organize a Table Tennis tournament, where each player had one game against every other player.

The results of each tournament can be represented using a directed *tournament graph*, where we draw an edge from vertex i to vertex j if and only if player j beat player i in the tournament.

For example, consider this tournament graph with $n = 4$ players.



From the above graph, we see that Player 1 won one game (against Player 3) and lost two games (against Players 2 and 4).

For each player i, let w_i be the number of games they won. This number is identical to the *in-degree* of vertex i in the tournament graph. So in the above tournament, we have $(w_1, w_2, w_3, w_4) = (1, 2, 1, 2)$.

Convince yourself that in every tournament graph, we must have $w_1 + w_2 + \ldots + w_n = \binom{n}{2} = \frac{n(n-1)}{2}$.

(a) Let $n = 7$. Construct a tournament graph for which $(w_1, w_2, w_3, w_4, w_5, w_6, w_7) = (6, 5, 4, 3, 2, 1, 0)$. Briefly explain how you constructed your graph.

(b) Let $n = 7$. Construct a tournament graph for which $(w_1, w_2, w_3, w_4, w_5, w_6, w_7) = (3, 3, 3, 3, 3, 3, 3)$. Briefly explain how you constructed your graph.

(c) Consider a tournament with n players.

Determine all integers $n \geq 2$ for which there exists a tournament graph for which $(w_1, w_2, \ldots, w_n) = (k, k, \ldots, k)$ for some integer k. In other words, such a tournament has the property that every player wins exactly the same number of games.

Problem #3

In this question, you will create a **mini-portfolio** consisting of any **two** LeetCode problems on Graphs, chosen from the four problems below:

`https://leetcode.com/problems/clone-graph/`
`https://leetcode.com/problems/is-graph-bipartite/`
`https://leetcode.com/problems/find-the-town-judge/`
`https://leetcode.com/problems/find-if-path-exists-in-graph/`

As always, you may code your algorithms in the programming language of your choice.

Here is how your mini-portfolio will be graded.

There will be a total of 10 points for any of the combination of problems in your mini-portfolio: For each of these, provide the problem number, problem title, difficulty level, and the screenshot of you getting your solution accepted by LeetCode (10 points).

Note that you are allowed to work with Teammates on this part of the problem. Make sure you write all names of the collaborators.