CSCI 3104 Fall 2022
Instructors: Prof. Grochow and Chandra Kanth Nagesh

Problem Set 11

Due Date ..... **Tuesday** November 29, 2022
Name ..... **Aidan Reese**
Student ID **108418975**
Collaborators .....

Contents

Instructions

- The solutions **must be typed**, using proper mathematical notation. We cannot accept hand-written solutions. Useful links and references on LaTeXcan be found here on Canvas.
- You should submit your work through the **class Canvas page** only. Please submit one PDF file, compiled using this LaTeX template.
- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please submit this document with no fewer pages than the blank template (or Gradescope has issues with it).
- You are welcome and encouraged to collaborate with your classmates, as well as consult outside resources. You must **cite your sources in this document. Copying from any source is an Honor Code violation. Furthermore, all submissions must be in your own words and reflect your understanding of the material.** If there is any confusion about this policy, it is your responsibility to clarify before the due date.
- Posting to **any** service including, but not limited to Chegg, Reddit, StackExchange, etc., for help on an assignment is a violation of the Honor Code.
- You **must** virtually sign the Honor Code (see Section Honor Code). Failure to do so will result in your assignment not being graded.

Honor Code (Make Sure to Virtually Sign the Honor Pledge)

Problem HC. On my honor, my submission reflects the following:

- My submission is in my own words and reflects my understanding of the material.
- Any collaborations and external sources have been clearly cited in this document.

- I have not posted to external services including, but not limited to Chegg, Reddit, StackExchange, etc.
- I have neither copied nor provided others solutions they can copy.

In the specified region below, clearly indicate that you have upheld the Honor Code. Then type your name.

Honor Code. Aidan Reese I agree to the above. □

25 Standard 25: Design a Dynamic Programming Algorithm (Synthesis Standard)

Problem 25. The Placing Electrons on a Tree problem is defined as follows.

- Instance: A tree T, with root $r \in V(T)$.
- Solution: A subset $S \subseteq V(T)$ such that there are no edges (s, s') for $s, s' \in S$, and such that $|S|$ is as large as possible.

The *idea* is that you are placing electrons on the vertices of a tree, but electrons repel each other, so you cannot place two electrons adjacent to one another, with the goal being to place as many electrons as possible.

The goal of this problem set is to design a dynamic programming algorithm to solve the Placing Electrons on a Tree problem.

Problem 25??

(a) For each vertex $v \in V(T)$, let $L[v]$ denote the maximum number of electrons that can be placed on the subtree rooted at v. Write down a mathematical recurrence for $L[v]$. Clearly justify each case. *Hint:* your recurrence should involve the children and grandchildren of v. You may use notation such as "children(v)" to denote the set of children of v, and "grandchildren(v)" to denote the set of grandchildren of v. Remember to have base case(s).

Answer. $\mathrm{L[v]} = \begin{cases} \max(\sum_{v \in \mathrm{child(v)}} L[v], 1 + \sum_{v \in \mathrm{grand(v)}} L[v]) & \mathrm{child(v)} \neq 1 \\ 1 & \mathrm{child(v)} = 0 \end{cases}$

Case 1 : Case where we count v and do not use any child vertices towards the count of electrons. $1 + \sum_v$

Problem 25(b)

(b) Clearly describe how to construct and fill in the lookup table. For the cell $L[v]$, clearly describe the sub-cases we consider, and which optimal sub-case we select. *Hint:* Similar to counting paths in a DAG, what order should you put the vertices in so that you can fill in the lookup table "in order"?

Answer. □

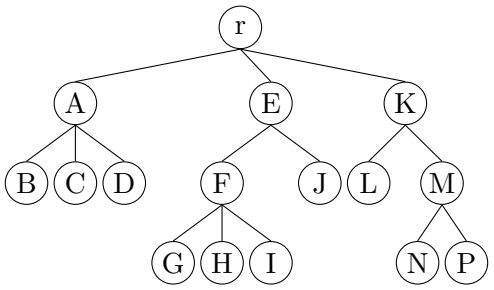
Problem 25(c)

(c) Let `back`$[v]$ denote another array storing "backpointer" information to enable you to reconstruct the optimal solution. Clearly describe what information should be stored in `back`[v] and how it is generated based on the optimal sub-cases selected in the computation of $L[v]$ (as you described in Problem2**??** above).

Answer. □

Problem 25(d)

(d) Work through an example of your algorithm using the following tree. Clearly show how to recover an optimal solution using your lookup table. You may hand-draw your lookup table, but your explanation must be typed.



Answer. □

27 Standard 27: Amortized Analysis & Doubling Lists

Problem 27. *Note: this is your second attempt on this standard—the first was on Midterm 2. You will also see it on a quiz and on the final. Because we pushed back the schedule on S27, despite having four chances at it, you only have to demonstrate proficiency once to get Standard 27 to count.*

A dynamic array-list is a data structure that can support an arbitrary number of append (add to the end) operations by allocating additional memory when the array becomes full. The standard process covered in class is to double (adds n more space) the size of the array each time it becomes full. You cannot assume that this additional space is available in the same block of memory as the original array, so the dynamic array must be copied into a new array of larger size. Here we consider what happens when we modify this process. The operations that the dynamic array supports are

- `Indexing A[i]`: returns the `i`-th element in the array
- `Append(A,x)`: appends `x` to the end of the array. If the array had `n` elements in it (and we are using 0-based indexing), then after `Append(A, x)`, we have that `A[n]` is `x`.

27(a)

a. Consider a dynamic array-list that adds 7 to its length (that is, increases length from n to $n+7$) each time it's full. Derive the amortized runtime of Append for this data structure. Clearly explain the reasoning behind your calculations.

Answer. □

27(b)

b. Derive the amortized runtime of Append for a dynamic array that squares its length (that is, increases length from n to n^2) each time it's full. Derive the amortized runtime of Append for this data structure. Clearly explain the reasoning behind your calculations.

Answer. □