CSCI 3104 Fall 2021
Instructors: Profs. Grochow and Waggoner

Problem Set 11

Due Date ..... TODO
Name ..... **John Blackburn**
Student ID **Jobl2177**
Collaborators ..... **List Your Collaborators Here**

Contents

1 Instructions

- The solutions **must be typed**, using proper mathematical notation. We cannot accept hand-written solutions. Here's a short intro to LaTeX.
- 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 2). Failure to do so will result in your assignment not being graded.

2 Honor Code (Make Sure to Virtually Sign)

Problem 1.

- 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.

Agreed (john blackburn). □

3 Standard 24- Dynamic Programming: Design DP Algorithms

Problem 2. The Placing Electrons on a Tree problem is defined as follows.

- Instance: A tree rooted 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.

3.1 Problem 2(a)

(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.

$$L[v] = \begin{cases} 1 & \text{v is a leaf node} \\ \text{number of kids(v)} & \text{v has kid nodes and no grandkid nodes} \\ \text{1+ L[kids(v)]} & \text{v has kid nodes and has grandkid nodes and } L[kids(v)] = L[grandkids(v)] \\ \text{L[kids(v)]} & \text{v has kid nodes and has grandkid nodes and } L[kids(v)] > L[grandkids(v)] \end{cases}$$

□

3.2 Problem 2(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. □

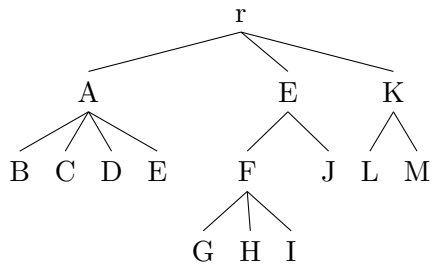
3.3 Problem 2(c)

(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(b) above).

Answer. □

3.4 Problem 2(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. □