CSC373 Fall'20
Midterm 1

Due By: 20th October 2020, 6:10pm ET

Instructions

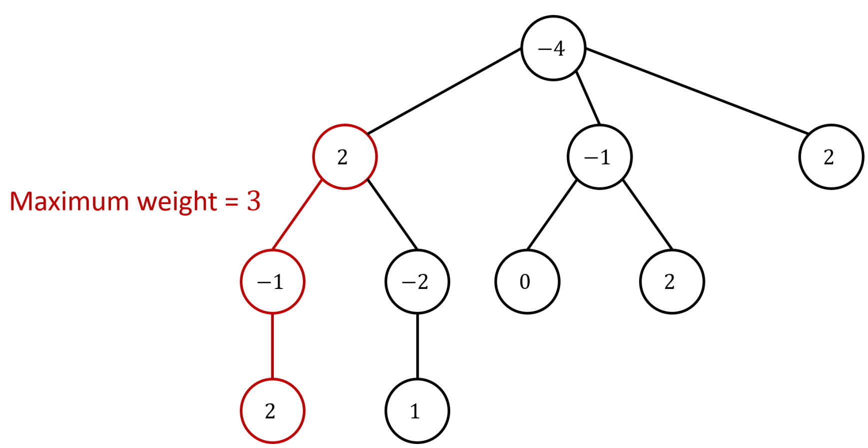
1. Upload a single PDF with your solutions to MarkUs at https://markus.teach.cs.toronto.edu/csc373-2020-09
2. For handwritten solutions, please make sure that your handwriting is legible, and your scan is high-quality and not distorted.
3. Remember: You will receive 20% for any (sub)question when you leave it blank (or cross off any written solution) and write "I do not know how to approach this problem.", and 10% if you leave it blank but do not write this or a similar statement.

Q1 [20 Points] The Best Part of a Tree

You are given a *rooted tree* T with n nodes. Each node v has a weight $w_v \in \mathbb{R}$. Your goal is to find the maximum total node weight in any *connected subgraph* of T. (You do not need to find the actual subgraph inducing this weight.)

A subgraph H of T consists of a subset of nodes in T, and it is connected if every pair of nodes in H has a path connecting them in T. Empty subgraph is allowed and has total node weight 0.

Here is an example tree with its maximum-weight connected subgraph highlighted in red.



Design a divide-and-conquer algorithm for the problem. Describe your algorithm (either verbally or through a pseudocode) and provide a brief justification of its correctness and worst-case running time. For full credit, your algorithm must run in $O(n)$ time.

[Hint: Remember that a tree with n nodes has $m = n - 1$ edges.]

Q2 [20 Points] Protect the Paintings

A corridor of a museum is represented by the interval $[a, b]$ (with $a < b$) and contains valuable paintings. There are n guards stationed along the corridor. Guard i can protect the interval $[s_i, f_i]$, where $a \leq s_i \leq f_i \leq b$. Your goal is to find a minimum-size subset of guards $P \subseteq \{1, \ldots, n\}$ who collectively cover the entire corridor (i.e. $\cup_{i \in P}[s_i, f_i] = [a, b]$); assume that such a subset exists.

Consider the following greedy algorithm, which always looks at the rightmost covered point (initially a) and, among the guards who cover it, selects one whose range goes as much to the right as possible.

- $G \leftarrow \emptyset$ and $t \leftarrow a$
- While $t < b$:
  - $S_t \leftarrow$ set of guards who cover point t, i.e., $S_t = \{i : t \in [s_i, f_i]\}$
  - $i^* \leftarrow$ guard in S_t with the greatest endpoint, i.e., so that $f_{i^*} \geq f_i$ for all $i \in S_t$
  - $G \leftarrow G \cup \{i^*\}$ and $t \leftarrow f_{i^*}$
- Return G

(a) [15 Points] Prove that this algorithm will always produce an optimal solution.

(b) [5 Points] Briefly describe how you will implement this algorithm in $O(n \log n)$ time. (You do not need to write a formal pseudocode.)

Q3 [10 Points] Friendly Committee [Difficult]

There are n uniformly spaced houses along a straight road. Each house i has a head member whose is age is $A[i]$. For some unexplainable reason, heads of houses i and j are enemies whenever their age difference exceeds the distance between their houses, i.e., whenever $|A[i] - A[j]| > |i - j|$.

You want to form a friendly committee by choosing heads from a subset of houses such that *no two heads chosen are enemies*. Design a dynamic programming algorithm that takes array $A[1 \ldots n]$ as input and computes the maximum number of heads the committee can have. For full credit, your algorithm must run in $O(n^2)$ time and use $O(n)$ space.

(a) [2.5 Points] Define an array storing the necessary information from subproblems. Clearly define what each entry means and how you would compute the desired solution given this array.

(b) [2.5 Points] Write a Bellman equation and briefly justify its correctness.

(c) [2.5 Points] In what order would you compute the entries in a bottom-up implementation? (Pseudocode is *not* needed.)

(d) [2.5 Points] Analyze the worst-case running time and space complexity of your algorithm.