York University EECS 4101/5101 May 19, 2020

# Homework Assignment #1
# Due: May 26, 2020 at 11:30 a.m.

**1.** Several classical algorithms construct a tree: Dijkstra's algorithm, Prim's algorithm, bread-first search, depth-first search. They each build the tree by starting with a root node and adding nodes to the tree one by one. Here we assume that each node has a unique label that is a natural number. An up-tree ADT can be used by each of these algorithms to represent the tree that is constructed. Here is a formal specification of an up-tree ADT.

$$\begin{aligned} Q &= \{\langle V, \mathit{root}, \pi\rangle : V \text{ is a finite subset of } \mathbb{N} \text{ and } \mathit{root} \in V \text{ and } \pi : (V - \{\mathit{root}\}) \to V\} \\ \mathit{OPS} &= \{\textsc{AddNode}(v,p) : v, p \in \mathbb{N}\} \cup \{\textsc{Parent}(v) : v \in \mathbb{N}\} \\ \mathit{RES} &= \{\textsc{Ack}, \textsc{Error}\} \cup \mathbb{N} \end{aligned}$$

We define the transition function for all states $\langle V, \mathit{root}, \pi\rangle \in Q$ as follows.
If $v \notin V$ and $p \in V$, then
$\delta(\langle V, \mathit{root}, \pi\rangle, \textsc{AddNode}(v,p)) = (\textsc{Ack}, \langle V \cup \{v\}, \mathit{root}, \pi'\rangle)$, where $\pi'(v) = p$ and $\pi'(u) = \pi(u)$ for all $u \in V$.
If $v \in V$ or $p \notin V$, then $\delta(\langle V, \mathit{root}, \pi\rangle, \textsc{AddNode}(v,p)) = (\textsc{Error}, \langle V, \mathit{root}, \pi\rangle)$.
If $v \in V - \{\mathit{root}\}$, then $\delta(\langle V, \mathit{root}, \pi\rangle, \textsc{Parent}(v)) = (\pi(v), \langle V, \mathit{root}, \pi\rangle)$.
If $v \notin V - \{\mathit{root}\}$, then $\delta(\langle V, \mathit{root}, \pi\rangle, \textsc{Parent}(v)) = (\textsc{Error}, \langle V, \mathit{root}, \pi\rangle)$.

**(a)** Define a reasonable measure $n$ of the size of the state of this ADT.

**(b)** Now, assume that all the node labels are from the set $\{1, 2, \ldots, N\}$. Give a *simple* data structure that implements the up-tree ADT. Be explicit about how you represent the information stored by your data structure in memory, and give algorithms for implementing AddNode and Parent. In designing your data structure, you should aim for *fast* operations, even if this means using a lot of space.

**(c)** What is the worst-case running time of your operations in (b)? State your answer using $\Theta$ notation.

**(d)** How many words of memory does your data structure in part (b) use? You can assume that a node label or a pointer can fit into a single memory word. State your answer using $\Theta$ notation.

**(e)** **For EECS5101 students only:** Assume that the ADT is initialized to a state of the form $\langle \{\mathit{root}\}, \mathit{root}, \pi_0\rangle$. Then, any sequence of AddNode operations is performed, leaving the ADT in state $\langle V, \mathit{root}, \pi\rangle$. Then, for some $v \in V$, the following code is executed.

```
while $v \neq root$
    $v \leftarrow$ Parent($v$)
end while
```

Prove that this loop terminates.

**2.** Suppose you are writing a programme to keep track of the locations of cellular towers across Canada. Assume each tower has a unique serial number. You would like to be able to add towers to the set (when a new tower is built) and remove a tower from the set (when some idiot burns the tower down because he thinks it causes covid-19). You would also like to answer the following type of queries which are useful for routing transmissions to customers: given the location of a cellular telephone, which cell tower is the closest to that telephone's location? Give the formal specification of an ADT for this task. Justify any design decisions that you make. Is your ADT deterministic or non-deterministic?

Note that this question is *not* asking you to develop a data structure to implement the ADT.