# CSC 225 - SUMMER 2019
# ALGORITHMS AND DATA STRUCTURES I
# WRITTEN ASSIGNMENT 3
# UNIVERSITY OF VICTORIA

**Due**: Wednesday, July 3rd, 2019 before noon. **Late assignments will not be accepted.**

**Submit your answers on paper to the CSC 225 drop box on the second floor of ECS (in front of the elevators). You are expected to submit a typed solution (handwritten documents will not be marked). However, you are permitted to draw mathematical formulas or diagrams onto the typed copy by hand if that is more convenient than typesetting them.**

**Question 1**: Radix Sort [5 marks]
Describe an algorithm which takes a sequence $S$, containing $n$ integers in the range $[0, n^2 - 1]$, and sorts $S$ in $O(n)$ time. You should not provide pseudocode (an English description of the algorithm is sufficient), but you must justify the running time of your algorithm (in one or two sentences).

**Question 2**: The Master Theorem [8 marks]
Use the Master Theorem (as stated in the lecture slides) to find a simple Big-Theta expression for the recurrences below. Clearly indicate which case of the Theorem applies and what the values $a, b, c$ and $g(n)$ are for each recurrence. In the event that case 3 of the theorem applies, remember to clearly state whether the extra condition $ag(n/b) < \delta g(n)$ is satisfied.

(a) (Assume $n$ is a power of 2)

$$T(n) = \begin{cases} 6 & \text{if } n = 1 \\ 8T(n/2) + (n \log_2 n)^3 & \text{if } n > 1 \end{cases}$$

(b) (Assume $n$ is a power of 2)

$$T(n) = \begin{cases} 10 & \text{if } n = 1 \\ 3T(n/2) + 4n & \text{if } n > 1 \end{cases}$$

(c) (Assume $n$ is a power of 2)

$$T(n) = \begin{cases} 17 & \text{if } n = 1 \\ 4T(n/2) + n^4 \log_2(n) & \text{if } n > 1 \end{cases}$$

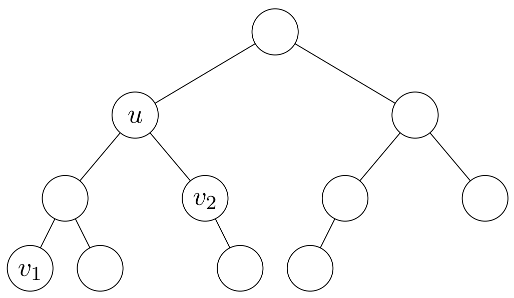
(d) (Assume $n$ is a power of 3)

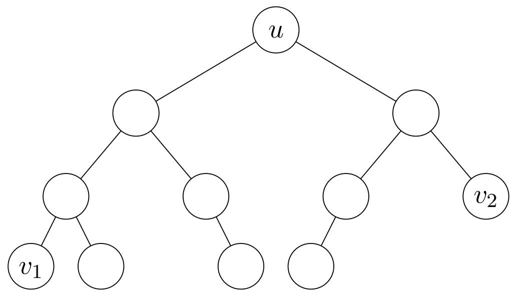
$$T(n) = \begin{cases} 187 & \text{if } n = 1 \\ 9T(n/3) + n & \text{if } n > 1 \end{cases}$$

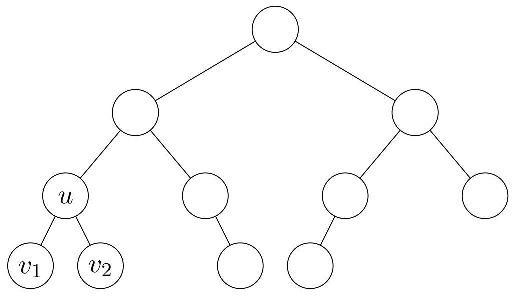
**Question 3**: Trees [5 marks]
Let $T$ be a binary tree with height $h$. Recall that a node $u$ is an *ancestor* of a node $v$ if the path from $v$ up to the root of $T$ includes $u$. Note that any node $v$ is considered an ancestor of itself.

For two nodes $v_1$ and $v_2$, the *lowest common ancestor* is the lowest (that is, furthest from the root) node $u$ such that $u$ is an ancestor of both $v_1$ and $v_2$.

The diagrams below show the lowest common ancestor (denoted by $u$) of several $v_1, v_2$ pairs.



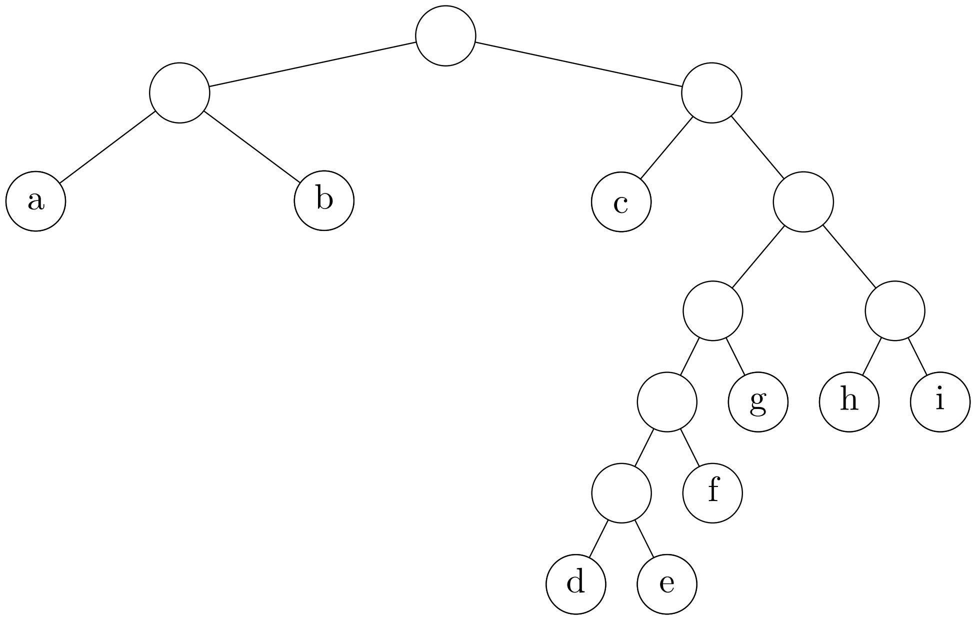




Give pseudocode for a $O(h)$ algorithm to find and return the lowest common ancestor of two nodes $v_1$ and $v_2$. Use the notation Left(v), Right(v) and Parent(v) to refer to the left child, right child and parent (respectively) of a node $v$. Include a brief justification for the running time of your solution.

**Question 4**: Priority Queues [5 marks]
Prove that it is impossible to implement a priority queue data structure such that both of RemoveMin and Insert require $O(\log_2 \log_2 n)$ time on a priority queue containing $n$ elements.

**Question 5**: Huffman Coding [5 marks]
For this question, assume that the Huffman Coding algorithm, when merging two trees, always places the tree with higher frequency on the left and the tree with lower frequency on the right. Consider the Huffman Tree below.



Construct a table of frequencies (for each of the symbols $a, b, c, d, e, f, g, h, i$) such that the Huffman Coding algorithm would produce the tree above (under the assumption about merge order stated above).