HKUST – Department of Computer Science and Engineering

COMP3711H: Honors Design and Analysis of Algorithms

Fall 2016

#

Midterm Examination

Date: Thursday, Oct. 20, 2016 Time: 19:00–21:00 Venue: Room 2304

Name: _____ Student ID: _____

Email: _____

Instructions

- This is a closed book exam. It consists of 28 pages and 8 questions .
- Please write your name, student ID and ITSC email at the top of this page.
- For each subsequent page that you write on, please write your student ID at the top of the page in the space provided.
- Please sign the honor code statement on page 2.
- Answer all the questions within the space provided on the examination paper. You may use the back of the pages for your rough work. The last 2 pages are scrap paper and may also be used for rough work. Each question is on a separate page and most have at least one extra page for writing answers. These are provided for clarity and are not meant to imply that each question requires all of the blank pages. Many can be answered using much less space.

Questions	1	2	3	4	5	6	7	8	Total
Points	10	8	14	12	12	20	12	12	100
Score									

Student ID: _____

As part of HKUST's introduction of an honor code, the HKUST Senate has recommended that all students be asked to sign a brief declaration printed on examination answer books that their answers are their own work, and that they are aware of the regulations relating to academic integrity. Following this, please read and sign the declaration below.

I declare that the answers submitted for this examination are my own work.

I understand that sanctions will be imposed, if I am found to have violated the University regulations governing academic integrity.

Student's Name: _____

Student's Signature: _____

Student ID: _____

1. **Huffman Coding [10 pts]**
   You are given that the following letters appear in a long message with the associated frequencies:

Letter	a	b	c	d	e	f	g
Frequency	3	3	7	8	9	14	15

   (a) **Build a Huffman Tree for these 7 letters with their associated frequencies.**
   Draw the Huffman tree to the left of the table below (labelling each leaf with its associated letter) and then fill in the table by writing down the codeword from the tree associated with each letter. You can show your work on the next page. (It is not necessary to show your work, but if you don't and make an error we will not be able to give you partial credit).

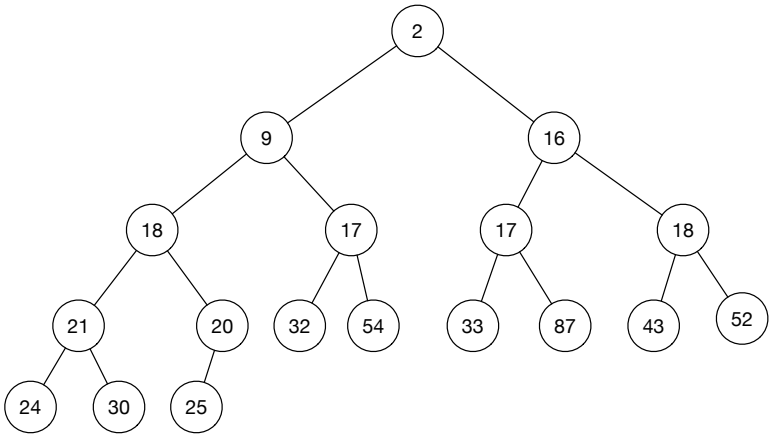
letter	codeword
a	
b	
c	
d	
e	
f	
g	

   (b) Write down the encoding of the word **feed** using the Huffman code you created.

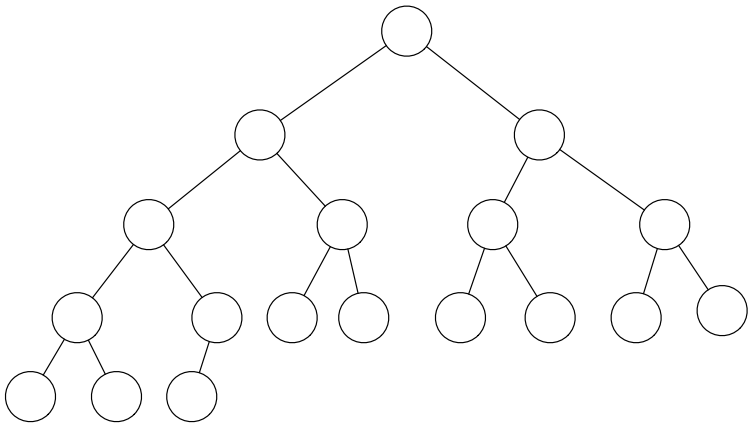
Student ID: _____

2. **Heaps [8 pts]**

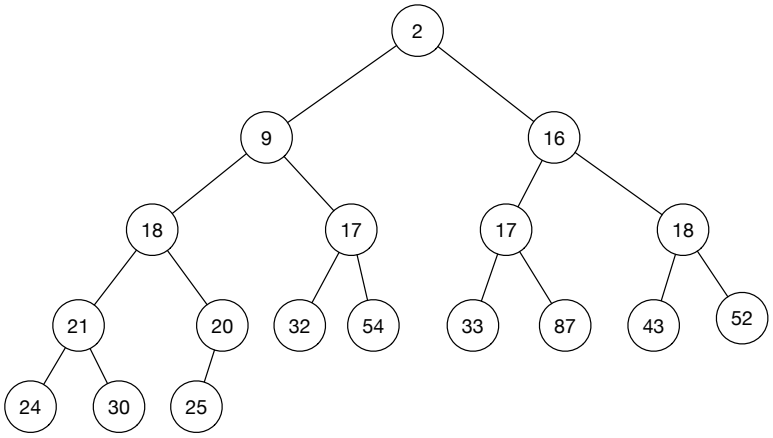
(a) Consider the Min-Heap below.



Insert a new key with the value **7** into the heap and fill in the values in the empty tree below to show the structure of the resulting heap. The new heap might have more or fewer nodes than the tree pictured. You should add or erase nodes appropriately.

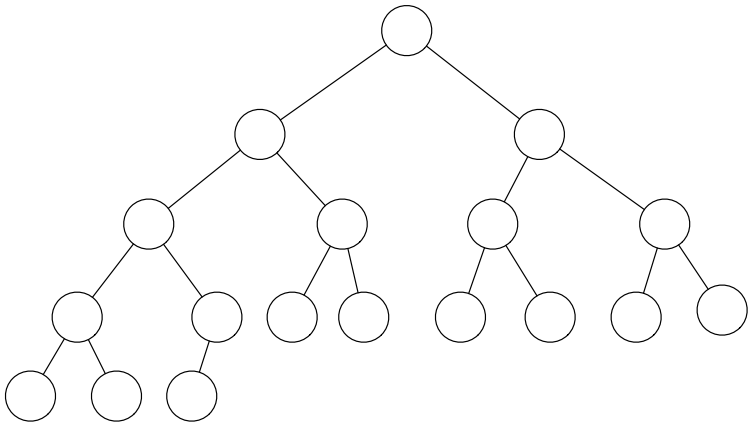


(b) Now reconsider the same original Min-heap as before (below).



Perform **Extract-Min** on this heap and fill in the values in the empty tree below to show the structure of the resulting heap.

The new heap might have more or fewer nodes than the tree pictured. You should add or erase nodes appropriately.



3. **Divide-and-Conquer [14 pts]**

Consider an array of numbers $A[1..n]$. We say that $A[\,]$ is k**-sorted** if the array is split into k equal-sized parts with all of the items in each part being less than all of the items in the previous part.

Formally, for every $j = 0, \ldots, k-2$, the items in locations

$$A\left[j\frac{n}{k}+1, \ldots, (j+1)\frac{n}{k}\right]$$

are all less than all the items in

$$A\left[(j+1)\frac{n}{k}+1, \ldots, (j+2)\frac{n}{k}\right].$$

For example, the 12 item array $A = [\mathbf{2}, \mathbf{4}, \mathbf{3}, 9, 8, 7, \mathbf{13}, \mathbf{14}, \mathbf{12}, 24, 22, 21]$ is 4-sorted (the 1st and 3rd parts have been highlighted for to show this) but is not 3-sorted.

Design an $O(n \log k)$ worst-case time algorithm for k-sorting an array.

You must describe (either in plain english or pseudocode) how your algorithm works, justify its correctness and prove that it runs in $O(n \log k)$ worst-case time.

Hint: Consider modifying quicksort with appropriate partitioning and stopping early.

Your algorithm may use any subroutine or algorithm learned in class as a black-box. If you do use such a subroutine or algorithm, you must explicitly state the subroutine or algorithm that you are using. its input and output and its running time.

For simplicity, you may assume that both n and k are powers of 2 and that all elements in the array are distinct.

4. **Interval Selection [12 pts]**
In this problem you must describe and explain the correctness of the greedy interval selection algorithm that was taught in class.

The problem: Assume you are given a set of n intervals along with their start and finish times s_i, f_i, $i = 1, \ldots, n$. An interval starts at time s_i and finishes at time f_i. Two intervals i and j are *compatible* if they do not overlap i.e., if $s_i \geq f_j$ or $s_j \geq f_i$. The activity selection problem is to select a maximum size set of mutually compatible intervals. That is, to find a largest size set in which no two activities overlap.

(a) Give the pseudocode for the greedy algorithm.
Also, analyze your algorithm's running time (and state what it is).

You *may not* assume that the s_i and/or f_i are originally sorted.

(b) Prove that your algorithm's output is correct, i.e., that it gives a **maximum** size set of mutually compatible intervals.

5. **Sorting [12 pts]** (continued on next tpage)

   (a) For each of the 4 sorts below write down the worst case number of comparisons used when the algorithms are run on an input of n items.

   Your answer should be one of $\Theta(n)$, $\Theta(n \log n)$ and $\Theta(n^2)$.

   You only need to write down your answer. You do not need to justify it.

Sorting Algorithm	**Worst-Case**
Insertion Sort	
Mergesort	
Quicksort	
Heapsort	

(b) Prove that sorting n items requires $\Omega(n \log n)$ comparisons.
Note that for the purposes of this problem you should assume that you are given n items $a_1, a_2, \ldots, a_n$ and you are only allowed to compare them using the operation "$\leq$". In particular, you are not allowed to get the values of the a_i.

The only mathematical statements that you may assume are the ones provided on the mathematical facts handout that was distributed along with this exam. All other statements used must be proven.

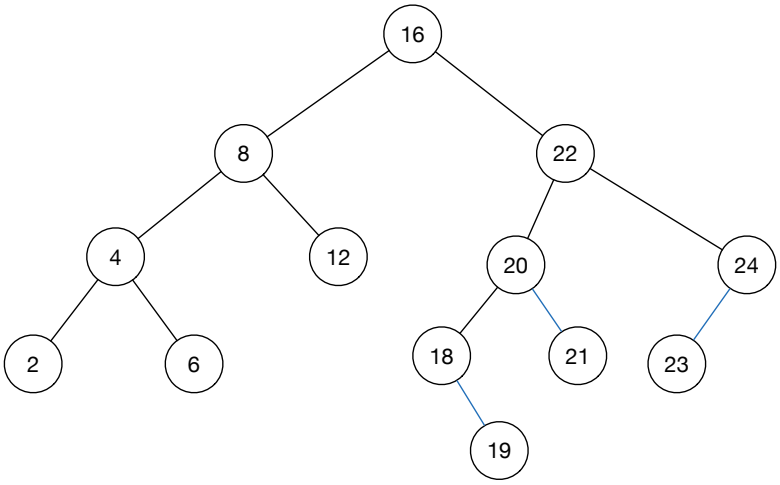
6. **AVL Trees [20 pts]** (parts (d) and (e) are on pages 18-19)

   (a) Define an AVL tree.

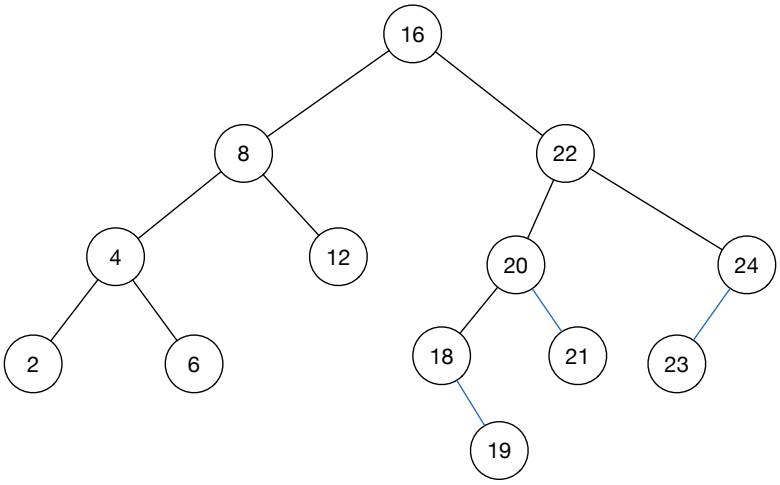
   (b) Prove that the minimum number of nodes in an AVL tree of height h is $\Omega(c^h)$ for some $c > 1$.

   (c) Prove that AVL trees with n nodes have height $O(\log n)$

(d) Draw the AVL tree that results after inserting a **3** into the AVL tree below.
There is no need to explain the rules you followed to do this insertion.



(e) Draw the AVL tree that results after inserting a **5** into the AVL tree below (the same original tree from the previous page).
As before, there is no need to explain the rules you followed to do this insertion.



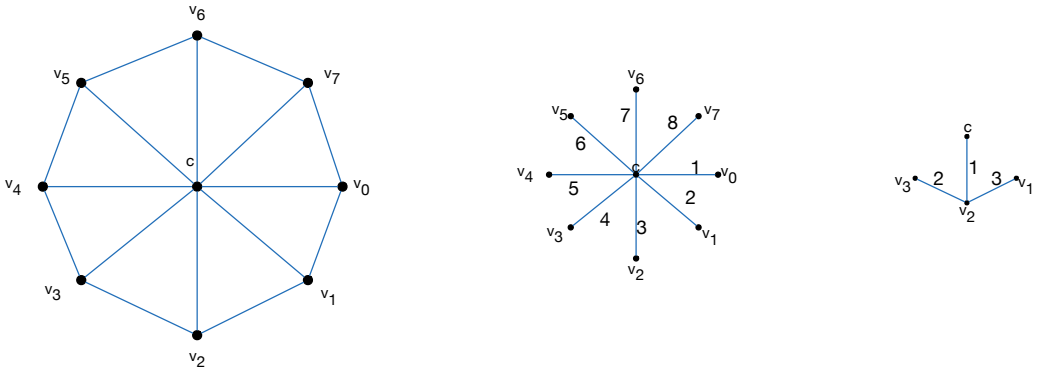
7. **BFS and DFS** [12 pts]

Consider the *wheel graph* $W_n = (V_n, E_n)$ with $n+1$ vertices $V_n = \{c\} \cup \{v_0, \ldots, v_{n-1}\}$ and edges E_n as given by the following adjacency lists:

$$c :\to v_0, \ldots, v_{n-1} \quad \text{and} \quad \forall i \in \{0, \ldots, n-1\} \quad v_i :\to c, v_{(i+1) \bmod n}, v_{(i-1) \bmod n},$$

i.e., c points to the vertices in increasing index order and every other vertex points to c first, then to the vertex following it and then the vertex preceding it.

The diagram below illustrates W_8, the ordering of the adjacency lists around c and, as an example, around v_2.



(a) Describe the tree produced by Breadth First Search run on W_n starting at root c.

(b) $\forall i \in \{0, \ldots, n-1\}$, describe the tree produced by Breadth First Search run on W_n starting at root v_i.

(c) Describe the tree produced by Depth First Search run on W_n starting at root c.

(d) $\forall i \in \{0, \ldots, n-1\}$, describe the tree produced by Depth First Search run on W_n starting at root v_i.

You should assume that when the algorithms process a node u's neighbors (in the code this was written as **foreach v** $\in$ **Adj(u)**), it processes them from left to right in the order that they are given in the adjacency

list. For example, when processing node v_5's neighbors, it processes them in the order c, v_6, v_4, and when processing node v_0's neighbors, it processes them in the order c, v_1, v_7

To describe the trees you only have to describe which edges are in the trees. You may either give a clear (English) description or you may simply give a formula that lists all of the edges in the trees. In both cases it would be advisable to illustrate your answer with a diagram.

8. **Convolutions [12 pts]**
Let $A(x) = \sum_{j=0}^{n-1} a_j x^j$ be a degree $n-1$ polynomial. Recall that the FFT of $A(x)$ will return the n values $A(\omega_n^j)$, $j = 0, 1, \ldots, n-1$ where $\omega_n^j = e^{\frac{2\pi i j}{n}}$ are the n'th roots of 1.

   (a) Let $a' = A(\omega_n^j)$, $j = 0, 1, \ldots, n-1$ be the output from running the FFT on $A(x)$.
   Now run the FFT algorithm on $A'(x) = \sum_{j=0}^{n-1} a'_j x^j$.
   Describe the relationship of the output $A'(\omega_n^j)$, $j = 0, 1, \ldots, n-1$ and the original values a_j, $j = 0, 1, \ldots, n-1$.

   Note: this only has to be described. It does not have to be proven!

   (b) Given two degree $n-1$ polynomials

$$A(x) = \sum_{j=0}^{n-1} a_j x^j \quad \text{and} \quad B(x) = \sum_{j=0}^{n-1} b_j x^j$$

   describe an $O(n \log n)$ time algorithm for finding the $2n-1$ coefficients, c_j, $j = 0, 1, \ldots, 2n-2$ of their product polynomial $C(x) = \sum_{j=0}^{2n-2} c_i x^j = A(x)B(x)$.

   For this problem you may assume that n is a power of 2. You may also assume the existence of an $O(n \log n)$ Fast-Fourier-Transform algorithm. The FFT procedure may be considered as given; you do not have to describe how it works or why it runs in $O(n \log n)$ time. You may also assume the correctness of the result of part (a) of this problem.

   For this part you must

   i. describe the algorithm that calculates the coefficients of the product polynomial and
   ii. justify its correctness
   iii. explain why the algorithm runs in $O(n \log n)$ time.

   Justification requires stating the mathematical principles used. You do not have to prove the mathematical principals.

Student ID: _____

Scrap Paper

Student ID: _____

Scrap Paper