#

HKUST – Department of Computer Science and Engineering

COMP3711: Design and Analysis of Algorithms – Spring 2019

Midterm Examination

Date: Monday, March 18, 2019 Time: 19:20–21:20 Venue: LTC/LTD

Name: ______________	Student ID: ______________
Email: ______________	Lecture L1 / L2

Instructions

- This is a closed book exam. It consists of 23 pages and 6 questions.
- Please write your name, student ID and ITSC email and circle your Lecture section (L1 is TTh and L2 is WF) at the top of this page.
- For each subsequent page on which you write a solution, please also 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. The last 2 pages are for rough work but can be used for real solutions if you CLEARLY MARK THEM as such.
- Most questions provide at least one extra page for writing answers. This is for clarity and is not meant to imply that solutions require all of the blank pages. Many can be answered using much less space.
- All solutions must be written on the front (numbered) side of the pages. Page backs may only be used for rough work. Nothing written on the backs of pages will be marked.

Questions	1	2	3	4	5	6	Total
Points	15	10	18	20	15	22	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. **Time Complexity** [15 pts]

a) We have two algorithms, A and B. Let $T_A(n)$ and $T_B(n)$ denote the time complexities of algorithm A and B respectively, with respect to the input size n. Listed below are 9 different cases of time complexities or formulas for each algorithm. Complete the last column of the following table with "A", "B", or "U", where:

- "A" means that for large enough n, algorithm A is always faster;
- "B" means that for large enough n, algorithm B is always faster;
- "U" means that the information provided is not enough to justify stating that, for large enough n, one algorithm is always faster than the other.

Recall that an algorithm is "faster" if its running time is smaller. To get you started we have provided two example questions and their answers.

Notational comment: $\log^a b$ is shorthand for $(\log b)^a$. For example $\log^{10} n = (\log n)^{10}$.

Case	$T_A(n) =$	$T_B(n) =$	Faster
1	$\Theta(1)$	$\Theta(n)$	A
2	$\Omega(1)$	$O(n)$	U
3	$\Omega(n/\log n)$	$\Omega(10^{10})$	
4	$\Theta(n^3)$	$O(n^{\log_2 7})$	
5	$4T_A\left(\frac{n}{4}\right) + n$	$\Theta(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + ... + \frac{1}{2^n})$	
6	$\Theta(n^{1.5}\log^3 n)$	$\Theta(\log_3(n!))$	
7	$O(\log^{10}(\log n))$	$\Omega(n \log n)$	
8	$O(16^{\log_4 n} + n^{1.5})$	$5T_B\left(\frac{n}{2}\right) + n^2$	
9	$\Omega(\sqrt{5n})$	$\Theta\left(\left(\sqrt{5}\right)^n\right)$	

Continued on next page ....

(b) Derive from first principles (you can't use the Master Theorem) the tight asymptotic solution to

$$T(1) = 2; \qquad \forall n > 1, \quad T(n) = 10T(n/3) + n^2.$$

Show your steps. Your final solution should be in the form $T(n) = \Theta(n^c)$ or $T(n) = \Theta(n^c \log n)$ for some constant $c > 0$. You may assume that n is a power of 3.

Student ID: ________

2. **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	h
Frequency	2	2	2	3	4	5	7	10

Build a Huffman Tree for these 8 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	
h	

3. **Interval Scheduling [18pts]**

   In this problem you must describe and explain the correctness of the interval scheduling algorithm that was taught in class.

   The Interval Scheduling Problem:
   Assume you are given a set of n SORTED intervals $I_i = [s_i, f_i]$, $i = 1, \ldots, n$. s_i, f_i are, respectively, the starting and finishing times of interval I_i.

   Two intervals I_i and I_j are *compatible* if they do not overlap i.e., if $s_i \geq f_j$ or $s_j \geq f_i$. The *interval scheduling problem* is to select a maximum size set of mutually compatible intervals. That is, to find a largest size set in which no two intervals overlap.

   The Greedy Algorithm: ran the following code

```
1. $A = \{I_1\}$;
2. For $j = 2$ to $n$
3.     if $I_j$ is compatible with the intervals in $A$
4.         $A = A \cup \{I_j\}$; % Add $I_j$ to the set
5. Output set $A$
```

   The problem definition did not state HOW the intervals were sorted. Consider the following two ways of sorting.

   (a) Assume that the intervals are sorted by increasing starting time, i.e, $s_1 \leq s_2 \leq \cdots \leq s_n$.

   (b) Assume that the intervals are sorted by increasing finishing time, i.e, $f_1 \leq f_2 \leq \cdots \leq f_n$.

   For both (a) and (b) separately answer the following question.

   Will the greedy algorithm return the correct result, i.e., a maximum size set of mutually compatible intervals?

   If the answer is yes, provide a formal proof of correctness.

   If the answer is no, provide a counterexample. A counterexample is a set of intervals (sorted by starting time in (a) and finishing time in (b)) for which the greedy algorithm does not output a correct solution.

Student ID: ________

4. Sorting [20 pts]
   This problem has three parts. Each part is on a different page.

   (a)

      i. What does it mean to say that a sorting algorithm is *stable*?

      ii. Give the names of two stable sorting algorithms that you learned in class.

      iii. Give the name of a sorting algorithm that you learned in class that is not stable.

Continued on next page ....

(b) The problem input is n/k lists:
(i) Each list has size k, and
(ii) for $i = 2$ to n/k, the elements in list $i - 1$ are all less than all the elements in list i

The obvious algorithm to fully sort these items is to sort each list separately and then concatenate the sorted lists together. This uses $\frac{n}{k}O(k \log k) = O(n \log k)$ comparisons.

Show that this is the best possible. That is, any comparison-based sorting algorithm to sort the n/k lists into one sorted list with n elements will need to make at least $\Omega(n \log k)$ comparisons.

You may use any theorem or fact that we learned in class as long as you explicitly write out the statement of that theorem or fact. You may also use any fact from the Math sheet distributed with the exam. If you use a fact from the sheet you must explicitly identify that fact and state that it is from the sheet.

Continued on next page ....

Continued on next page ....

Student ID: _____

(c) The leftmost array gives 8 3-digit numbers. Run Radix-Sort on these numbers. Show the result after sorting the array on each digit. The rightmost array should be your final result.

3 2 2			
8 1 3			
1 2 3			
4 3 2			
3 5 2			
4 6 2			
7 1 8			
4 9 8			

Student ID: ________

5. **Indicator Random Variables** [15 pts]

(a) Recall the Hat-Check problem from the tutorial that was analyzed using the indicator random variable technique. In that problem, each of n customers gives a hat to a hat-check person at a restaurant. The hat-check person gives the hats back to the customers in a random order.

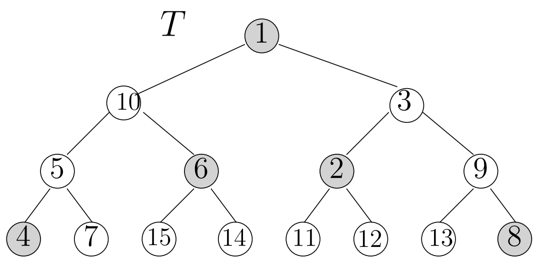
What is the expected number of customers who get back their own hat? Prove the correctness of your answer.

(b) For the purpose of this problem assume $n = 2^k - 1$ where k is a positive integer. Let $A[1 \ldots n]$ be an array. Recall the correspondence between arrays A and trees T that we learned in the Heapsort lecture.

The root is in array position 1. For any element in array position i, its left child is in position $2i$ and its right child is in position $2i + 1$.

Define a *local minimum* of the tree T to be a node in the tree whose value is less than all of its neighbors (nodes that it connects to).

In the example below, the shaded nodes are the local minima.



i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$A[i]$	1	10	3	5	6	2	9	4	7	15	14	11	12	13	8

Now suppose that the elements of A form a uniform random permutation of $< 1, 2, \ldots, n >$. That is, each of the $n!$ possible permutations is equally likely to occur.

Use the indicator random variable technique to calculate the expected number of local minima in T if the elements of A form a uniform random permutation of $< 1, 2, \ldots, n >$. Show your work.

Hint: Let $X_i = 1$ if $A[i]$ is a local minimum and $X_i = 0$ if it is not. Calculate the expected value of $\sum_{i=1}^{n} X_i$.

Student ID: ________

Student ID: ________

6. **Dynamic Programming** [22pts]

You are given an input array $A[1 \ldots n]$. Recall that the *maximum-contiguous subarray (MCS)* is a subarray $A[i \ldots j]$ such that $\sum_{k=i}^{j} A[k]$ is maximum among all subarrays. As an example, $A[4 \ldots 7]$ is a MCS of the array below

k	1	2	3	4	5	6	7	8	9	10
A[k]	-3	3	-5	18	-1	2	8	-50	-30	5

and has value $18 - 1 + 2 + 8 = 27$.

The problem is now modified so that for given $x > 0$ you need to find the *x-discounted MCS*. The x-discounted *cost* of $A[i \ldots j]$ is defined as

$$C(i, j : x) = A[j] + xA[j-1] + x^2 A[j-2] + \cdots + x^{j-i} A[i] = \sum_{k=0}^{j-i} A[j-k]x^k.$$

In the example array above,

$$C\,(2, 7 : 2) = 8 + 2 \cdot 2 - 2^2 \cdot 1 + 2^3 \cdot 18 - 2^4 \cdot 5 + 2^5 \cdot 3 = 168$$

Given $x > 0$, the *x-discounted* MCS is the subarray $A[i \ldots j]$ such that $C(i, j : x)$ is maximum among all subarrays.

By definition, if $x = 1$, the *x-discounted MCS* is exactly the MCS. If $x \neq 1$ it might be different.

In the array above, for example, if $x = 2$, then $A[2 \ldots 7]$ is the *x-discounted* MCS of the full array.

The full problem is, given array $A[1 \ldots n]$ and $x > 0$, to design an $O(n)$ time dynamic programming algorithm that calculates the cost of the x-discounted MCS.

Do this by completing parts (A),(B), (C) and (D) listed on the following page. Solutions (except for part (B) should be written on pages 20 and 21.

Continued on next page ....

(A) Prove that, for every i, j with $1 \le i < j \le n$ and $x > 0$,

$$C(i, j : x) = A[j] + xC(i, j - 1 : x).$$

Note: For clarity, we emphasize that the definition implies $C(j, j : x) = A[j]$. Also, if you are not able to prove part (A) you may still assume its correctness later in the problem.

(B) Define

$$V_j = \max_{1 \le i \le j} C(i, j : x).$$

Give a recurrence relation (write it in the space below) for V_j in terms of V_i with $i < j$ and the values in the array.

$$V_j = \begin{cases} \underline{\hspace{4cm}} & \text{if } j = 1 \\ \underline{\hspace{11cm}} & \text{if } j > 1 \end{cases}$$

Justify the correctness of your recurrence relation

(C) Give documented pseudocode for your DP algorithm to calculate the cost of the x-discounted MCS (based on the recurrence from part (B)), and explain why it is correct.

(D) Analyze the running time of your algorithm. Full credit will be given for $O(n)$ algorithms. Algorithms slower than that will only receive partial credit.

Student ID: ________

Scrap Paper

Student ID: __________

Scrap Paper