NATIONAL UNIVERSITY OF SINGAPORE
SCHOOL OF COMPUTING

FINAL TERM TEST FOR
Semester 2, AY2019/2020

CS3230 – DESIGN AND ANALYSIS OF ALGORITHMS

5 May 2020 Time Allowed: 2 hours

Instructions to Candidates:

1. This paper consists of **FOUR** questions (each consisting of **TWO** sub-questions) and comprises **FOUR (4)** printed pages, including this page.
2. Answer **ALL** questions.
3. This is an **OPEN BOOK** examination.
4. Write down **Metric No.** on top of every page of your answer sheet
5. **SUBMISSION: Page limit for each of the FOUR questions is three.** After writing down your answers scan them to convert to **.pdf** file. Create **exactly one file for each of the FOUR questions** and rename your file as **<Metric No.>_T<tutorial group no.>_Q<question no.>.pdf** (for example, **A324516H_T01_Q3.pdf**). Then submit each file separately in the corresponding LumiNUS folder (you can see four different folders for each of the FOUR questions inside Files→Final Exam).
6. **Plagiarism Policy:** If we will find plagiarism (either copy from someone or show your answer to someone) on any part of your answer you will get 0 on the whole paper.

QUESTION	POSSIBLE	SCORE
Q1	20	
Q2	20	
Q3	20	
Q4	20	
TOTAL	**80**	

IMPORTANT NOTE:

- *You can **freely quote** standard algorithms and data structures covered in the lectures, tutorials and assignments. Explain **any modifications** you make to them.*
- *Unless otherwise specified, you are expected to **prove (justify)** your results.*

Q1. (20 points)

(a) Consider a line with n+1 stations S_0, S_1, …, S_n, where S_0 is the initial station and S_n is the final station. We want to transfer an object from S_0 to S_n. At every station, we have a catapult. We can use the catapult at station S_i to throw the object to any station S_j where j = i+1, …, i+f(i). Note that f(i)≥1 for all i. We want to transfer the object from S_0 to S_n using the minimum number of shoots. You believe that this problem can be solved using dynamic programming. Given f(i) for i=0, 1, …, n-1 (as an array F[0…n-1]), can you give an $O(n^2)$-time dynamic programming algorithm to solve this problem? Please detail your algorithm and argue that it is correct. Please also state the time complexity of your algorithm.

(b) Suppose we have i+f(i)≥j+f(j) for all i>j. Under this extra assumption can you give an O(n)-time algorithm to solve this problem? Please detail your algorithm and argue that it is correct. Please also state the time complexity of your algorithm.

Q2. (20 points)

(a) You are given an integer array A[1..n] and an integer x. Can you develop an efficient algorithm Partition(A, x) that partitions A into two arrays X and Y such that all numbers in X are smaller than x and all numbers in Y are bigger than or equal to x? What is the running time of your algorithm?

(b) You have an unsorted integer array A[1..n]. You also have a sorted integer array B[1..k] (k<n), where B[1]<B[2]<…<B[k]. For any 1≤i<k, let S_i be the set { A[j] | B[i]≤A[j]<B[i+1] }. You need to compute the array C[1..k-1] where C[i] is the median of all integers in S_i if S_i is a non-empty set; otherwise set C[i]=0. Can you propose an $O(n \log k)$ time algorithm to compute C[1..k-1]?

Q3. (10+10= 20 points)

(a) Suppose you are given two strings x and y of length n each, over ternary alphabet, i.e., $x, y \in \{0,1,2\}^n$. Now consider the problem of finding a Longest Common Subsequence of x and y. Design an $O(n)$ time algorithm with approximation ratio 1/3. Provide clear description of your algorithm and its proof of correctness. (Note, in the lecture we have seen an algorithm that finds Longest Common Subsequence of x and y in $O(n^2)$ time.)

(b) The Independent Set problem is defined as: Given an undirected graph $G = (V, E)$ find a maximum sized subset $X \subseteq V$such that for all $u, v \in X$, $(u, v) \notin E$.
Now consider the following randomized algorithm. Among $n!$(where $|V| = n$) possible orderings of the vertices pick one uniformly at random. Then start processing each vertex based on this selected ordering. Initially take an empty set S. Then at any particular step, if $u \in V$ is processed and for all $(u, v) \in E$, $v \notin S$, add u to S. Otherwise continue with the next vertex based on the previously selected ordering. Stop when all the vertices have been processed, and output the set S.
Show that the set S output by the algorithm is an independent set of expected size at least $\frac{1}{d+1} OPT$, where OPT denotes the size of the maximum independent set and d is the maximum degree of any vertex.

Q4. (10+10 = 20 points)

(a) In the Set Union problem we have n elements, that each are initially in n singleton sets, and we want to support the following operations:

- **Union(A,B):** Merge the two sets A and B into one new set C = A∪ B destroying the old sets.
- **SameSet(x, y):** Return true, if x and y are in the same set, and false otherwise.

We implement it in the following way. Initially, give each set a distinct color. When merging two sets, recolor the smaller (in size) one with the color of the larger one (break ties arbitrarily). Note, to recolor a set you have to recolor all the elements in that set.
To answer **SameSet** queries, check if the two elements have the same color. (Assume that you can check the color an element in O(1) time, and to recolor an element you also need O(1) time. Further assume that you can know the size of a set in O(1) time.)

Use Aggregate method to show that the amortized cost is O(log n) for **Union**. That means, show that any sequence of m union operations takes time $O(m \log n)$. (Note, we start with n singleton sets.)

(b) Suppose Alice insists Bob to maintain a dynamic table (that supports both insertion and deletion) in such a way its size must always be a Fibonacci number. She insists on the following variant of the rebuilding strategy. Let F_k denote the k-th Fibonacci number. Suppose the current table size is F_k.

After an insertion, if the number of items in the table is F_{k-1}, allocate a new table of size F_{k+1}, move everything into the new table, and then free the old table.
After a deletion, if the number of items in the table is F_{k-3}, we allocate a new hash table of size F_{k-1}, move everything into the new table, and then free the old table.
Use either Potential method or Accounting method to show that for any sequence of insertions and deletions, the amortized cost per operation is still O(1). (If you use Potential method clearly state your potential function. If you use Accounting method clearly state your charging scheme.)

-- End of Paper ---