COMP 5711: Advanced Algorithms
Fall 2018 Final Exam

Name: ______________________________ Student ID: ______________________________

Instructions

1. This is an open-book, open-notes exam.
2. Time limit: 180 minutes.
3. You can write on the back of the paper if you run out of space. Please let us know if you need more scratch paper.
4. All the log's have base 2 unless stated otherwise.

Q1	Q2	Q3	Q4	Q5	Q6	Total

1. **Short-answer Questions** (15 pts)

   (a) If in the set of stack operations, MULTIPOP were replaced by MULTIPUSH, which pushes k items onto the stack, would the amortized cost per operation still be $O(1)$?

   (b) Quicksort is a Las Vegas algorithm with expected running time $O(n \log n)$. How do we convert it to a Monte Carlo algorithm, i.e., one that has worst-case running time $O(n \log n)$, but may fail with probability at most $1/2$?

   (c) Show that the Markov inequality is tight, namely, give an example of a non-negative random variable X and a value $k > 1$ such that $\Pr[X \geq kE[X]] = 1/k$. (Hint: X can just take two possible values.)

(d) Decide the treewidth of the following graphs:

(1) a tree;

(2) a cycle of n vertices ($n \geq 3$);

(3) a complete graph with n vertices ($n \geq 3$).

(e) Given an undirected graph $G = (V, E)$. We know that if $S \subseteq V$ is a vertex cover of G, then $V - S$ is an independent set of G, so the two problems are equivalent. We know that the k-vertex cover problem (i.e., decide if G has a vertex cover of size k) is in FPT. Does it imply that the k-independent set problem (i.e., decide if G has an independent set of size k) is also in FPT?

2. **Amortized Analysis** (15 pts)

A *multi-stack* is a data structure consisting of a series of stacks $S_0, S_1, S_2, \cdots$. For each i, S_i can hold up to 2^i elements. When we push an element x onto the multi-stack, if S_0 is not full, we simply push it onto S_0. If S_0 is full, we pop all the elements of S_0 and push them to S_1. Then we push x onto S_0. More generally, whenever we try to push an element onto S_i and S_i is full, we move all the elements of S_i to S_{i+1}, and then push the element onto S_i. We do not support the pop operation in the multi-stack.

(a) In the worst case, what's the cost of pushing an element onto a multi-stack containing n elements?

(b) Prove that the amortized cost of a push operation (onto the multi-stack) is $O(\log n)$ where n is the maximum number of elements in the multi-stack.

3. **Random Walk on a Complete Graph** (20 pts)

Let G be an undirected complete graph having n vertices. For a vertex v, $N(v)$ denotes the set of neighbors of v in G. A random walk on G is the following process, which occurs in a sequence of discrete steps : starting at a vertex v_0, we proceed at the first step to a neighbor of v_0 chosen uniformly at random. This may be thought of as choosing a random edge incident on v_0 and walking along it to a vertex $v_1 \in N(v_0)$. At the second step, we proceed to a randomly chosen neighbor of v_1, and so on.

(a) Show that the expected number of random walk steps needed to visit all vertices is $(n-1)H_{n-1}$ where $H_{n-1} = \sum_{i=1}^{n-1} 1/i$.

(b) For two vertices u and v of G, show that the expected number of steps of random walk that begins at u and ends upon first reaching v is $n-1$.

4. **Hashing** (20 pts)

Consider the following hash families from $\{0,1\}^n$ to $\{0,1\}$. For each hash family, decide if it is (i): pairwise-independent, (2) universal but not pairwise-independent, or (3) not universal. Give a proof of your claim. Note that you need to give a counterexample if your claim includes "not pairwise-independent" or "not universal".

(a) $H = \{h_{a_0,\ldots,a_n}(x_1,\ldots,x_n) \mid a_0,\ldots,a_n \in \{0,1\}\}$, where

$$h_{a_0,\ldots,a_n}(x_1,\ldots,x_n) = a_0 + a_1 x_1 + \ldots a_n x_n \mod 2.$$

(b) $H = \{h_{a_1,\ldots,a_n}(x_1,\ldots,x_n) \mid a_1,\ldots,a_n \in \{0,1\}\}$, where

$$h_{a_1,\ldots,a_n}(x_1,\ldots,x_n) = a_1 x_1 + \ldots a_n x_n \mod 2.$$

5. **FPT Algorithms** (15 pts)

   Consider a set $U = \{a_1, \ldots, a_n\}$ of n elements and a collection $\mathcal{S} = \{S_1, \ldots, S_m\}$ of m subsets of U. We say a sub-collection $\mathcal{C} \subseteq \mathcal{S}$ is a *set cover* of U if the union of the subsets in $\mathcal{C}$ is equal to U. The size of a set cover $\mathcal{C}$ is defined to be $|\mathcal{C}|$. For example, $U = \{1, 2, 3, 4, 5\}$ and $\mathcal{S} = \{\{1, 2, 3\}, \{1, 4\}, \{3, 5\}, \{2, 4, 5\}\}$. Clearly $\mathcal{S}$ itself is a set cover of U, and its sub-collection $\{\{1, 2, 3\}, \{2, 4, 5\}\}$ is another set cover.

   Suppose we are given an instance $(U, \mathcal{S})$ of the set cover problem, and we want to determine whether there is a set cover of size at most k. Furthermore, we assume that for each element $a \in U$, at most c subsets in $\mathcal{S}$ contain a, for some constant c. Give an algorithm that solves this problem with a running time of the form $O(f(c, k) \cdot \text{poly}(n, m))$, where $\text{poly}(\cdot)$ is a polynomial function, and $f(\cdot)$ is an arbitrary function that depends only on c and k, not on n or m.

6. **Balls and Bins** (15 pts)

   Consider the balls and bins problem. Show that if we throw $2n \ln n + 2cn$ balls to n bins for some constant c, then we can fill all the bins with probability at least $1 - \frac{1}{e^c}$. [Hint: You can use the following form of Chernoff inequality: For any $0 < \delta < 1, \Pr[X \leq (1-\delta)\mu] \leq \exp(-\mu\delta^2/2)$.]