COMP 5711: Advanced Algorithms
Fall 2019 Midterm Exam

Name: ______________________ Student ID: ______________________

Instructions

1. This is an open-book, open-notes exam.
2. Time limit: 80 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.

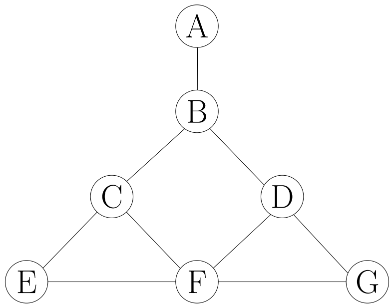
1. **Short-answer Questions** (25 pts)

   (a) (5 pts) Suppose we run Dijkstra's shortest path algorithm on a graph with n vertices and $n \log n$ edges. What's the running time if we use a binary heap and a Fibonacci heap, respectively?

   (b) (5 pts) Suppose we have designed a parameterized algorithm for a problem with running time $O(2^{2^k}) \cdot \text{poly}(n)$. For what values of k is this algorithm a polynomial-time algorithm?

   (c) (5 pts) Suppose you assign k jobs to k machines, each job to any of the k machines uniformly and independently at random. Let $S(k)$ be the expected number of machines that receive exactly one job. What is $\lim_{k\to\infty} S(k)/k$? Give the exact value, without using asymptotic notation.

(d) (10 pts) Draw a tree decomposition of width 2 for the following graph.



2. **Enumerating k-combinations** (30 pts)

   Define $[n] = \{0, 1, \ldots, n-1\}$. A k-combination of $[n]$ for $k < n$, is a subset of k elements from $[n]$. Consider the following algorithm that enumerates all such k-combinations. The array A is initialized to $A = [0, 1, \ldots, k-1]$. Then we repeatedly call the following procedure $\binom{n}{k}$ times:

   Algorithm 1: EnumerateNext

   ```
   1 for i ← k − 1 to 0 do
   2     if A[i] < n − k + i then
   3         A[i] ← A[i] + 1;
   4         for j ← i + 1 to k − 1 do
   5             A[j] ← A[j − 1] + 1;
   6         enumerate A;
   7         return
   ```

   We assume that line 6 runs in $O(1)$ time.

   (a) (10 pts) In trying to show that the algorithm EnumerateNext runs in $O(1)$ time amortized, one defines the potential as $\Phi =$ number of i's such that $A[i] = n - k + i$. Note that $n - k + i$ is the largest possible value $A[i]$ can take. However, this does not quite work. Where does the analysis go wrong?

(b) (20 pts) Can you fix the algorithm so that its amortized running time is $O(1)$? Show that your modified algorithm runs in $O(1)$ time amortized. [Hint: Introduce another global variable to remember an "important" location in A.]

3. **Approximate Shortest Path in the Plane** (25 pts)

   Suppose we are given n points in the unit square, which always include $(0,0)$ and $(1,1)$. We are also given a list of (u,v) pairs, meaning that we can travel from u to v or from v to u, and the distance is just the Euclidean distance between u and v. The goal is to find the shortest path from $(0,0)$ to $(1,1)$. (You are guaranteed that a path always exists.) The standard approach is to run Dijkstra's algorithm on the graph defined by these points.

   As we represent the coordinates of the points using floating-point numbers, there will be some rounding errors. Suppose the rounding error for each coordinate is at most θ. Prove that, if $\theta \le \frac{\varepsilon}{4n}$, then running Dijkstra's algorithm on the rounded instance returns a $(1+\varepsilon)$-approximation of the true shortest path, i.e., $dist(p) \le (1+\varepsilon)dist(p^*)$, where p is the path found by Dijkstra's algorithm on the rounded instance, and p^* is the true shortest path. Note that $dist(\cdot)$ denotes the true length of the path before rounding. [Hint: The path travels at most n edges of the graph, and its total distance is at least $\sqrt{2}$.]

4. **Skip List** (20 pts)

   We showed in class that the expected number of levels in a skip list is $O(\log n)$. Now show that it is also $\Omega(\log n)$, hence $\Theta(\log n)$. [Hint: It suffices to show that it has $\Omega(\log n)$ levels with at least constant probability.]