The exam will be 80 minutes long, but you will have 3 hours to take it. It will be open notes (you are allowed to bring in your own notes, handouts from the course, books, magazines, etc). No electronic devices (phones, tablets, kindles, laptops) will be allowed.

Practice Exam A

1. (Short Answers)

   (a) (Graph Transformations) Mark the following as true or false. If false, give a counterexample.

      i. Given an instance of the single-source shortest paths algorithm where the edge lengths can be negative, suppose the most negative number is $-C$. Then we can add $C$ to every edge length, and use Dijkstra’s algorithm to correctly compute shortest paths.

      ii. Given an instance of the minimum spanning tree (MST) problem where the edge lengths can be negative, suppose the most negative number is $-C$. Then we can add $C$ to every edge length, and use Kruskal’s algorithm to correctly compute the MST.

      iii. Given a MST instance with non-negative edge weights, suppose we replace each edge weight $w_e$ by its square $w_e^2$. The MST remains the same.

   (b) When we discussed amortized analysis, we saw that incrementing a binary counter $n$ times incurred cost at most $2^n$, to give an amortized cost of at most 2. What is the amortized cost if we have a decimal (base 10) counter and the incrementing cost is again the number of digits changed?

   (c) In the streaming lecture we saw an algorithm that output a majority element (one that occurs more than 50% of the time), and only stored one element and one number (say, of bit length logarithmic in the size of the universe). It had false positives—it could claim an element was a majority when it was not—but no false negatives.

      Dr. Demento claims he can do exactly the same thing without any errors. Give a stream of elements that would show Dr. D.’s claim must be false, and a short reason (1-2 sentence) why.

2. (All Inversions.) Give an algorithm that on an input array $A$ of $n$ distinct elements, counts the number of pairs $(i, j)$ such that $i < j$ and $A[j] < A[i]$. The faster your algorithm is, the better. (Hint: Divide and conquer) Be sure to include and short proof of both correctness and timing.

3. (A Lot of Change.) You have coins of $n$ integer denominations $0 < c_1 \leq c_2 \leq \cdots \leq c_n$ and would like to know whether you can make change in the amount of $m$ by using as many coins of each denomination as you want. In general this is a known hard problem. The goal is to find an algorithm that is efficient if $c_1$ is small.

   Let $C$ be the set of all values including zero that we can use the coins to make change for.

   We shall develop the algorithm in stages.
(a) Suppose we have an array $M$ with one value for each $0 \leq i < c_1$:

$$M_i = \min \left( \{\infty\} \cup \left\{ c \mid c \equiv i \pmod {c_1} \quad \text{and} \quad c = \sum_{i \in I} c_i, \quad I \subseteq [n] \right\} \right)$$

Show that if we have this list of values, we can answer whether change can be made for $m$ in $O(1)$ time.

(b) The goal of this part is to find a DP or shortest path algorithm for computing the array $M$.

   i. What should the value of $M_0$ be?

   ii. Consider the following graph $G = (V, E)$ where $V = \{V_0, \ldots, V_{c_1-1}\}$ and there is an edge from $V_i$ to $V_j$ of weight $c_k$ if

   $$i + c_k \equiv j \pmod {c_1}.$$  

   Let $d_i$ be the distance from $V_0$ to $V_i$ in $G$.

   Explain why $d_i \leq M_i$ by considering the set of coins used to make the amount $M_i$.

   Explain why $M_i \leq d_i$.

   iii. Explain how one can compute the values $d_i$ in $O(c_1 n \log c_1)$ time.

(c) Use the above part to give an algorithm for the make change problem that runs in $O(c_1 n \log c_1)$ time for the question: Can I make change for the amount $m$? If the answer is yes can you also return the change, the vector of quantities of each coin?

4. (A Matter of Degree.) You are given the complete graph $K_n$ on $n$ vertices. (Say the vertices are numbered $1, 2, \ldots, n$.) You retain each edge of this graph with probability $p := \frac{10 \ln n}{n-1}$, independently. This process results in a random subgraph $H$ of $K_n$.

   (a) Let $D_i$ be the degree of vertex $i$. What is $E[D_i]$? What is $\text{Var}(D_i)$?

   (b) Use Chebyshev’s inequality to show that the maximum degree is at most $O(\sqrt{n \log n})$ with probability at least $1/2$.

   (c) The degree-imbalance of the graph $H$ is the ratio of the largest degree to the smallest degree. Use Chernoff-Hoeffding to prove that the degree-imbalance of $H$ is at most 4 at most 10 with probability at least $1 - 1/n$. (You can probably show a better bound than 4 than 10, this is just for convenience.)

   (d) Let $T$ be the number of triangles in $H$. What is $E[T]$? Bonus 5 points: what is $\text{Var}(T)$?

5. (Treaping Along.)

   (a) Suppose that 4 elements are stored in a treap. What is the probability that the tree has following structure? Explain your answer.

```
               0
              / \
             0   0
            \   
             0
```
(b) Let $T$ be a binary tree on $n$ nodes, and $L$ and $R$ be its left and right subtrees, respectively. Let $\text{prob}(T)$ be the probability that we generate $T$ as a treap on $n$ nodes. Write a recurrence for $\text{prob}(T)$ in terms of $\text{prob}(L)$ and $\text{prob}(R)$. Give an explanation why your recurrence is correct. (Make sure it gives the correct answer for the above example.)
Practice Exam B

1. (Short Answer Questions)
   (a) In order to find the top $k$ principal components for an $n \times D$ matrix $A$, we find the top $k$ eigenvectors for which matrix?

   (b) If $X_i$ are independent Normal $N(0,1)$ random variables, then what is the distribution for $\sum_{i=1}^{k} c_i X_i$?

   (c) Consider the following hash family: which of the following statements is true about it.

   $$
   \begin{array}{c|ccc}
   h_1 & a & b & c \\
   h_2 & 1 & 0 & 1 \\
   h_3 & 1 & 1 & 0 \\
   h_4 & 0 & 0 & 0 \\
   \end{array}
   $$

   Is it universal? Is it pairwise independent?

   (d) You throw $m$ balls into $n$ bins, where each ball is independently thrown into a uniformly chosen bin. Let $L_i$ be the load in bin $i$. For a given bin $i$, what is $E[L_i]$? What is $\text{Var}(L_i)$? What is $\Pr[L_i = 0]$? What is the expected number of empty bins?

   (e) Dr. Evil claims he has a priority queue that can implement inserts and delete-mins in constant time.\(^1\) Give a one-sentence reason why he must be wrong.

   (f) Your friend has thought of a bit-string $x$ of length $n$ with a single 1 in it. You can ask $\log_2 n$ questions non-adaptively to find the location of this 1 (and hence recover $x$). Specifically, each question consists of a vector $v \in \mathbb{B}^n$, and its answer is the value $\langle v, x \rangle$. What should your $i^{th}$ question be?

2. (Can You Hear Me Now?) Your company is installing cell-phone towers along Route 66 (which for this purpose is a straight line), and there are $n$ houses along it. The requirement is that each house should have a tower within distance $D$ of it. The positions of the houses are real valued.

   Give an $O(n \log n)$-time algorithm that specifies locations for the smallest number of towers that suffice.

3. (Chopping a String.) You want to partitioning a given input string into segments in the cheapest way. Let $x_1, x_2, \ldots, x_n$ be a string. Let $C_{i,j}$ (for $i \leq j$) be the cost of a segment $x_i, \ldots, x_j$. Assume that these costs have been precomputed and given to you. The total cost of some segmentation is the sum of the costs of all segments. Let $S_j$ be the cost of the minimum cost way of breaking $x_1, x_2, \ldots, x_j$ into segments.

   (a) Give the fastest algorithm you can to compute $S_n$, and analyze its runtime.

   (b) Suppose that the cost of a segmentation also depended on characters at the boundary between successive segments. That is, suppose that if $(i, i+1)$ is a segment boundary ($i$ is the right end of one segment and $i+1$ is the left end of another segment) then there’s an additional term $B(x_i, x_{i+1})$ in the cost. Show how to solve this problem by changing your solution above.

---

\(^1\)In case this is helpful, you may assume that the priority queue can be instantiated with an arbitrary comparison oracle.
4. (How to Deal with a Little Negativity) Recall that Dijkstra’s algorithm takes a digraph and a source vertex $s$, and computes shortest paths from $s$ to all other vertices in $G$, as long as edge lengths are non-negative.

(a) Give an example where Dijkstra fails on a graph with negative length edges.

(b) Given digraph $G = (V, E)$ with source vertex $s$, with a single edge $e = (u, v)$ that has negative length, all other edges having positive edge weights, and without negative-weight cycles, show how to compute shortest paths from $s$ to all other vertices. You are allowed to run Dijkstra’s algorithm a constant number of times, plus do $O(n)$ more work.

5. (Would you Like a Sample?) You want a sample of size $k$ from the packets that go through your home router. Specifically, you want to maintain a $k$-sized set $S$ that is equally likely to be any of the $\binom{n}{k}$ possible subsets, if $n$ packets have been sent. (If $n \leq k$, then $S$ must contain all the packets.) Let’s construct an algorithm for this.

- Add the first $k$ packets to $S$.
- Upon seeing the $i^{th}$ packet (for $i \geq k + 1$), ignore this packet with probability $\underline{\hspace{2cm}}$, else, with probability $\underline{\hspace{2cm}}$, add it to $S$ and drop a uniformly random packet from $S$.

Fill in the blanks and prove your algorithm correct.