COMP 5711: Advanced Algorithms
Fall 2021 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.
5. For questions with multiple sub-questions, you can still try to solve later sub-questions even if you can't solve earlier ones.

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Total

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

   (a) (5 pts) What does it mean when we say that the running time of an algorithm is $\Omega(f(n))$? Let $T(I)$ be the running time of the algorithm on instance I. Express your answer as a mathematical/logical statement. You can use $|I|$ to denote the size of I.
   Answer: $\exists c, n_0, \forall n > n_0, \exists I, |I| = n$ and $T(I) \geq c \cdot f(n)$.

(b) (5 pts) The closest pair algorithm is a Las Vegas algorithm with expected running time $O(n)$. How do we convert it to a Monte Carlo algorithm, i.e., one that has worst-case running time $O(n)$, but may fail with probability at most $1/2$?

Answer: Assume the constant factor hidden in the expected running time is c. Our strategy is to terminate the algorithm when it performs more than $2cn$ steps. By Markov inequality, the probability that the algorithm requires more than $2cn$ steps to finish its computation is at most $1/2$. The worst-case running time is $2cn$, i.e., $O(n)$.

(c) (5 pts) For each of the following statements, indicate whether pairwise independence suffices.

i. Birthday paradox: Throw $\Theta(\sqrt{n})$ balls into n bins, we expect to see one collision.

ii. MAX-3SAT: Randomly set each variable true with probability $1/2$, then the expected number of clauses satisfied is $7k/8$.

iii. The expected cost per operation in a hash table (using chaining to resolve collisions) is $O(1)$.

iv. Random sampling: Given an array of 0's and 1's, sampling n random locations yields an estimation of the fraction of 1's with an error of $\pm\frac{3}{\sqrt{n}}$ with probability at least $8/9$.

v. Occupancy: Throw n balls into n bins, the largest bin has $O(\log n/\log\log n)$ balls with probability at least $1 - 1/n$.

Answer: For i, iii, iv, pairwise independence suffices.

2. **Dense Subgraph** (20 pts) Given an undirected graph G with n nodes and m edges, and an integer $k \leq n$, we want to find a subgraph of G with k nodes and at least $\frac{mk(k-1)}{n(n-1)}$ edges.

   (a) (10 pts) Design a polynomial-time randomized algorithm that finds a subgraph which has $\frac{mk(k-1)}{n(n-1)}$ edges in expectation.

   (b) (10 pts) Design a polynomial-time randomized algorithm that finds a subgraph with at least $\frac{mk(k-1)}{n(n-1)}$ edges with at least constant probability.

Answer: (a) The algorithm chooses k nodes randomly. For each edge $e = (u, v)$ in the original graph, the probability that it is chosen in the subgraph is the probability that both u and v are chosen, which is $\binom{n-2}{k-2}/\binom{n}{k} = \frac{k(k-1)}{n(n-1)}$. Therefore, the expected number of edges in the subgraph is $\frac{mk(k-1)}{n(n-1)}$ by linearity of expectation.

(b) To turn the algorithm into one that has high probability to find a subgraph with at least $\frac{mk(k-1)}{n(n-1)}$ edges, we need to repeat the algorithm in for enough times.

Let p_j be the probability that the number of edges in the subgraph is j. We have

$$\begin{aligned}
\frac{mk(k-1)}{n(n-1)} =& \sum_{j\geq 0} jp_j \\
=& \sum_{j<\frac{mk(k-1)}{n(n-1)}} jp_j + \sum_{j\geq\frac{mk(k-1)}{n(n-1)}} jp_j \\
\leq& (\frac{mk(k-1)}{n(n-1)} - \frac{1}{n(n-1)}) \sum_{j<\frac{mk(k-1)}{n(n-1)}} p_j + m \sum_{j\geq\frac{mk(k-1)}{n(n-1)}} p_j \\
\leq& \frac{mk(k-1)}{n(n-1)} - \frac{1}{n(n-1)} + m \sum_{j\geq\frac{mk(k-1)}{n(n-1)}} p_j
\end{aligned}$$

Therefore, we have

$$\sum_{j\geq\frac{mk(k-1)}{n(n-1)}} p_j \geq \frac{1}{n(n-1)m}.$$

We repeat the algorithm $n(n-1)m$ times. The probability that it does not find a dense subgraph is at most

$$(1 - \frac{1}{n(n-1)m})^{n(n-1)m} \leq \frac{1}{e}.$$

3. **Disk Intersection** (10 pts) In geometry, a disk is the region in a plane bounded by a circle. Given a set of n disks of radius r, design an algorithm to determine if there are two disks that intersect. Your algorithm should run in $O(n)$ time in expectation.

   Answer: We partition the plane into a grid of cells of size $r \times r$. For each disk, we map it to the cell that contains its center, thus each disk can be represented as a (cell, center) pair. Then for each disk D, we do the following: We first check the surrounding 5×5 cells. For any disk D' whose center is in one of the 25 cells, we check if D and D' intersects. If yes, we return `True` and terminate the algorithm. Otherwise, we insert D into the hash table and continue. Finally, we return `False` after all disks have been inserted.

   First, note that if two disks are mapped to the same cell, they must intersect and the algorithm will terminate before inserting the second disk into the hash table. So the hash table has at most one disk per cell. Furthermore, if two disks intersect, their centers are closer than $2r$, so one must be in the same or the surrounding 5×5 cells of another. Therefore, the algorithm is correct.

   For the running time, there are n disks in total. For each disk, checking and inserting takes $O(1)$ expected time. Thus, the algorithm takes linear time in expectation.

4. **Biased Coin** (10 pts) Suppose you are given a biased coin that has $\Pr[\texttt{HEAD}] = p \geq a$ for some $a \in (0, 1)$. You are given the value of a but not p. We would like to estimate p by flipping the coin n times. Let X_i be the random variable indicating whether the i-th flip is `HEAD`, and set $X = \sum_i X_i$. Then our estimator for p would be $\tilde{p} = \frac{X}{n}$. How large should n be so that we can guarantee $\Pr[|\tilde{p} - p| \geq \varepsilon p] \leq \delta$ for some $0 < \varepsilon, \delta < 1$? Express your answer as a function of a, ε, δ. Note that the constant does not matter.

   Answer: We know that $\mathbb{E}[X_i] = p$ and $\mathbb{E}[X] = pn$. All the X_i's are independent, so we can apply the Chernoff bounds, and have

$$\begin{aligned} Pr[|\tilde{p} - p| \geq \varepsilon p] =& Pr[X \geq (1+\varepsilon)\mathbb{E}[X]] + Pr[X \leq (1-\varepsilon)\mathbb{E}[X]] \\ \leq& \exp(-\frac{pn\varepsilon^2}{3}) + \exp(-\frac{pn\varepsilon^2}{2}) \\ \leq& 2\exp(-\frac{an\varepsilon^2}{3}) \end{aligned}$$

   Let $2\exp(-\frac{an\varepsilon^2}{3}) \leq \delta$, we get $n \geq \frac{3}{a\varepsilon^2}\ln(\frac{2}{\delta})$.

5. **Balls and Bins** (15 pts) Consider the balls-and-bins problem with m balls and n bins, where each ball is thrown to a bin uniformly and independently.

   (a) (5 pts) Prove that when $m = \sqrt{2n} + 1$, the expected number of collisions is at least 1.

   (b) (10 pts) Prove that when $m = \sqrt{2\ln(\frac{1}{\delta})n} + 1$, there is a collision with probability at least $1 - \delta$. [Hint: $1 + x \leq e^x$ for any x.]

Answer: (a) Let $Y_{i,j}$ be the random variable indicating whether the i-th ball and the j-th ball collide, so that $\mathbb{E}[Y_{i,j}] = Pr[Y_{i,j}] = \frac{1}{n}$. Let Y be the number of collision, and we have

$$\begin{aligned}
\mathbb{E}[Y] =& \mathbb{E}[\sum_{1 \leq i < j \leq m} Y_{i,j}] \\
=& \binom{m}{2} \frac{1}{n} \\
=& 1 + \frac{1}{\sqrt{2n}} \\
>& 1
\end{aligned}$$

(b) The probability that there is no collision when throwing m balls is

$$\begin{aligned}
Pr_m &= \prod_{i=0}^{m-1} (1 - \frac{i}{n}) \\
&\leq \prod_{i=0}^{m-1} \exp(-\frac{i}{n}) \\
&= \exp(-\sum_{i=0}^{m-1} \frac{i}{n}) \\
&= \exp(-\frac{m(m-1)}{2n})
\end{aligned}$$

When $m = \sqrt{2\ln\frac{1}{\delta}n} + 1$, we have (set $c = \sqrt{2\log\frac{1}{\delta}}$)

$$\begin{aligned}
Pr_m =& \exp(-\frac{c \cdot \sqrt{n}(c \cdot \sqrt{n} + 1)}{2n}) \\
\leq& \exp(-\frac{c^2}{2}) \\
=& \delta
\end{aligned}$$

6. **Morris Counter** (20 pts) Given a stream A, the problem is to count the number of elements in the stream. A trivial solution uses $O(\log n)$ bits to count n elements, while Morris counter is an approximate counting algorithm using much less space.

Algorithm 1: `MorrisCounter`

```
c ← 0;
for element i ∈ A do
    Pick y uniform over [0, 1];
    if y < 2^{-c} then
        c ← c + 1;
return 2^c − 1;
```

Let C_n be the value of c after counting n elements, and $X_n = 2^{C_n} - 1$ be the estimated count.

(a) (10 pts) Prove $\mathbb{E}[X_n] = n$ and $\mathbb{E}[X_n^2] = \frac{3}{2}n^2 - \frac{1}{2}n$. [Hint: Let $Y_n = X_n + 1$ and prove by induction.]

(b) (5 pts) Prove that $Pr[|X_n - \mathbb{E}[X_n]| \geq tn] \leq \frac{1}{2t^2}$ for any $t \geq 1$. [Hint: $Var(X) = \mathbb{E}[X_n^2] - \mathbb{E}[X_n]^2$.]

(c) (5 pts) Show that C_n has $O(\log \log n)$ bits with probability at least $1 - 1/n$.

Answer: (a) Let $Y_n = X_n + 1$ so that $Y_n = 2^{C_n}$. We have

$$\begin{aligned}
\mathbb{E}[Y_n] =& \sum_i Pr[C_n = i] \cdot 2^i \\
=& \sum_i (Pr[C_{n-1} = i-1] \cdot 2^{-(i-1)} + Pr[C_{n-1} = i](1 - 2^{-i})) \cdot 2^i \\
=& \sum_i Pr[C_{n-1} = i-1] \cdot 2 + \sum_i Pr[C_{n-1} = i] \cdot 2^i - \sum_i Pr[C_{n-1} = i] \\
=& \sum_i Pr[C_{n-1} = i] \cdot 2^i + 1 \\
=& \mathbb{E}[Y_{n-1}] + 1
\end{aligned}$$

We have $\mathbb{E}[Y_1] = 2$, therefore, $\mathbb{E}[Y_n] = n + 1$ and $\mathbb{E}[X_n] = n$.

$$\begin{aligned}
\mathbb{E}[Y_n^2] =& \sum_i Pr[C_n = i] \cdot 2^{2i} \\
=& \sum_i (Pr[C_{n-1} = i-1] \cdot 2^{-(i-1)} + Pr[C_{n-1} = i](1 - 2^{-i})) \cdot 2^{2i} \\
=& \sum_i Pr[C_{n-1} = i-1] \cdot 2^{i+1} + \sum_i Pr[C_{n-1} = i] \cdot 2^{2i} - \sum_i Pr[C_{n-1} = i] \cdot 2^i \\
=& \sum_i Pr[C_{n-1} = i] \cdot 2^{2i} + 3 \cdot \sum_i Pr[C_{n-1} = i] \cdot 2^i \\
=& \mathbb{E}[Y_{n-1}^2] + 3 \cdot \mathbb{E}[Y_{n-1}] \\
=& \mathbb{E}[Y_{n-1}^2] + 3n
\end{aligned}$$

We have $\mathbb{E}[Y_1^2] = 4$, therefore, $\mathbb{E}[Y_n^2] = \frac{3}{2}n^2 + \frac{3}{2}n + 1$ and $\mathbb{E}[X_n^2] = \mathbb{E}[(Y_n - 1)^2] = \frac{3}{2}n^2 - \frac{1}{2}n$.

(b) We can compute $Var(X) = \mathbb{E}[X_n^2] - \mathbb{E}[X_n]^2 = \frac{1}{2}n^2 - \frac{1}{2}n$, and $\sigma = \sqrt{\frac{1}{2}n^2 - \frac{1}{2}n}$, so that according to Chebyshev's inequality, we have

$$Pr[|X_n - \mathbb{E}[X_n]| \geq tn] \leq Pr[|X_n - \mathbb{E}[X_n]| \geq \sqrt{2}t\sigma] \leq \frac{1}{2t^2}.$$

(c) Set $t = \sqrt{n}$ in (b). We have that with probability at least $1 - 1/2n$, $|X_n - \mathbb{E}[X_n]| \leq n^{3/2}$, so $X_n \leq n + n^{3/2}$. When this is the case, $C_n = \log(X_n + 1)$ has at most $O(\log \log n)$ bits.

7. **Parallel Merge Sort** (10 pts) Assume there are sufficiently many processors.

   (a) (5 pts) Given two sorted arrays each of size n, design a parallel algorithm in the PRAM model to merge them into one sorted array of size n. Your algorithm should have $O(\log n)$ time and $O(n \log n)$ work.

   (b) (5 pts) Using the algorithm above as a subroutine, design a parallel version of the merge sort algorithm. Analyze the parallel time and work of your algorithm. You may assume that n is a power of 2.

Answer:

(a) For each element in each array, we need to find its target position in the merged array. For the i-th element in one array, we do a binary search in the other array. Suppose it is greater than j elements in the other array. Then its target position is $i + j$. All the binary searches can be done in parallel, taking $O(\log n)$ time and $O(n \log n)$ work. Finally, we write each element to its target location in one parallel step.

(b) We merge the n elements in a binary tree fashion bottom-up. Each node in the binary tree corresponds to the sorted sub-array consisting of all elements in its subtree. Applying the algorithm from (a) for each node, the total time is $O(\log^2 n)$ and the total work is $O(n \log^2 n)$.