COMP 5711: Advanced Algorithms
Fall 2023 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. **An Ordered Stack** (15 pts) An ordered stack is a data structure that stores a sequence of items and supports the following operations.

   - OrderedPush(x) removes all items smaller than x from the beginning of the sequence and then adds x to the beginning of the sequence.
   - Pop() deletes and returns the first item in the sequence or NULL if the sequence is empty.

   Prove that if we start with an empty ordered stack, the amortized cost of OrderedPush(x) or Pop() is $O(1)$ using *the potential method.*

   Answer: Let the potential of the data structure be the number of items in the order stack so that $\Phi(D_0) = 0$ and $\Phi(D_n) \geq 0$. For OrderedPush(x), the potential change is $\Phi(D_i) - \Phi(D_{i-1}) = -k$, where k is the number of items smaller than x in D_{i-1}, and the amortized cost is

   $$c_i' = c_i + \Phi(D_i) - \Phi(D_{i-1}) = k + 1 - k = 1.$$

   For Pop(), the potential change is $\Phi(D_i) - \Phi(D_{i-1}) = -1$, and the amortized cost is

   $$c_i' = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 1 - 1 = 0.$$

   Therefore, the amortized cost of OrderedPush(x) or Pop() is $O(1)$.

2. **The Two Largest Numbers** (25 pts) Given a sequence of n unique numbers $x_1, x_2, \ldots, x_n$, Algorithm 1 finds the two largest numbers.

```
Algorithm 1: The Two Largest Numbers
  Input  : A sequence of $n$ unique numbers $x_1, x_2, \ldots, x_n$
  Output: The two largest numbers $S_1$ and $S_2$
1 if $x_1 > x_2$ then  $(S_1, S_2) \leftarrow (x_1, x_2)$ ;
2 else  $(S_1, S_2) \leftarrow (x_2, x_1)$ ;
3 for $i = 3, \ldots, n$ do
4     if $x_i < S_2$ then  $(S_1, S_2) \leftarrow (S_1, S_2)$ ;
5     else if $x_i < S_1$ then  $(S_1, S_2) \leftarrow (S_1, x_i)$ ;
6     else  $(S_1, S_2) \leftarrow (x_i, S_1)$ ;
7 end
8 return $(S_1, S_2)$;
```

(a) (5 pts) Analyze the number of comparisons for Algorithm 1 in the worst case. Give the exact expression without using asymptotic notation.

(b) (10 pts) Design a *randomized* algorithm to find the two largest numbers using $n + O(\log n)$ comparisons in expectation.

(c) (10 pts) Show that your algorithm makes $n + O(\log n)$ comparisons in expectation.

Answer: (a) The worst-case scenario occurs when the sequence of numbers is arranged in descending order. In this situation, the algorithm requires 1 comparison for the first two numbers and 2 comparisons for each of the remaining $n - 2$ numbers. Consequently, the total number of comparisons needed amounts to $2n - 3$.

(b) The randomized algorithm simply permutes all the numbers and runs Algorithm 1 on the permuted sequence. The key observation is that after permutation, with high probability, x_i is smaller than S_2, so we can save the comparison in Line 5.

(c) For $i = 3, \ldots, n$, let $X_i = 1$ if Line 5 is executed, namely, $x_i > S_2$, at iteration i, and 0 otherwise. Since we randomly permute the sequence, the probability that x_i is the largest or the second largest element in $\{x_1, \ldots, x_i\}$ is $2/i$, and

$$\mathrm{E}[X_i] = \frac{2}{i}.$$

Then the number of comparisons in expectation is

$$1 + (n-2) + \sum_{i=3}^{n} \mathrm{E}[X_i] = 1 + (n-2) + \sum_{i=3}^{n} \frac{2}{i} = O(n \log n).$$

3. **Parameterized Algorithm** (30 pts) Given a simple, connected undirected graph $G = (V, E)$ and its tree decomposition T with width k and $|T| = O(|V|)$. You are asked to design an algorithm running in $O(f(k) \cdot |V|)$ time to find the shortest distance (each edge has length 1) between every pair of vertices u, v that belong to the same bag. Describe the algorithm and analyze the running time. You don't have to prove its correctness if it is obvious.

   [Hint: The 3 properties of a tree decomposition implies the following (you don't have to prove this): Consider any $u, v \in V$. Suppose u is in bag t_u, and v in t_v. For any path from u to v, there exists a path in T from t_u to t_v that covers the edges in the path in order.]

   Answer: For each bag t and every $u, v \in t$, we first initialize $\text{dist}(u, v)$ to be the shortest distance within the bag t. This takes $O(k^2)$ time by doing a BFS from each $u \in t$, totaling $O(k^3 \cdot |V|)$. Note that if u, v appear together in multiple bags, then we take the smallest distance in these bags.

   The main algorithm consists of two main steps. In Step 1, we traverse the tree bottom up. For each bag $t \in T$, each pair $u, v \in t$, we update

$$\text{dist}(u, v) \leftarrow \min\{\text{dist}(u, v), \text{dist}(u, v') + \text{dist}(v', v), v' \in t_c, \text{where } t_c \text{ is a child bag of } t\}.$$

   Based on the hint, after this traversal, $\text{dist}(u, v)$ is the distance from u to v using only vertices in the subtree tree below t. In Step 2, we visit the bags in top-down order using the same formula above to update $\text{dist}(u, v)$, except that t_c is replaced by the parent bag of t. Evaluating the formula takes $O(k \cdot \text{number of children of } t)$ time, which adds up to $O(k \cdot |V|)$.

   Note that other formulas also work:

$$\begin{aligned}
&\text{dist}(u, v) \leftarrow \min\{\text{dist}(u, v), \text{dist}(u, v') + \text{dist}(v', v), v' \in t\};\\
&\text{dist}(u, v) \leftarrow \min\{\text{dist}(u, v), \text{dist}(u, v') + \text{dist}(v', v), v' \in t_c \cap t \text{ for some } t_c\};\\
&\text{dist}(u, v) \leftarrow \min\{\text{dist}(u, v), 1 + \text{dist}(v', v), v' \text{ is a neighbor of } u\}.
\end{aligned}$$

4. **Maximum Satisfiability Problem** (30 pts) In Assignment 2, you designed an algorithm that satisfies at least $0.6 \cdot \mathsf{OPT}$ clauses in expectation for the MAX-SAT problem. In this question, you are asked to improve this to $(1-1/e)\cdot\mathsf{OPT} \approx 0.63\cdot\mathsf{OPT}$. [Hint: Formulate the problem as a linear program and then do randomized rounding.]

You may need the following inequalities:

$$\prod_{i=1}^{k} y_i \le \left(\frac{1}{k}\sum_{i=1}^{k} y_i\right)^k;$$

$$\text{if } w \le 1, w\cdot(1-\frac{1}{e}) \le w\cdot(1-(1-\frac{1}{k})^k) \le 1-(1-\frac{w}{k})^k.$$

Answer: Suppose we have n variables $x_1,\ldots,x_n$ and m clauses $C_1,\ldots,C_m$ where

$$C_j = \Big(\bigvee_{i\in S_j^+} x_i\Big) \vee \Big(\bigvee_{i\in S_j^-} \bar{x}_i\Big).$$

For each variable x_i, let $y_i = 1$ denote the case that x_1 is set to 1 and 0 otherwise. For each clause C_j, let $z_j = 1$ denote the case that C_j is satisfied and 0 otherwise. The maximum satisfiability problem can be modeled as the following integer linear program.

$$\begin{aligned} \max \quad & \sum_{j=1}^{m} z_j \\ \text{subject to} \quad & z_j \le \sum_{i\in S_j^+} y_i + \sum_{i\in S_j^-}(1-y_i), \forall j \\ & y_i, z_j \in \{0,1\}, \forall i,j \end{aligned}$$

We can relax it to a linear program.

$$\begin{aligned} \max \quad & \sum_{j=1}^{m} z_j \\ \text{subject to} \quad & z_j \le \sum_{i\in S_j^+} y_i + \sum_{i\in S_j^-}(1-y_i), \forall j \\ & y_i, z_j \in [0,1], \forall i,j \end{aligned}$$

Let $\{y_i^*\}_i, \{z_j^*\}_j$ denote the optimal solution to the linear program. We can then assign each variable $x_i = 1$ with probability y_i^*. For each clause C_j, let $Z_j = 1$ if clause C_j is satisfied, let k_j denote the number of literals in C_j. We have

$$\begin{aligned} \mathrm{E}[Z_j] =& 1 - \prod_{i\in S_j^+}(1-y_i^*)\prod_{i\in S_j^-} y_i^* \\ \ge& 1 - \left(\frac{1}{k_j}\Big(\sum_{i\in S_j^+}(1-y_i^*) + \sum_{i\in S_j^-} y_i^*\Big)\right)^{k_j} \\ \ge& 1 - \left(\frac{k_j - z_j^*}{k_j}\right)^{k_j} \\ \ge& z_j^*\cdot(1-\frac{1}{e}) \end{aligned}$$

The second last inequality is due to the constraints in the linear program, namely,

$$\sum_{i \in S_j^+} (1 - y_i^*) + \sum_{i \in S_j^-} y_i^* \leq k_j - z_j^*.$$

According to the linearity of expectation, the expected number of clauses satisfied is

$$\mathrm{E}[Z] = \sum_{j=1}^{m} \mathrm{E}[Z_j] \geq \sum_{j=1}^{m} z_j^* \cdot (1 - \frac{1}{e}) \geq m^*(1 - \frac{1}{e}).$$