CSCI 670 Midterm
October 6, 2021

Name:	
Student ID:	

Don't panic!

Print your name, student id, and email in the box above, and print your name on top of every page.

Please write your answers on the front of the exam pages. Use the back of the page as scratch paper. Let us know if you need more paper.

Read the entire exam before writing anything.
Make sure you understand what the question is asking.
If you give a beautiful answer to a wrong question, you'll get no credit. If any question is unclear, please ask us for clarification.

Don't spend too much time on any single problem (unless you solved all but one). If you get stuck, move on to something else and come back later.

Work on your own! Plagiarism and other anti-intellectual behavior will be dealt with severely. This includes the possibility of failing the course or being expelled from the University.

Don't panic!

Problem 1: Binary Search in Two Databases [15 Total points]

After talking CSCI 670, some of you are considering industry interviews (summer interns or potential full-time job). So, our TA, Yusuf Kalayci, kindly provides a sample problem to help you familiarize how algorithmic problems maybe formulated in industry.

Let A and B be two large databases of real numbers, both containing n elements. Suppose you are granted the privilege to perform queries of the form Access(X, k) where $X \in \{A, B\}$ and k is a positive integer, which returns the k-th smallest elements in database X.

Design an algorithm that makes $O(\log n)$ calls to Access to find the median of elements in A and B. In other words, returns the n^{th} or the $(n+1)^{st}$ smallest element of $A \cup B$ (assuming A and B has no element in common).

Solution. *Consider the following algorithm.*

Algorithm 1 FindMedian(A, B, $begin_A$, end_A, $begin_B$, end_B)

```
if end_A − begin_A ≤ 1 and end_B − begin − B ≤ 1 then
    Query remaining items and find their median.
else
    mid_A ← [(begin_A + end_A)/2]
    mid_B ← [(begin_B + end_B)/2]
    if Access(A, mid_A) > Access(B, mid_B) then
        return FindMedian(A, B, start_A, mid_A, mid_B, end_B)
    else
        return FindMedian(A, B, mid_A, end_A, start_B, mid_B)
    end if
end if
```

WLOG, assume that $|A| = |B| = n = 2^k$ *for some* $k \in \mathbb{Z}_{\geq 1}$. *Then, in each call of the function we have* $end_A - start_A = end_B - start_B$. *Observe that* $Access(A, mid_A)$ *is the median of elements in* A *which have an index between* $start_A$ *and* end_A *when* A *is sorted. Thus, if* $Access(A, mid_A) > Access(B, mid_B)$, *none of the elements in* A *with a value higher than* $A[mid_A]$ *or none of the elements in* B *with a value less than* $B[mid_B]$ *can be the median of* $A \cup B$. *Therefore, the median of the remaining elements will be the median of* $A \cup B$, *and so the problem size is shrink by* 2. *We have the runtime* $T(n) = T(n/2) + O(1)$ *where* $T(n) \in O(\log n)$.

Problem 2: Insider Trading [15 Total points]

The first company, a promising startup named DATADRIVENAI, immediately offered you a job. So you decided to abandon your Ph.D. advisor.

Upon joining DATADRIVENAI, you are offered an employee benefit to buy and then sell one share of DATADRIVENAI stock in the next n days. To your surprise and delight, the DATADRIVENAI's invisible Wall Street agent immediately tells you that the DATADRIVENAI stock prices in the next n days will be $(s_1, s_2, ..., s_n)$. But in order to ensure that SEC does not give you any problem, you need to issue an advance trading instruction with you broker, that is, you need to decide immediately which day i to buy and which day j to sell so that you can profit the most. You must issue the trading instuction immediately after you learn the insider's data. Again the only restriction is that you have to buy before you can sell, i.e., $i < j$. There is only one catch, that is, if you do not find a way to buy and sell optimally, DATADRIVENAI would take back the job offer to you to reduce your career options to (a) going back to your advisor or (b) working at HUMANINTELLIGENCE.

We will be more generous :) You can earn up to 7 total points for an algorithm that runs in $O(n^2)$ time and up to all 15 points for an algorithm that runs in $O(n)$ time. If you are using Dynamic Programming, please write down the recurrence.

1. **(60% points)**: Give an algorithm to find the best day i to buy and the best day j to sell.
2. **(30% points)**: Argue that your algorithm is correct,
3. **(10 % points)**: Establish the running time of your algorithm.

Solution. *Consider the following algorithm.*

Algorithm 2 OptimalTrading

```
B = 1, S = 1
for i = 1 to n do
    if s[i] < s[BestBuyDay] then
        BestBuyDay ← i
    end if
    if s[S] − s[B] ≤ s[i] − s[BestBuyDay] then
        B ← BestBuyDate
        S ← i
    end if
end for
Buy at B and sell at S.
```

Let $BestBuyDay_i$ be the value of $BestBuyDay$ at iteration i. Assuming that we will sell the stock at day i, then the optimal day for buying the stock is the minimum value earlier than day i which is $BestBuyDay_i$. Throughout the algorithm, we compare the cases where we sell on day i and buy on the optimal day before i, and output the optimal solution. Thus, the algorithm is optimal. Since, it is a single pass over entries of s, it works in linear time.

Problem 3: Computational Mean Value Theorem [20 Total points]

In the end, the dealing of DATADRIVENAI with Wall Street gives you some pause. So, you decided to consider more established companies..

At Microsoft Research, you are asked to solve the following algorithmic problem.

1. **[up to 15 points]** Design an algorithm for the following problem: Given a binary array $A[1...n]$, such that $A[1] = 0$, $A[n] = 1$, and $A[i] \in \{0, 1\}$, find an $0 < i < n$ such that $A[i] = 0$, $A[i + 1] = 1$.

   - If your algorithm is correct and its running time is $O(\log n)$ then you can earn all 15 points.
   - If your algorithm is correct and its running time linear in n, then you can earn at most 5 points below for correctness proof.

2. **[5 points]** Prove that your algorithm finds a correct i in the grid.

Solution. *Consider the following algorithm.*

Algorithm 3 $FindJump(A, begin, end)$

```
if end = begin + 1 then
    Output: begin
else
    mid ← (begin + end)/2
    if A[mid] == 0 then
        Output: FindJump(A, mid, end)
    else
        Output: FindJump(A, begin, mid)
    end if
end if
```

First of all, observe that the run time of the algorithm is $O(\log n)$ (similar to binary search). We can prove by induction that the algorithm always outputs the correct index. The following lemma completes the discussion.

Lemma. *If $A[begin] = 0$ and $A[end] = 1$, the algorithm outputs the correct index.*

Proof. We can apply induction on the sequence length $n = end - begin + 1$. Base case: when $n = 2$, it clearly outputs the correct output. Assume the theorem is correct for any $n > 2$. Let $end - begin = n + 1$, then consider the $mid = (end + begin)/2$. If $A[mid] = 0$, then $A[mid : end]$ is a sequence starting with 0 and ending 1 whose length is less than equal to k. Similarly, if $A[mid] = 1$, then $A[begin : mid]$ is a sequence starting with 0 and ending 1 whose length is less than equal to k. Thus, algorithm always find such an index for $end - begin = n + 1$ as well. □

Problem 4: Graph Influence [25 Total Points]

During your interview with Twitter, you are asked about a simple question regarding graph influence.

Background: Suppose a directed graph $G = (V, E)$ models the Twitter interaction graph, where nodes in V represent its members and edges in E represent the "following relations" among its members.

Influence Structures: Building on the *six-degrees-of-separation* hypothesis,[1] the Twitter manager interviewing you considers the influence-sphere of a node $u \in V$ to be the set of all nodes that can reach u in **two or less** following relations. More generally, for any subset $S \subseteq V$, let

$$\text{INFLUENCE-SPHERE}_G(S) := \{ \; v \in V : \exists u \in S \text{ and a path of at most two "following relations" from } v \text{ to } u \text{ in } G\}$$

Note: $S \subseteq \text{INFLUENCE-SPHERE}_G(S)$, $\forall S$.

Part 1 (10 points): For all $S \subseteq V$, let

$$\text{INFLUENCE-POWER}_G(S) = |\text{INFLUENCE-SPHERE}_G(S)|$$

Prove that INFLUENCE-POWER_G is submodular.

Solution. *Let $N_2(u)$ be the set of vertices two step away to u. Consider subsets $S \subseteq T \subseteq V$ and $u \in V \setminus T$. Then,*

$$\text{INFLUENCE-POWER}_G(S + \{u\}) - \text{INFLUENCE-POWER}_G(S) = |N(u) \setminus N(S)|.$$

Observe that $N(S) \subseteq N(T)$ since $S \subseteq T$. So,

$$|N(u) \setminus N(S)| > |N(u) \setminus N(T)|$$

Thus,

$$\text{INFLUENCE-POWER}_G(S \cup \{u\}) - \text{INFLUENCE-POWER}_G(S) \geq \text{INFLUENCE-POWER}_G(T \cup \{u\}) - \text{INFLUENCE-POWER}_G(T).$$

$\text{INFLUENCE-POWER}_G(\cdot)$ *is a submodular function.*

[1] By Wikipedia: *Six-degrees-of-separation* is the idea that all people are six, or fewer, social connections away from each other. As a result, a chain of "a friend of a friend" statements can be made to connect any two people in a maximum of six steps. It was originally set out by Frigyes Karinthy in 1929 and popularized in an eponymous 1990 play written by John Guare. It is sometimes generalized to the average social distance being logarithmic in the size of the population.

Part 2 (10 points): Design a polynomial-time approximation algorithm that, when given G and an integer k, identifies a set of k nodes in V that has INFLUENCE-POWER at least $(1 - 1/e)$ of the maximum possible INFLUENCE-POWER achievable by k nodes. Present your approximation algorithm. What is the running time of your algorithm? Give the main reason why your algorithm has approximation ratio $(1 - 1/e)$.

Solution. *Consider the following algorithm.*

Algorithm 4 $FindInfluencer(G, k)$

Let $IP_G(S) :=$ INFLUENCE-SPHERE$_G(S)$
$S \leftarrow \emptyset$
for $i = 1$ to k **do**
  Let $v^* \leftarrow argmax\{IP_G(S \cup \{v\}) - IP_G(S) \mid v \in V\}$
  $S \leftarrow S \cup \{v^*\}$
end for
Output: S

Observe that $IP_G(S)$ can be computed in $O(n^2)$ time. We can find the optimal vertex by computing the marginal influence power for each vertex in $O(n^3)$ time. Thus the overall runtime is $O(n^3 \cdot k)$.

Observe that this is the greedy algorithm we discussed in class. Thus, it is $\left(1 - \frac{1}{e}\right)$-approximation, thus we omit the proof.

Part 3 (5 points): Fresh from your USC CSCI 670 classes, you gracefully solved Part 1 and 2. But at this moment, another Twitter manager comes in and begins to debate with the first manager. The second manager thinks that the INFLUENCE-POWER of a node should be defined as its out-degree in G, and the INFLUENCE-POWER of a subset $S \subseteq V$ should be defined as the number of nodes either in S or follows someone in S. Let's denote this number by $\Delta_G(S)$. The two managers eveutually comprise so that you can complete your interview. They define the new influence-power of S as

$$\text{INFLUENCE-POWER}_G(S) + 2\Delta_G(S).$$

They ask you to demonstrate the following: Suppose $f : 2^V \to \mathbb{R}^+ \cup \{0\}$ and $g : 2^V \to \mathbb{R}^+ \cup \{0\}$ are two submodular set functions with ground set V. Prove that $f + 2g$ is also submodular.

Solution. *Let $S \subseteq T \subseteq V$, and $e \in V \setminus T$. Then, we have*

$$f(S \cup \{e\}) - f(S) \geq f(T \cup \{e\}) - f(T)$$

and,

$$g(S \cup \{e\}) - g(S) \geq g(T \cup \{e\}) - g(T).$$

Adding the two inequalities by multiplying with the right constant we obtain

$$(f(S+e)+2g(S+e))-(f(S)+2g(S)) \geq (f(T+e)+2g(T+e))-(f(T)+2g(T)).$$

So, $f + 2g$ is also a submodular function.

Problem 5: Netflix Interview [25 Total points]

During your Netflix interview, you are asked to solve an algorithmic problem regarding the following notion of *preference*:

(Background): In the Netflix service, a viewer can assign a score to a movie from the real interval $[0, 1]$, indicating how much she or he likes the movie, where 1 and 0, respectively, represent the highest and lowest preferences. Suppose the current database of Netflix contains n movies $M = \{m_1, ..., m_n\}$, and an energic viewer Gavin (who only gave his first name) watched and scored all movies: he assigned $W(m_i)$ to movie m_i. In addition, suppose Netflix has k disjoint categories $(G_1, ..., G_k \subset M)$ of movies of equal size, i.e., for all $i \neq j$, $G_i \cap G_j = \emptyset$ and $|G_i| = |G_j| = n/k$.

Now, in building Gavin's profile, for any two categories G_i, G_j, Netflix defines that Gavin *lexicographically prefers* G_i over G_j if there exists a bijection $f : G_i \to G_j$ such that for all $m \in G_i$, $W(m) \geq W(f(m))$.

- **(Upto 18 points)** Design an algorithm that can determine if Gavin lexicographically prefers category G_i over G_i in time polynomial in $|G_i|$ and analyze the running time of your algorithm.
  - If your algorithm is correct and its running time is $O(|G_i| \log |G_i|)$, then you can earn all 18 points.
  - If your algorithm is correct and its running time is quadratic in $|G_i|$, then you will earn at most 5 points.
- **(7 points)** Prove that your algorithm is correct.

Solution. • *Consider the following algorithm.*

Algorithm 5 *LexCheck*(G_i, G_j)

Set $\ell := n/k$
Sort $G_i = \{m_{i_1}, \ldots m_{i_\ell}\}$ so that $W(m_{i_t}) \leq W(m_{i_{t+1}})$ for any $1 \leq t < \ell$.
Sort $G_j = \{m_{j_1}, \ldots m_{j_\ell}\}$ so that $W(m_{j_t}) \leq W(m_{j_{t+1}})$ for any $1 \leq t < \ell$.
for $t \in \{1 \ldots \ell\}$ **do**
  if $W(m_{i_t}) < W(m_{j_t})$ **then**
    Output: False
  end if
end for
Output: True

Observe that the runtime of the algorithm is $O(\ell \cdot \log \ell)$ *dute sorting the elements of* G_i *and* G_j*.*

- *By an exchange argument, one can show that if greedy mapping does not imply that Gavin prefers* G_i *over* G_j*, then no map can satisfy the desired property. So, Gavin does not prefer* G_i *over* G_j*.*