# CPSC 320 2016W2, Midterm #1

February 13, 2017

First page replaced by cover

Second page replaced by rules

## 1 GradeScope Student # [1 mark]

Please enter your three-digit GradeScope Student # in the following blank: __ __ __.

## 2 O'd to a Pair of Runtimes [8 marks]

Consider the following pairs of functions representing runtimes of algorithms that take a **directed** graph with $n$ vertices and $m$ edges as the input. **ASSUME both $n$ and $m$ are greater than one.** For each pair, write between them the best choice of:

**LEFT:** to indicate that the left one is big-$O$ of the right one, i.e., left $\in$ O(right)

**RIGHT:** to indicate that the right one is big-$O$ of the left one, i.e., right $\in$ O(left)

**SAME:** to indicate that two are $\Theta$ of each other, i.e., left $\in \Theta$(right)

**INCOMPARABLE:** to indicate that none of the previous relationships holds for all allowed values of $n$ and $m$.

Do not write **LEFT** or **RIGHT** if **SAME** is true. The first one is filled in for you.
**[1 mark per problem]**

| Left Function | **LEFT**, **RIGHT**, **SAME**, or **INCOMPARABLE** | Right Function |
|---|---|---|
| $n$ | **LEFT** | $n^2$ |
| $n^3$ | | $\frac{n^2(n+\lg n)}{10}$ |
| $\frac{m}{\lg m}$ | | $\frac{m}{\sqrt{m}}$ |
| $n+m$ | | $nm$ |
| $n \lg n$ | | $\lg(n^m)$ |
| $n^{10}$ | | $2^n$ |
| $n+m^2$ | | $m+n^2$ |
| $\log_3(n+15)$ | | $\log_8(10n)$ |
| $m^{\lg m}$ | | $m^2$ |

## 3 eXtreme True And/Or False [15 marks]

Each of the following statements may be **always** true, **sometimes** true, or **never** true. Select the best of these three choices and then:

- If the statement is **always** true, (1) give and very briefly explain an example instance in which it is true and (2) sketch the key points of a proof that it is always true.
- If the statement is **never** true, (1) give and very briefly explain an example instance in which it is false and (2) sketch the key points of a proof that it is never true.
- If the statement is **sometimes** true, (1) give and very briefly explain an example instance in which it is true and (2) give and very briefly explain an example instance in which it is false.

Note that you will always select a choice and then give two answers. There is space for this below.
**[5 marks per part]**

1. The prerequisite structure of the courses in a **reasonable** university forms a directed acyclic graph (where courses are nodes and there is an edge leading to a course from each of its prerequisites).

   Circle **one**: **ALWAYS** **SOMETIMES** **NEVER**

   (1)

   (2)

2. In a SMP problem (with $n > 1$) in which every man except man $m$ ranks a particular woman $w$ last, while $m$ ranks $w$ first, there is a stable matching in which $w$ marries $m$.

   Circle **one**: **ALWAYS** **SOMETIMES** **NEVER**

   (1)

   (2)

3. A single edge contraction on a previously disconnected graph $G$ results in a new graph that is connected.

   Circle **one**: **ALWAYS** **SOMETIMES** **NEVER**

   (1)

   (2)

**The rest of the page is intentionally blank.**

**If you write answers below, CLEARLY indicate here what question they belong with AND on that problem's page that you have answers here.**

## 4 You Have Chosen. . . Randomly [11 marks]

Consider the following problem that we call YHC. (As a side note, it's related to the problem of choosing an item at random with a likelihood proportional to its frequency.) Given:

1. a list $L = [(i_1, c_1), \ldots, (i_n, c_n)]$ of $n > 0$ pairs, where each $i$ is an item and each $c$ is that item's (positive integer) count, and
2. a number $t \in [1, C]$, where $C$ is the total count of the items $\sum_{j=1}^{n} c_j$,

produce the smallest number $j$ for which the sum of the first $j$ items' counts is at least $t$. That is, the first j such that $(\sum_{k=1}^{j} c_k) \geq t$.

For instance, given the list $L = [(a, 2), (b, 1), (c, 1), (d, 4)]$ and $t = 1$, we would choose $j = 1$ (item $a$) since $2 \geq 1$ but 0 is not. Given $t = 4$ instead, we would choose $j = 3$ (item $c$) since $2 + 1 + 1 \geq 4$ but $2 + 1$ is not. Given $t = 5$, we would choose $j = 4$ (item $d$) since $2 + 1 + 1 + 4 \geq 5$ but $2 + 1 + 1$ is not.

1. Complete the pseudocode function below with a worst-case $O(n)$ "brute force" algorithm to solve YHC. **[4 marks]**

```
procedure FindJ($L$, $t$)



end procedure
```

2. It isn't really the individual counts that we need to solve this problem but the running sums of the counts. That is, for item $i$, we don't want $c_i$ but $\sum_{j=1}^{i} c_j$. In the example list above, for index 3 (item $c$), we don't want its count 1 but the total count up to index 3: $2 + 1 + 1 = 4$.

   Complete the logarithmic-time algorithm below that—given a non-empty list $L'$ of pairs of items and their running sums (rather than their counts) and a number $t \in [1, C]$ where $C$ is the final running sum in $L'$—produces the first index $j$ at which the running sum is at least as large as $t$. **NOTE:** We assume 1-based indexing, i.e., that the indexes of $L'$ are $1, 2, \ldots, n$. **[5 marks]**

```
procedure FindJ($L'$, $t$)
    $n \leftarrow$ Length($L'$)
    if $n = 1$ then
        return ______________
    else
        mid $\leftarrow \lfloor \frac{n}{2} \rfloor$
        if __________________________________________ then
            $j' \leftarrow$ FindJ($L'[1 \ldots \text{mid}]$, $t$)
            return ______________
        else
            $j' \leftarrow$ FindJ($L'[\text{mid}+1 \ldots n]$, $t$)
            return ______________
        end if
    end if
end procedure
```

3. To use your algorithm, we first need to produce the running sums of the counts. **Briefly** answer these **two** questions:

   (a) Give and **very briefly** justify a good worst-case asymptotic bound on the runtime of (efficiently) generating the running sums and then calling an $O(\lg n)$ algorithm to solve the problem. **[1 mark]**

   (b) Now, imagine that we need to solve the YHC problem $\Theta(n)$ times for the same list (but different values of $t$). Give and **very briefly** justify a good worst-case asymptotic bound on the runtime of this running sum algorithm. **[1 mark]**

## 5 Greedy banks resequencing debits [10 marks]

Recall the "Greedy banks resequencing debits" problem, repeated here verbatim:

> Predatory banks take the debits to an account that occur over the day and reorder them to maximize the fees they can charge. For each debit that results in taking an account into overdraft (having negative balance in the account) or where the account is already in overdraft, the bank charges the customer an overdraft fee.

So, for example, you may have spent \$3, \$7, \$2, and \$1 in a day when your account balance started at \$4. If we keep the debits in the order they're given, the \$3 debit takes your balance to \$1. The \$7 debit takes your balance to -\$6 and is the first to go into overdraft. All subsequent debits are also in overdraft. The \$2 debit takes your balance to -\$8. The \$1 debit takes your balance to -\$9.

Imagine that the overdraft fee on a debit of $d$ dollars that goes into overdraft by $k \leq d$ dollars is $\frac{k^2}{100}$. ($k$ is the amount of the debit that cannot be paid. So, for the \$7 debit in the example above, $k = 6$. For the \$2 debit, $k = 2$, and for the \$1 debit, $k = 1$.)

In this problem, the bank wants to generate an order of the debits that will maximize the fees it collects.

1. Give a small but non-trivial instance of the problem along with its optimal solution and the value (total fees) of that solution. **[2 marks]**

2. Give pseudocode for a very simple greedy algorithm that solves the problem optimally in $O(n \lg n)$ time for $n$ debits. **[1 mark]**

3. Complete the following proof of the correctness of your algorithm. **[4 marks]**

   We compare the greedy algorithm's debit ordering $\mathcal{G}$ against an optimal ordering $\mathcal{O}$. If $\mathcal{O}$ and $\mathcal{G}$ are the same, then $\mathcal{G}$ optimal. Otherwise, let $d_i$ and $d_{i+1}$ be a pair of neighboring debits that are in one order in $\mathcal{O}$ and the opposite order (and not necessarily neighbouring) in $\mathcal{G}$. Let the total of all the debits before $d_i$ in O be $T$ and the initial account balance be $B$. We now show that we can swap $d_i$ and $d_{i+1}$ without decreasing the overall value of the solution in four cases:

   **Case 1, $B \leq T$ (i.e., both debits are entirely in overdraft):** After swapping, both are still entirely in overdraft and so contribute the same amount to the fees collected.

   **Case 2, $T < B < T + d_i$ (i.e., $d_i$ is the debit that takes the account into overdraft)**
   **YOU ARE NOT REQUIRED TO COMPLETE THIS CASE OF THE PROOF** (but see the bonus problems)

   **Case 3, $T + d_i \leq B < T + d_i + d_{i+1}$ (i.e., $d_{i+1}$ is the debit that takes the account into overdraft)**

   **Case 4, $T + d_i + d_{i+1} \leq B$ (i.e., neither debit is in overdraft)**

   Thus, we can start from $\mathcal{O}$ and repeatedly swap neighbouring debits that are in the opposite order to their order in $\mathcal{G}$ until the solution is identical to $\mathcal{G}$ without reducing the value (fees) of the solution, and so $\mathcal{G}$ is optimal. QED

4. Imagine the fee were $\frac{k}{100}$ instead instead of $\frac{k^2}{100}$. A friend proposes to you an algorithm that resequences the debits. Without knowing the details of the proposal, what can you say about how close that algorithm's result is to the optimal result? **Briefly and clearly justify your answer. [3 marks]**

## 6 Access Allowed [5 marks]

Recall the "Access Allowed" problem, repeated here verbatim:

> Based on accessibility guidelines, an institution has classified all of its campus's pathways connecting points of interest into three groups, those that are: at recommended specifications (R), at minimum specifications (M), and below minimum specifications (X). For each pathway that is rated M or X, you also have a cost to upgrade to the higher specification(s). Occasionally, that cost is indicated at $\infty$ where it's considered impossible to perform the upgrade.
>
> Note that a "pathway" may take any of various forms (e.g., a flight of steps), but that it will always connect exactly two points of interest. Furthermore, although two points of interest may be physically connected "in the real world" by multiple pathways (e.g., a flight of stairs and an elevator), the "logical" pathway will appear only once in the data rated according to the most accessible of the physical pathways connecting the two points.

The R Bridge Problem (RBP) is the problem of selecting the least expensive plan to upgrade pathways so that all points are connected to all other points using only pathways rated at R. Specifically, a solution to RBP is the set of M and X edges that should be upgraded.

**Give a correct and efficient reduction from RBP to MST** and **very** briefly explain why your reduction works.

- Recall that the MST (minimum spanning tree) problem is: given a weighted, undirected graph, find the minimum weight subset of the graph's edges that connects the graph's vertices into a (spanning) tree.
- Recall that you will need to explain both how you will transform an instance of RBP into an instance of MST and how you will transform the corresponding solution to MST into a solution to your problem.

## 7 BONUS: From the Cutting-Room Floor [2 BONUS marks]

This is a section filled with problems that are **too hard for the amount of points they're worth**. Each is worth $\frac{1}{2}$ of a bonus point. Your total earned bonus points—on this midterm exam and toward the course's bonus point reward program—is your total score here, **rounded down**. We will be ridiculously harsh marking these. **Don't waste your time here!**

For your (in)convenience, we've rated—and sorted—the problems from "hard-ish" to "hardest-ish".

1. Complete **Case 2** of the proof of correctness of your greedy strategy from Greedy banks resequencing debits. Rating: **Hard-ish**.

*Follow the eXtreme True And/Or False rules on the remaining problems.*

2. The diameter of an unweighted, undirected graph (with at least one vertex) is **more than** twice the height of a BFS tree rooted at an arbitrary node in that graph. Rating: **Harder-ish**.

   Circle **one**: **ALWAYS** **SOMETIMES** **NEVER**

   (1)

   (2)

3. In a debit resequencing problem where (1) the fee is 10% of the entire amount of each debit that goes into overdraft, and (2) there are at least two debits, there is an optimal solution where the largest debit is the one that causes the account to go into overdraft. Rating: **Harder-ish**.

   Circle **one**: **ALWAYS** **SOMETIMES** **NEVER**

   (1)

   (2)

4. When we perform BFS on an unweighted, undirected graph with at least two vertices and diameter $d$, at least one vertex from the lowest layer of the BFS tree must itself have a BFS tree of height $d$. (I.e., at least one vertex in the bottom layer is "on the diameter" of the graph.) Rating: **Hardest-ish**.

   Circle **one**: **ALWAYS** **SOMETIMES** **NEVER**

   (1)

   (2)

**This page intentionally left (almost) blank.**
**If you write answers here, you must CLEARLY indicate on this page what question they belong with AND on the problem's page that you have answers here.**

**This page intentionally left (almost) blank.**
**If you write answers here, you must CLEARLY indicate on this page what question they belong with AND on the problem's page that you have answers here.**