CSC265 F23: Assignment 1

Due: September 29, by midnight

Guidelines: (read fully!!)

- Your assignment solution must be submitted as a *typed* PDF document. Scanned handwritten solutions, solutions in any other format, or unreadable solutions will **not** be accepted or marked. You are encouraged to learn the LaTeX typesetting system and use it to type your solution. See the course website for LaTeX resources.
- Your submission should be no more than 6 pages long, in a single-column US Letter or A4 page format, using at least 10 pt font and 1 inch margins.
- To submit this assignment, use the MarkUs system, at URL `https://markus.teach.cs.toronto.edu/2023-09`
- This is a *group assignment.* This means that you can work on this assignment with *at most one other* student. You are encouraged to work with a partner. Both partners in the group should work on and arrive at the solution together. Both partners receive the same mark on this assignment.
- Work on all problems together. For each problem, one of you should write the solution, and one should proof-read and revise it. The first page of your submission must list the *name*, *student ID*, and *UTOR email address* of both group members. It should also list, for each problem, which group member wrote the problem, and which group member proof-read and revised it.
- You **may not** consult any other resources except: your partner; your class notes; your textbook and assigned readings. *Consulting any other resource, or collaborating with students other than your group partner, is a violation of the academic integrity policy!*
- You may use any data structure, algorithm, or theorem previously studied in class, or in one of the prerequisites of this course, by just referring to it, and without describing it. This includes any data structure, algorithm, or theorem we covered in lecture, in a tutorial, or in any of the assigned readings. Be sure to give a *precise reference* for the data structure/algorithm/result you are using.
- Unless stated otherwise, you should justify all your answers using rigorous arguments. Your solution will be marked based both on its completeness and correctness, and also on the clarity and precision of your explanation.

Question 1. (8 marks)

In the following code, the subroutine F takes as input an integer value x, and a matrix (two-dimensional array) $M[1..n][1..n]$, where the rows and columns are indexed from 1 through n, and the entries are integers.

```
FUNC($x, M$)
i = 1
j = n
while $i \leq n$ and $j \geq 1$
    if $M[i][j] = x$
        return $(i, j)$
    elseif $M[i][j] < x$
        $i = i + 1$
    elseif $M[i][j] > x$
        $j = j - 1$
return $\perp$
```

Part a. (4 marks)

Give a tight asymptotic upper bound in terms of n on the worst-case running time of FUNC(x, M). Briefly justify your answer.

Part b. (4 marks)

Show that your upper bound above is tight by describing, for each integer $n > 1$, an integer x, and a matrix $M[1..n][1..n]$ with integer entries, for which the running time of FUNC(x, M) is asymptotically at least as large as the upper bound. Moreover, ensure that the entries of M are sorted in non-decreasing order going left to right in each row, and top to bottom in each colum. I.e., in each M you define, it should be the case that $M[i][j] \leq M[i+1][j]$ and $M[i][j] \leq M[i][j+1]$ whenever all indexes are within range. Briefly justify your answer.

Question 2. (12 marks)

Consider the following algorithm, which takes as input an array $A[1..m]$ (i.e., an array indexed by the integers from 1 to m), for some integer $m > 1$. Each element $A[i]$ of the array is a tuple with two fields, $A[i].id$ and $A[i].count$. $A[i].id$ is an integer in $\{1, \ldots, n\}$, for an integer $n > 1$. $A[i].count$ is a positive integer. The algorithm also takes as input a positive integer parameter k. You can think of $A[i].id$ as an identifier, e.g., a user ID, and $A[i].count$ as some count associated with the identifier, e.g., how many minutes a user with that ID spent logged into a website in one session. The algorithm below can then give a very rough approximation of the total time spent by a user logged into a website in total over different sessions. The larger k is, the better the approximation.

```
FUNC($A, k$)
$S = \emptyset$
for $i = 1$ to $m$
    if $\exists x \in S$ such that $x.id = A[i].id$
        $x.count = x.count + A[i].count$
    elseif $|S| \leq k - 1$
        Create an element $x$ with $x.id = A[i].id$ and $x.count = A[i].count$
        $S = S \cup x$
    else Let $x \in S$ be such that $x.count = min\{y.count : y \in S\}$
        $x.id = A[i].id$
        $x.count = x.count + A[i].count$
Return $S$
```

In words, the algorithm goes through the entries of A while keeping a set S of size at most k. When a new element $A[i]$ is being processed, the algorithm first checks if an element with that ID is already in S, and, if it is, it adds $A[i]$'s count to the count of the existing element in S. If $A[i]$ is not found in S, and S has size less than k, we add it to S. If $A[i]$ is not found in S, and S has reached its maximum size k, we replace the ID of the smallest count element in S with the ID of $A[i]$, and we add the count of $A[i]$ to the existing smallest count.

Part a. (6 marks)

Show that this algorithm can be implemented so that it runs in worst-case time complexity $O(n + m \log k)$. I.e., describe an algorithm that outputs the same set S as $\textsc{Func}(A, k)$ when given the same input A, k, and runs in worst-case time complexity $O(n + m \log k)$. Justify your answer.

Hint: One of the data structures you use should be an array keeping track of which elements are in S.

Part b. (6 marks)

Suppose that each $A[i].\mathit{count}$ equals 1. Show that, in this case, $\textsc{Func}(A, k)$ can be implemented to run in worst case time complexity $O(n+m)$. I.e., describe an algorithm that outputs the same set S as $\textsc{Func}(A, k)$ when given the same input A, k, and runs in worst-case time complexity $O(n + m)$.

Hint: Consider using a data structures based on linked lists