CS 6363.005 Midterm 1—Problems and Instructions

February 26, 2019

Please read the following instructions carefully before you begin.

- Write your name and Net ID on the *answer sheets* cover page and your Net ID on each additional page. Answer each of the four questions on the answer sheets provided. One sheet was intentionally left blank to provide you with scratch paper.
- Questions are not necessary given in order of difficulty, so read through them all before you begin writing!
- This exam is closed book. No notes or calculators are permitted.
- You have one hour and 15 minutes to take the exam.
- Please turn in these problem sheets, your answer sheets, and scratch paper at the end of the exam period.
- If asked to describe an algorithm, you should state your algorithm clearly (preferably with pseudocode) and briefly explain its asymptotic running time in big-O notation in terms of the input size.
- Feel free to ask for clarification on any of the problems.
- ***This exam does not cover greedy algorithms, so do not use them.***

1. Answer each of the following questions. There is no need to justify your answers for these questions.

   (a) **(2.5 out of 10)** Using Θ-notation in terms of n, what is the solution to the recurrence $T(n) = 3T(n/3) + n$?

   (b) **(2.5 out of 10)** Using Θ-notation in terms of n, what is the solution to the recurrence $T(n) = 3T(n/2) + n^2$?

   (c) **(2.5 out of 10)** Using Θ-notation in terms of n, what is the solution to the recurrence $T(n) = T(7n/9) + T(n/9) + n$?

   (d) **(2.5 out of 10)** Consider the following recursive function which is defined in terms of a fixed array $X[1 .. n]$.

$$WTF(i,j) = \begin{cases} 0 & \text{if } i \le 0 \text{ or } j \le 0 \\ X[j] + WTF(i-1,j) + WTF(i,\lfloor j/2 \rfloor) & \text{otherwise} \end{cases}$$

   Using Θ-notation in terms of n, how long does it take to compute $WTF(n,n)$ using dynamic programming?

2. (a) **(5 out of 10)** Consider the following generalization of the Blum-Floyd-Pratt-Rivest-Tarjan SELECT algorithm we discussed in class. It partitions the input array into $\lceil n/7 \rceil$ blocks of size 7 instead of $\lceil n/5 \rceil$ blocks of size 5 but is otherwise identical.

```
MOM7SELECT(A[1 .. n], k):
  if n ≤ 49
      use brute force
  else
      m ← ⌈n/7⌉
      for i ← 1 to m
          M[i] ← MEDIANOF7(A[7(i−1)+1 .. 7i])
      mom7 ← MOM7SELECT(M[1 .. m], ⌊m/2⌋)

      r ← PARTITION(A[1..n], mom7)

      if k < r
          return MOM7SELECT(A[1 .. r−1], k)
      else if k > r
          return MOM7SELECT(A[r+1 .. n], k−r)
      else
          return mom7
```

   We obtained a worst-case run time recurrence of $T(n) \le T(n/5) + T(7n/10) + n$ for the traditional blocks of size 5 version based on the observation that there are at at least $3n/10$ elements smaller than the median-of-medians.

   State a worst-case running time recurrence for MOM$_7$SELECT. Give a brief justification for your answer.

(b) **(5 out of 10)** Recall the edit distance problem: We're given two strings $A[1 .. n]$ and $B[1 .. n]$, and we're asked for the minimum number of insertions, deletions, and substitutions needed to transform A into B.

Consider a version of the problem where different operations have different costs; inserting a character into A has cost 2, deleting a character from A has cost 3, and substituting one character of A for a different character from B has cost 1.

Let $Edit2(i, j)$ be the minimum total cost of any set of insertions, deletions, and substitutions that transform $A[1 .. i]$ into $B[1 .. j]$. Fill in the blanks to complete the following recursive definition of $Edit2(i, j)$. There is no need to justify your answer.

$$Edit2(i,j) = \begin{cases} 3i & \text{if } j = 0 \\ \_\_\_\_\_ & \text{if } i = 0 \\ \min \left\{ \begin{array}{r} Edit2(i, j-1) + \_\_\_\_\_ \\ Edit2(i-1, j) + \_\_\_\_\_ \\ Edit2(i-1, j-1) + \_\_\_\_\_ \cdot [A[i] \neq B[j]] \end{array} \right\} & \text{otherwise} \end{cases}$$

3. **(10 points)** Suppose you are given a sorted array of n distinct numbers that has been *rotated* k steps, for some ***unknown*** integer k between 1 and $n-1$. That is, you are given an array $A[1..n]$ such that the prefix $A[1..k]$ is sorted in increasing order, the suffix $A[k+1..n]$ is sorted in increasing order, and $A[n] < A[1]$.

   For example, you might be given the following 16-element array (where $k = 10$):

9	13	16	18	19	23	28	31	37	42	−4	0	2	5	7	8

   Describe and analyze a divide-and-conquer/binary search algorithm to compute the unknown integer k in $O(\log n)$ time. You should briefly justify the correctness and running time of your algorithm.

4. You decide to play a simple solitaire card game. At the beginning of the game, the cards are dealt face up in a long row. Each card is worth a different number of points. On each turn, you have a choice of *taking* or *discarding* the leftmost card. If you decide to take the leftmost card, then you collect the number of points the card is worth. However, you must immediately discard the *three* cards immediately to its right before you begin your next turn. Your goal is to collect as many points as possible.

   For this problem, you may assume the input is given as an array $P[1 .. n]$ where the ith card from the left has point value $P[i]$.

   (a) **(5 out of 10)** Let $MaxPoints(i)$ be the maximum number of points you can collect using only cards i through n ordered left to right.

   Give a recursive definition, including the base case(s), for $MaxPoints(i)$. You should *briefly* explain each of the cases in your recursive definition.

(b) **(5 out of 10)** Describe and analyze an efficient dynamic programming algorithm to determine the maximum number of points you can collect given the initial set of n cards. Stating 1) the evaluation order for computing each $MaxPoints(i)$ value, 2) which value to return as the algorithm's output, and 3) the running time of your algorithm is enough for full credit.