University of Waterloo
CS240R Fall 2018
Midterm Help Problems

Reminder: Midterm on Thursday, October 18 2018

Note: This is a sample of problems designed to help prepare for the midterm exam. These problems do *not* encompass the entire coverage of the exam, and should not be used as a reference for its content.

True/False

Indicate "True" or "False" for each of the statements below.

a) If $T_1(n) \in O(f(n))$ and $T_2 \in O(g(n))$, then $\frac{T_1(n)}{T_2(n)} \in O\left(\frac{f(n)}{g(n)}\right)$.

b) If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = e^{42}$, then $f(n) \in \Theta(g(n))$.

c) If $f(n) \in o(n \log n)$, then $f(n) \in O(n)$.

d) The smallest element of a max-heap can be found in $o(n)$ time.

e) HeapSort is valid as the auxiliary digit-sorting algorithm for MSD-RadixSort.

f) Randomized QuickSort on a sorted list of distinct integers has an expected runtime of $\Theta(n^2)$.

g) All heaps satisfy the AVL height-balance requirement.

h) An AVL tree with 7 nodes must have a height of 2.

i) Given an AVL tree, we can generate a sorted array of keys in $o(n \log n)$ time.

j) The probability of a skip list tower having a height of exactly 1 is $\frac{1}{4}$.

Runtime Analysis

After assignment grading is complete, Henry wants to make sure all the grades are okay, and have them listed in sorted order. He assigns either Benjamin or Sajed to perform this task. Benjamin and Sajed have many sorting algorithms to choose from, so their decision is based on their mood. Their moods fluctuate while they check assignment grades, and eventually depends on the final student's grade relative to the other grades. Assignment grades are integers ranging from 0 to 100.

```
BenjaminSort (A):
    mood := verify(A)
    if mood = +1
        HeapSort(A)
    else if mood = -1
        SelectionSort(A)
    else MSDRadixSort(A, 10)
```

```
SajedSort (A):
    mood := verify(A)
    if mood = +1
        SelectionSort(A)
    else if mood = -1
        LSDRadixSort(A, 10)
    else MergeSort(A)
```

The function `verify(A)` runs in $\Theta(n)$ time and returns:

- $+1$, if the final element of A is the maximum element of the array,
- -1, if the final element of A is the minimum element of the array,
- 0, otherwise.

a) What is the best-case, worst-case, and average-case runtime for `BenjaminSort`?

b) What is the best-case, worst-case, and average-case runtime for `SajedSort`?

c) Henry's decision, which we refer to as `HenrySort`, is to flip a coin. If it flips heads, then he assigns the task to Benjamin (`BenjaminSort`). Otherwise, if it flips tails, then he assigns the task to Sajed (`SajedSort`). What is the best-case expected runtime, worst-case expected runtime, and average-case expected runtime for `HenrySort`?

d-ary Heaps

Suppose instead of binary heaps, we have d-ary heaps, where each internal node contains d children, except possibly the last one.

a) What is the height of a d-ary heap of n nodes?

b) Suppose the d-ary heap is represented in an array similar to binary heaps. For a node at index i, give the indices of its parent and all of its children.

c) Give an efficient algorithm for `Insert` and analyze its runtime.

d) Give an efficient algorithm for `deleteMax` and analyze its runtime.

Updating Partial Sum

Consider the problem where we have a sequence of n elements: $S = a_1, a_2, ..., a_n$, and 3 operations:

- $Add(S, b) \rightarrow a_1, a_2, ..., a_n, b$
- $Update(S, i, \Delta) \rightarrow a_1, ..., a_{i-1}, \Delta, a_{i+1}, ..., a_n$
- $PartialSum(S, k) \rightarrow \sum_{i=1}^{k} a_i$

Design a data structure that can perform each of these operations in $O(\log n)$ time.

Dominant Element

Given an array $\mathcal{A}$ and the promise that there exists an element in $\mathcal{A}$ which appears more than half the time, give an algorithm to find this dominant element in $\mathcal{O}(n)$ time with $\mathcal{O}(1)$ extra space. You should make only 1 pass through $\mathcal{A}$.

Hint: Suppose there is a counter to track a specific element such that the counter increments whenever that element is encountered, and decrements for any other element. What can we learn from the final state of the counter?

Order Notation

Prove the following asymptotic relationships from first principles:

a) $2n^7 + 15n^3 \log n + 2018 \in O(n^9)$

b) $\frac{n^2}{n+\log n} \in \Theta(n)$

c) $n! \in o(n^n)$

Professor Aniobi has recently invented a new class of functions called $Onion(f)$. A function $g(n)$ is in $Onion(f(n))$ if there exists a constant $c > 0$ such that $g(n) \leq cf(n)$ for all $n \geq 0$. We assume that $f(n)$ and $g(n)$ are functions that map positive integers to positive real numbers.

d) Give functions g and f such that $g(n) \in Onion(f(n))$.

e) Professor Aniobi says: "If $g(n) \in O(f(n))$, then $g(n) \in Onion(f(n))$ because O is the first letter of $Onion$". Prove this claim by first principles, or disprove with a counterexample.

String Sorting

a) Given n strings where the total length of all strings is m, give an algorithm to sort the strings in $O(m)$ time in lexicographical ordering, e.g., $a < ab < b$.

b) Suppose the strings now represent numbers, and the total length is still m. Give an algorithm to sort the numbers in $O(m)$ time by value, e.g. $8 < 12 < 13$.

Sorting Lower Bound

Consider the problem of sorting an array $\mathcal{A} = (a_1, a_2, \ldots)$ of elements with multiplicity $\frac{n}{k}$. That is, $\mathcal{A}$ is made up of k distinct elements $(y_1, y_2, \ldots, y_k)$, where each y_i occurs $\frac{n}{k}$ times in $\mathcal{A}$. Prove that any algorithm in the comparison model requires $\Omega(n \log k)$ comparisons to sort $\mathcal{A}$ in the worst-case. Note: $\left(\frac{n}{e}\right)^n \leq n! \leq n^n$.

Union Selection

Suppose you have two sorted arrays, A and B of sizes m and n respectively, where the elements in the union of A and B are distinct. Given an integer k with $k \leq m + n$, design an $O(\log k)$ algorithm to find the k-th smallest element in the union of A and B.

Pseudocode Runtime Analysis

Analyze the worst-case runtime for the following pseudocodes:

a)
```
function STOOGE($\mathcal{A}, i = 0, j = n - 1$)
    if $\mathcal{A}[j] < \mathcal{A}[i]$ then
        SWAP($\mathcal{A}[j], \mathcal{A}[i]$)
    end if
    if $(j - i + 1) > 2$ then
        $t = \lfloor \frac{(j-i+1)}{3} \rfloor$
        STOOGE($A, i, j - t$)
        STOOGE($A, i + t, j$)
        STOOGE($A, i, j - t$)
    end if
    return $\mathcal{A}$
end function
```

b)
```
j := 0
k := 1
while (j <= n)
    j = j + k
    k = k + 2
```

c)
```
!!!!!BONUS!!!!!
x := n
while (x > 1)
    if (x is even)
        x = x / 2
    else
        x = 3 * x + 1
!!!!!BONUS!!!!!
```