CS 3510 Quiz 1

Izaan Kamal

TOTAL POINTS

68 / 100

QUESTION 1

1 Problem 1 15 / 25

- **- 0 pts** Correct
- **- 25 pts** Missing answer/incorrect approach
- **- 15 pts** Correct recurrence, very incorrect solution: Incorrect base case or substitution procedure/incomprehensible or incorrect formulas/incomplete attempts/work shown would not get to the correct solution.
- **- 15 pts** Incorrect recurrence, correct approach towards solving by substitution
- ✓ **- 10 pts Correct recurrence, a minor mistake led to incorrect answer.**

QUESTION 2

2 Problem 2 13 / 20

- ✓ **- 0 pts (a) Correct**
- **- 10 pts** (a) Missing answer/incorrect approach
- **- 7 pts** (a) Incorrect answer, meaningful work shown
- **- 3 pts** (a) Gave pseudocode without explanation
- **- 5 pts** (a) Working algorithm, does not run in required time
- **- 2 pts** (a) Runtime not justified
- **- 4 pts** (a) Major error
- **- 2 pts** (a) Minor error
- **- 0 pts** (b) Correct
- **- 10 pts** (b) Missing answer/incorrect approach
- ✓ **- 7 pts (b) Incorrect answer, meaningful work shown**
- **- 3 pts** (b) Gave pseudocode without explanation
- **- 5 pts** (b) Working algorithm, does not run in required time
- **- 2 pts** (b) Runtime not justified
- **- 4 pts** (b) Major error
- **- 2 pts** (b) Minor error

b) You need to start from the max which you get from a). not the mid

QUESTION 3

3 Problem 3 30 / 30

- ✓ **- 0 pts Correct**
- **- 30 pts** Missing answer/incorrect approach
- **- 20 pts** Incorrect Alg
- **- 5 pts** Missing/Incorrect argument for correctness
- **- 5 pts** Missing/Incorrect argument for runtime
- **- 2 pts** Minor error
- **- 10 pts** Incorrect algorithm, meaningful work shown

QUESTION 4

4 Problem 4 10 / 25

- **- 0 pts** Correct
- **- 25 pts** Missing answer/incorrect approach
- ✓ **- 15 pts Incorrect recurrence, meaningful work shown**
- **- 5 pts** Incorrect/Missing base case or default entry value
- **- 5 pts** Incorrect/Missing argument for running time
- **- 8 pts** Does not run in required time
- **- 3 pts** Did not state what to return/Did not return correct final answer
- **- 3 pts** Minor error

Last Name: Kamal First Name: Izaan Email: ikamal3@gatech.edu

Problem 1: Analysis of Recursive Algorithm (25 points)
Consider the function Mystery defined below.

Mystery(n)
for $i := 1$ to n
  for $j := 1$ to n
    write("x")
if $n > 16$ then begin
    for i:=1 to 4
      Mystery($\frac{n}{2}$)
    end.

If we call Mystery(n), where n is a power of 2 and $n \geq 16$, how many x's (as an exact function of n, ie order of growth is not sufficient) does call Mystery(n) print? Justify your answer/show your work (ie give recurrence and solve by substitution.)

Answer: $T(n) = 4T(\frac{n}{2}) + n^2$ with $T(16) = 16^2$

$\frac{n}{2^k} = 16$

$n = 16 \cdot 2^k$

$\frac{n}{16} = 2^k$

$\log_2 \frac{n}{16} = k$

$= 4(4T(\frac{n}{2^2}) + n^2) + n^2$

$= 4^2 T(\frac{n}{2^2}) + 4n^2 + n^2$

$= 4(4^2 T(\frac{n}{2^3}) + 4n^2 + n^2) + n^2$

$= 4^3 \cdot T(\frac{n}{2^3}) + 4^2 n^2 + 4n^2 + n^2$

$= \ldots$

$= 4^k T(\frac{n}{2^k}) + 4^{k-1} n^2 + 4^{k-2} n^2 + \ldots + n^2$

$= \ldots$

$=$ Since $T(16) = 16^2$, stop when $\frac{n}{2^k} = 16$ or $k = \log_2 \frac{n}{16}$

$= 4^{\log_2 \frac{n}{16}} T(16) + 4^{\log_2 \frac{n}{16} - 1} n^2 + 4^{\log_2 \frac{n}{16} - 2} n^2 + \ldots + n^2$

$= \boxed{4^{\log_2 \frac{n}{16}} (16)^2 + n^2 \left(\frac{4^{\log_2 \frac{n}{16}} - 1}{3} \right)}$

Last Name: Kamal First Name: Izaan Email: ikamal3@gatech.edu

Problem 2: Sorting and Searching Techniques (20 Points)

You are given an array $a_1 \ldots a_n$, which is semi-sorted in the sense that it consists of distinct integers, and for some $1 < k < n$ you have $a_1 < a_2 < \ldots < a_k > a_{k+1} > a_{k+2} > \ldots > a_n$. That is, the values aof the array are strictly ascending from index 1 up to index k, and then strongly descending from index k up to index n.

(a) On input an array $a_1 \ldots a_n$ of distinct integers which is guaranteed to be semi-sorted, give (describe/outline) an $O(\log n)$ comparison algorithm to find the maximum value item of the array, and the index k which holds the value of this item. Correctness and running time should be justified.

(b) On input an array $a_1 \ldots a_n$ of distinct integers which is guaranteed to be semi-sorted, and an additional integer x, give (describe/outline) an $O(\log n)$ comparison algorithm which determines if x is an element of the array. Correctness and running time should be justified.

$5 6 4

Answer:

1 2 3 4 7 6 5 4 3 2 1 0 −1 −2

a) Find the $\frac{n}{2}$ element of the array and compare it to the previous and next element. If prev < $a_{\frac{n}{2}}$ < next, call the method again and input the array $a_{\frac{n}{2}} \ldots a_n$, as all the elements from $a_1 \ldots a_{\frac{n}{2}}$ are smaller. If prev > $a_{\frac{n}{2}}$ > next, call the method again with input $a_1 \ldots a_{\frac{n}{2}}$, as all the elements from $a_{\frac{n}{2}} \ldots a_n$ will be smaller. When previous < $a_{\frac{n}{2}}$ > next, $a_{\frac{n}{2}}$ is the max value and $\frac{n}{2} = K$. As we are halving the size of our input on each call, this is $O(\log n)$.

b) Find the element $\frac{n}{2}$, and compare it to prev and next. If prev < $a_{\frac{n}{2}}$ < next recurse down the left if $a_{\frac{n}{2}} > x$ and the right if $a_{\frac{n}{2}} < x$. Similarly, if prev > $a_{\frac{n}{2}}$ > next, recur to the left if $a_{\frac{n}{2}} < x$ and the right if $a_{\frac{n}{2}} > x$. Recur with either $a_1 \ldots a_{\frac{n}{2}}$ or $a_{\frac{n}{2}} \ldots a_n$ depending.

As we are doing an array half the size each recursion O(log n)

Last Name: Kamal First Name: Izaan Email: ikamal3@gatech.edu

Problem 3: Application of KSelect, Divide and Conquer (30 points).
Let $a(1,n) = a_1 \dots a_n$ be an set of n distinct unsorted numbers, where n is a power of 2.
Also, let k be a power of 2, with $1 < k < n$.
The k-quantiles of $a(1,n)$ are $k-1$ elements of $a(1,n)$, say $q_1 < q_2 < \dots < q_{k-1}$ that partition the elements of $a(1,n)$ in k sets $S(1,\frac{n}{k})$, $S(\frac{n}{k}+1, 2\frac{n}{k})$, $S(2\frac{n}{k}+1, 3\frac{n}{k})$, ..., $S((k-1)\frac{n}{k}+1, n)$ with the properties:

$$|S\left(1,\frac{n}{k}\right)| = |S\left(\frac{n}{k}+1, 2\frac{n}{k}\right)| = |S\left(2\frac{n}{k}+1, 3\frac{n}{k}\right)| = \dots = |S\left((k-1)\frac{n}{k}+1, n\right)| = \frac{n}{k}$$

$$S\left(1,\frac{n}{k}\right) = \{a_i \in a(1,n), \text{with } a_i \le q_1\}$$

$$S\left((j\frac{n}{k}+1, (j+1)\frac{n}{k}\right) = \{a_i \in a(1,n), \text{with } q_j < a_i \le q_{j+1}\ \} \qquad \text{for } 1 \le j \le k-2$$

$$S\left((k-1)\frac{n}{k}+1, n\right) = \{a_i \in a(1,n), \text{with } a_i > q_{k-1}\}$$

For example, if $a(1,16) = \{29, 87, 1, 23, 67, 50, 73, 26, 24, 16, 81, 63, 68, 6, 77, 40\}$ and $k=4$, the 4-quantiles of $a(1,16)$ are $23 < 40 < 68$ and partition $a(1,16)$ to sets $S(1,4) = \{1, 23, 16, 6\}$, $S(5,8) = \{29, 26, 24, 40\}$, $S(9,12) = \{67, 50, 63, 68\}$ and $S(13,16) = \{87, 73, 81, 77\}$.
On input $a(1,n) = a_1 \dots a_n$ distinct unsorted numbers, and k such that $1 < k < n$,
where both n and k are powers of 2,
give (describe/outline) an $O(n \log k)$ algorithm that outputs the k-quantiles of $a(1,n)$ in sorted order $q_1 < q_2 < \dots < q_{k-1}$. Correctness and running time should be justified.

Answer: Use a $O(n)$ K-select algorithm to find u_1 such that $n+1$ elements are less than u_1 and n elements are greater. Create two arrays, B_{small} and B_{up}, and add all element less than u_1 to B_{small} and all greater to B_{up}, and add u_1 to a list. Then do K-Select on B_{small} and B_{up} to find u_{small} and u_{up}, and add those to the ends of the list such that $u_{small} \to u_1 \to u_{up}$. Repeat until list is size $K-1$, then return list.

K-Select is $O(n)$ and we do this operation $\log k$ times, so this is $O(n \log k)$.

Last Name: KAMAL First Name: IZAAN Email: ikamal3@gatech.edu

Problem 4: Dynamic Programming (25 points)
Given an unlimited supply of coins of denominations $1 = x_1 < x_2 < \ldots < x_n$ we wish to make change for a value v, where $v > x_n$, using the minimum number of coins. Realize that, since $1 = x_1$ is among the available denominations, we can make change for any value k, where $1 \le k \le v$, using k coints of type $x_1 = 1$. But this may not be the minimum number of coins that form k. For example, for each denomination x_i, $2 \le i \le n$, we can realize the value $x_i > 1$ using a single coin of denomination x_i. Give an $O(nv)$ dynamic-programming algorithm for the following problem:
Input: $1 = x_1 < x_2 < \ldots < x_n$ and value v, $v > x_n$.
Output: The minimum number of coins of the above denominations that can form the value v.
Correctness and running time should be justified.

Answer:

15

10

```
num = v
count = 0
For i = 1 to n
  For j = 1 to num
    if (v - x_{n-i+1} >= 0)
      v = v - x_{n-i+1}
      count += 1
return count
```

This algorithm sees how many times x_n goes into v and subtracts the value from v and increments the counter, then with the new value of v checks how many coins of x_{n-1} fit in, and so on. As $x_1 = 1$, you can always make change for v using some combination.

As we have two nested loops one with length n and the other with length v, this is $O(nv)$