CS 5633 Analysis of Algorithms – Fall 18

Exam 1

NAME:

- This exam is closed-book and closed-notes, and electronic devices such as calculators or computers are not allowed. You are allowed to use a cheat sheet (half a single-sided letter paper).
- Please try to write legibly – if I cannot read it you may not get credit.
- **Do not waste time** – if you cannot solve a question immediately, skip it and return to it later.

1) Loop Invariant		20
2) Recursion Tree		17
3) Asymptotic Growth		11
4) Master Method		9
5) Randomized Analysis		18
6) Divide and Conquer		25
		100

1 Loop Invariant (20 Points)

Below we have the for loop from the Partition algorithm for Quicksort. This for loop puts every number at most the pivot to the left of every number that is greater than the pivot.

Algorithm 1 Partition(integer array A[1..n])

```
1: x = A[1]
2: i = 1
3: for all j = 2 to n do
4:    if A[j] ≤ x then
5:       i = i + 1
6:       Swap A[j] and A[i].
```

1. State a loop invariant for the for loop that is true in each iteration of the loop, and in the terminating iteration implies that the algorithm is correct. Briefly argue why the invariant indeed implies the correctness of the algorithm.

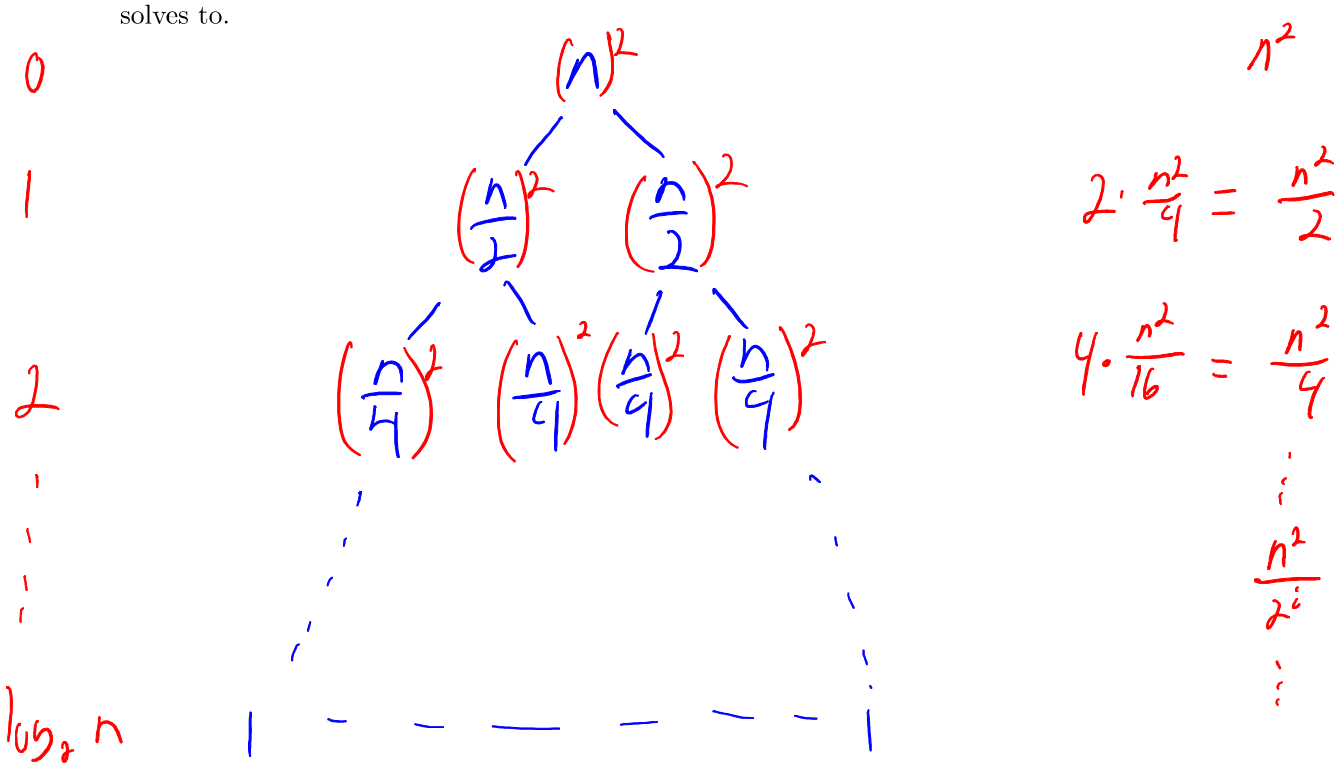
2. Prove that your loop invariant is true by induction.

1. Before the jth iteration, the numbers originally in indices $A[2 .. (j-1)]$ are arranged such that $A[2..i]$ are $\leq X$ and $A[i+1 .. (j-1)]$ are $> X$. This implies correctness because the algorithm terminates with $j = n+1$ and the invariant applies for all of $A[2..n]$.

2. Initially $j=2$ so the invariant trivially applies. Suppose it was true before iteration j. We will show it remains true before iteration $j+1$. If $A[j]$ is $> X$ then it trivially holds. If $A[j] \leq X$ then we increment i, and by inductive hypothesis $A[i] > X$ (after incrementing i). Therefore when we swap $A[i]$ and $A[j]$ the invariant will hold.

2 Recursion Tree (17 Points)

Let $T(n) = 2T(n/2) + n^2$ with $T(1) = 1$. Use a recursion tree to generate a guess of what $T(n)$ solves to.



$$\sum_{i=0}^{\log n} \frac{n^2}{2^i} = n^2 \sum_{i=0}^{\log n} \left(\frac{1}{2}\right)^i < 2n^2. \Rightarrow \boxed{T(n) = \Theta(n^2)}$$

Guess:

3 Asymptotic Growth (11 Points)

1. Use the definition of Big-oh to show that $2n^2 + 5n - 6 \in \Theta(n^2)$.

$f(n)$ $g(n)$

$2n^2 + 5n - 6 \in O(n^2)$

$2n^2 + 5n - 6 < 2n^2 + 5n < 2n^2 + 5n^2 = 7n^2.$

$f(n) \leq c \cdot g(n)$ for $c = 7$ for all $n \geq n_0 = 1.$

$g(n) \in O(f(n))$

$n^2 \leq n^2 + (n^2 + 5n - 6)$ when $n^2 + 5n - 6 \geq 0$ which is true for all $n \geq 1.$

$= 2n^2 + 5n - 6$

$g(n) \leq c \cdot f(n)$ for $c = 1$ for all $n \geq n_0 = 1.$

4 Master Method (9 Points)

Solve the following recurrences using the master theorem. Justify your answers shortly (i.e. specify ϵ and check the regularity condition if necessary).

1. $T(n) = 16T(n/4) + n^2$

$a=16, b=4, n^{\log_b a} = n^2. \quad f(n) = n^2.$

Case 2

$T(n) = \Theta(n^2 \log n).$

2. $T(n) = T(n/4) + n$

$a=1, b=4, f(n) = n^1, n^{\log_b a} = n^0, n^1 \in \Omega(n^{0+\epsilon})$ for $\epsilon = 1.$

$a \cdot f(\frac{n}{b}) = f(\frac{n}{4}) = \frac{n}{4} \leq cn$ for $c = \frac{1}{4}.$

Case 3 holds.

$T(n) = \Theta(n).$

3. $T(n) = 9T(n/3) + n \log n$

$a=9, b=3, n^{\log_b a} = n^2, f(n) = n \log n$

$n \log n \in O(n^{2-\epsilon})$ for $\epsilon = \frac{1}{2}.$

Case 1.

$T(n) = \Theta(n^2).$

5 Randomized Analysis (18 Points)

Suppose someone offers to let you play a game. They will randomly (and independently) pick three numbers x_1, x_2, x_3 in the range 1-50. If x_1 is odd, you will be paid \$4 and if it is even you will pay \$2. If x_2 is in the range 1-25 you will be paid \$3, and if it is any other value you will pay \$1. If $x_3 = 1$ then you must pay \$50 (and will be paid nothing nothing otherwise). What is the expected value of this game from your perspective?

Let X_1, X_2, X_3 denote the outcome from each number x_1, x_2, x_3 respectively.

$X_1 = 4$ if x_1 is odd and $X_1 = -2$ o.w.

$X_2 = 3$ if $x_2 \in [1, 25]$ and $X_2 = -1$ o.w.

$X_3 = -50$ if $x_3 = 1$ and $X_3 = 0$ o.w.

$$E[X_1 + X_2 + X_3] = E[X_1] + E[X_2] + E[X_3]$$

$$E[X_1] = 4 \cdot P(x_1 \text{ is odd}) + -2 \cdot P(x_1 \text{ is even})$$
$$= 4 \cdot \tfrac{1}{2} + -2 \cdot \tfrac{1}{2} = 2 - 1 = 1$$

$$E[X_2] = 3 \cdot P(X_2 \in [1, 25]) + -1 \cdot P(X_2 \in [26, 50])$$
$$= 3 \cdot \tfrac{1}{2} - 1 \cdot \tfrac{1}{2} = 1$$

$$E[X_3] = -50 \cdot P(x_3 = 1) = -50 \cdot \tfrac{1}{50} = -1$$

Expected val $= 1 + 1 - 1 = (1)$

6 Divide and Conquer (25 Points)

Let A be a *sorted* array of $n+1$ integers in the range from $[1, n]$ such that each integer is in A at least once and there is a single value that is duplicated. For example if $n = 5$, then A might be $[1, 2, 3, 3, 4, 5]$ or A might be $[1, 1, 2, 3, 4, 5]$. Give a divide and conquer algorithm that finds the value that is duplicated. Your algorithm should run as quickly as possible. What is the running time recurrance relation of your algorithm? Use the master theorem to give a Θ bound on the running time.

I'm assuming A is indexed 1 to n.

The key observation is that if $A[i] = i$ then the duplicate is somewhere "to the right". If $A[i] = i+1$, then it is "to the left". In this manner, I can check the midpoint and then recurse on whichever half has the duplicate.

```
DupFinder(A, p, q) {
    if (p ≥ q-1) return p.
    // At least 3 values in subproblem if here
    m = (p+q)/2;
    if (A[m-1] == A[m] || A[m+1] == A[m])
        return A[m];
    if (A[m] == m)
        return DupFinder(A, m+1, q);
    else
        return DupFinder(A, p, m-1);
}
```

$T(n) = T(\frac{n}{2}) + O(1)$. $a=1$, $b=2$, $n^{\log_b a} = n^0$

$f(n) = n^0$. Case 2 $\Rightarrow T(n) = \Theta(\log n)$