CSCI 6470 Midterm Exam Questions Answers

October 10, 2023 (12:45am-2:00pm EST)

Student Name ______________________________ **Student ID**____________________________

> **Rules**. Violation will result in **zero credit** for the exam and possibly the final grade.
> 1. Closed book/note/electronics/neighborhood.
> 2. Surrender your cell phone to the podium before using the restroom.

There are 8 questions and 100 points in total. Good luck!

1. (**10 points**) This question concerns big-O and big-Ω.

   (1) $T(n) = O(f(n))$ is equivalent to that $\exists c > 0, n_0 > 0$, such that $T(n) \leq cf(n)$ for all $n \geq n_0$. $\boxed{3\ points}$

   (2) $T(n) = \Omega(g(n))$ is equivalent to that $\exists c > 0, n_0 > 0$, such that $T(n) \geq cg(n)$ for all $n \geq n_0$. $\boxed{3\ points}$

   (3) Fill in the blank spaces with either O or Ω to make the equalities correct.

   $\boxed{each question: 1\, point}$, $\boxed{without\ (,\ ): 1\, point}$

   (a) $n^2 + 4n\log_2 n = \Omega(\ n\log_2 n)$ (b) $24 \times 2^n = \Omega(2^{\frac{n}{2}})$

   (c) $n^k = O(k^n)$, for constant $k > 1$ (d) $(\log_2 n)^k = O(n^{\frac{1}{k}})$, for constant $k > 1$

2. (**10 points**) Write a recursive `Randomized-QuickSort` algorithm. You do not need to write the `partition` subroutine, however you can use it. Be succinct.

```
function Randomized-QuickSort(L, low, high);   2 point
  1. if low < high   1 point
  2.     r = randomly generated index;   2 points
  3.     swap(L[r], L[high]);   1 point
  4.     k = partition(L, low, high);   2 points
  5.     Randomized-QuickSort(L, low, k-1);   1 point
  6.     Randomized-QuickSort(L, k+1, high);   1 point
```

3. (**10 points**) Run the `partition` subroutine (of `Quicksort`) on the following list to show the partition result (which should be a list): `[8 4 3 1 6 7 11 9 2 10 5]`

   Result: `[4 3 1 2 5 7 11 9 8 10 6]` $\boxed{one\ mistake: 1\, point}$

4. (**20 points**) Consider the following recursive algorithm `Work` that takes as inputs number p and integer n. You may assume that n is always a power of 2.

```
function Work(p, n);
   1. if (n=1)
   2.     return p;
   3. else
   4.     M = Work(p, n/2);
   5.     M = M x M;          // x represents multiplication
   6.     return M;
```

(1) What does `Work` do? computing p^n $\boxed{5\ points}$

(2) Assume p is small, formulate worst case time function $T(n)$ for `Work`:

$$T(n) = \begin{cases} a & n = 1 \text{ (base case)} \quad \boxed{2\ points} \\ T(n/2) + b & n \geq 2 \quad \boxed{5\ points} \end{cases}$$

(3) According to what you give in (2), $T(n) = O(\log_2 n)$; $\boxed{2\ points}$

(4) Assume p can be as large as of n bits in length. The recursive case for $T(n)$ becomes

$T(n) = T(n/2) + bn^2$ $\boxed{4\ points}$

(5) According to what you give in (4), $T(n) = O(n^2)$; $\boxed{2\ points}$

5. (**15 points**) Assume that in design of `Selection` algorithm that finds the k^{th} smallest element from a given set, groups of 7, instead of 5, elements are constructed. Give recursive formula for the time complexity $T(n)$ of the algorithm, where c is a constant (**don't worry about it**).

$$T(n) = \begin{cases} a & n \leq c \quad \boxed{3\ points} \\ T(n/7) + T(5n/7) + bn & n > c \quad \boxed{5 + 5 + 2 = 12\ points} \end{cases}$$

6. (**10 points**) In the lower bound proof for **Sorting** problem, we have resorted to the comparison-based (i.e., decision tree) model. Explain each of the following terms in the context of this proof: $\boxed{each\ question : 2\ points}$

(1) A *decision tree* represents an algorithm;

(2) A *leaf* on a decision tree represents the output for one specific scenario of the input;

(3) The *depth* of a decision tree represents worst case running time of the algorithm;

(4) A *longest path length* on a decision tree represents worst case running time of the algorithm;

(5) The *number of leaves* on a decision tree represents number of computation paths/input scenarios that the algorithm needs to deal with separately;

7. (**10 points**) Consider the following typical DFS algorithm.

```
function dfs-main(G);
  1. for every vertex x in G
  2.     visited(x) = false;
  3. for every vertex x in G
  4.     if visited(x) equals false
  5.        explore(G, x)
```

```
function explore(G, x);
  1. visited(x) = true;
  2. for every edge (x, y) in G
  3.     if visited(y) equals false
  4.        explore(G, y);
```

The DFS can be used/modified to solve the following problem:

Input: a directed graph $G = (V, E)$, vertices $s, t \in V$, and a forbidden set of vertices $F \subset V$
Output: answer "yes" if and only if there is a path $s \rightsquigarrow t$ that avoids completely the set F;

Describe

(1) How to use this algorithm to determine if $s \rightsquigarrow t$: call `explore(s)`; $\boxed{\textit{3 points}}$
in line 2 of function `explore(G, x)`, check `y = t` $\boxed{\textit{4 points}}$

(2) How to slightly modify the algorithm to avoid vertices in F: between lines 3 in function `explore(G, x)`, modify `if visited(y) equals false` to
`if visited(y) equals false` AND `fobidden(y) equals false` $\boxed{\textit{3 points}}$

You may use the pre-set variable `forbidden`$(v) =$ `true` to determine if any vertex $v \in F$.

8. (**15 points**) `MergeSort` algorithm has following recursive formula for its time complexity $T(n)$,

$$T(n) = \begin{cases} a & n = 1 \\ 2T(\frac{n}{2}) + bn & n \geq 2 \end{cases}$$

We claim that $T(n) = \Omega(n \log_2 n)$.

- To prove $T(n) = \Omega(n \log_2 n)$, it is equivalent to prove the statement that
  $\exists c > 0, n_0 > 0,\ T(n) \geq cn \log_2 n$ for all $n \geq n_0$ $\boxed{\textit{4 points}}$
- To prove the statement by induction on n,

  (1) We first prove the statement holds for base case(s). That is,
  $n = 1$, indeed because $a > 0$, $T(1) = a$, is great than $c \times 1 \times \log_2 1 = 0$. So $c > 0$ exists;
  $\boxed{\textit{2 points}}$

  (2) Then we assume the statement holds for $n = \frac{k}{2}$. That is, $T(\frac{k}{2}) \geq c\frac{k}{2} \log_2 \frac{k}{2}$; $\boxed{\textit{3 points}}$

(3) Finally, by induction, we show when $n = k$

$$\begin{aligned}
T(k) &= 2T(\frac{k}{2}) + bk \ \text{(by the given recursive formula)} \ \boxed{2 \ \textit{points}} \\
&\geq 2 \times c\frac{k}{2} \log_2 \frac{k}{2} + bk \ \text{(by the assumption on } n = \tfrac{k}{2}) \ \boxed{2 \ \textit{points}} \\
&= ck(\log_2 k - \log_2 2) + bk \\
&= ck \log_2 k - ck + bk \\
&\geq ck \log_2 k \ \text{(by choosing } c \leq b) \ \boxed{2 \ \textit{points}}
\end{aligned}$$

So we have proved that there exists $c = b$, and $n_0 = 1$ such that

$$T(n) \geq cn \log_2 n$$

for all $n \geq n_0$. That is, $T(n) = \Omega(n \log_2 n)$.