University of Toronto
Department of Electrical and Computer Engineering

# Midterm Examination

ECE 345 Algorithms and Data Structures
Fall 2007

Print your name and ID number neatly in the space provided below; print your name at the upper right corner of every page.
The exam is eight (8) pages including the cover page; if not, report it to the instructor or TA.

| | |
|---|---|
| Name: | |
| ID Number: | |

- This is an *open book* exam. You are permitted to use the textbook for the course but nothing else is permissible. You may reference/use textbook page/section/algorithms if you wish so. Non-native English speakers may use a dictionary.
- Do all five problems in this booklet. Try not to spend too much time on one problem. Use terminology from the textbook. You must define any different terms before you use them.
- Write clearly and only in the space provided. Ask the proctor if you need more paper. **Nothing on the back of the sheets will be graded**.
- You have 120 minutes for this exam. Raise your hand if you have a question. Turn off your mobile phone.
- Do not give C code! Write pseudocode and analyze time/memory requirements of your algorithms when asked to receive full credit. All logarithms are base 2 unless otherwise noted.

| Question | Points | Score | Grader |
|---|---|---|---|
| 1 | 20 | | |
| 2 | 20 | | |
| 3 | 20 | | |
| 4 | 20 | | |
| 5 | 20 | | |
| Total | 100 | | |

1. **Recurrences.**

   (a) Solve the recurrence $A(n) = 3A(n/5) + log^3 n$ using the Master Theorem. Show clearly your intermediate steps.

   (b) Solve the recurrence $B(n) = B(n/5) + B(4n/5) + \Theta(n)$ by drawing a recursion tree. Show clearly your intermediate steps.

2. **Sorting.**

The recent discovery of the following fragment of uncommented procedural C code in the Sunlab has caused a big scandal.

```
foo(int a[],int l,int r,int k)
{ int i;
i = partition(a,l,r);
if (i == k) return a[k];
else if (i < k) return foo(a,i+1,r,k);
else return foo(a,l,i-1,k); }
```

Since the author of the code is still hiding, we will have to decipher it. The TAs determined that function `partition(int a[],int l,int r)` works as follows: let `v=a[r]`; `partition` rearranges the subarray `a[l]` through `a[r]` such that the elements equal to `v` precede those greater than `v` and follow those smaller than `v`, and returns an index `i` such that `a[i]=v`. Thus, `partition` works in the same manner as the Partition of Quicksort discussed in class, returning the position of the partitioning element `v` in the rearranged subarray `a[l...r]`. Let `a` be an array of size $N$ of integers, and $1 \leq k \leq N$.

(a) What will `foo(a,1,N,k)` return? Explain.

(b) What is the worst-case time complexity of `foo(a,1,N,k)`, and for which inputs does it occur?

3. **Combinatorics.**

   Use induction to prove that any positive integer can be written as the sum of distinct powers of 2. For example, $42 = 2^5 + 2^3 + 2^1, 25 = 2^4 + 2^3 + 2^0$. State clearly your induction base, hypothesis and the inductive step. Writing the number in binary is not a proof, it just restates the problem!

4. **Heaps and Data Structures.**

   You are given a heap of $n$ numbers stored so that the value at every node is larger than all the values in its subtrees. Design an $O(k \lg k)$-time algorithm to find the $k$-th largest value in the heap. Partial credit will be given for sub-optimal solutions. *Hint:* Generate a separate heap of size $O(k)$.

5. **Dynamic Programming.** Suppose you are given a complete binary tree $T$ with a root $r$ and $n$ nodes. Assume that on each node $v \in T$ there is a prize $p_v \geq 0$. You are also given a number $k \geq 0$, the number of prizes you are allowed to pick up. Your goal is to select a set of $k$ nodes that form a tree rooted at $r$ that contains the largest total prize. The set of edges of the new tree is a subset of those in the original tree.

   Here is a recurrence relation for this problem: in the description, we assume that $v$ is a node, and unless it is a leaf, its children are $v_1$ and $v_2$.

$$\begin{array}{lcll} OPT(0,v) & = & 0, & \text{for all nodes } v \\ OPT(i+1,v) & = & p_v, & \text{for all leaves } v \text{ and all } i \\ OPT(i+1,v) & = & p_v + \max\limits_{0\leq j\leq i}\{OPT(j,v_1)+OPT(i-j,v_2)\}, & \text{for all internal nodes } v \text{ and all } i. \end{array}$$

   Each of the following questions is 5 points. The question continues in the following page.

   (a) Explain precisely what is the meaning of $OPT(i,v)$.

   (b) Prove that the recurrence above correctly captures the meaning you gave so that you can later use it to solve the dynamic programming problem.

(c) How would you compute the value of the best overall prize once you have computed $OPT(i, v)$ for all $i$ and $v$.

(d) What would be the running time of a bottom-up implementation of this recurrence? Explain.

"My client is advised not to answer that question."

(DO NOT REMOVE – This page left intentionally blank)