CS 170 Algorithms
Spring 2009 David Wagner

MT1

PRINT your name: _____, _____
(last) (first)

SIGN your name: _____

PRINT your Unix account login: _____

Your TA's name: _____

Name of the person sitting to your left: _____

Name of the person sitting to your right: _____

You may consult any books, notes, or other paper-based inanimate objects available to you. Calculators and computers are not permitted. Please write your answers in the spaces provided in the test. We will not grade anything on the back of an exam page unless we are clearly told on the front of the page to look there. Unless specified otherwise, you do not need to justify your answers on this exam, and we will not grade any justification you may provide. When asked for answers using $O(\cdot)$ notation, provide the most specific answer possible (for example: do not write $O(n^3)$ if $O(n^2)$ is also correct).

You have 120 minutes. There are 6 questions, of varying credit (100 points total). The questions are of varying difficulty, so avoid spending too long on any one question.

Do not turn this page until your instructor tells you to do so.

Problem 1	
Problem 2	
Problem 3	
Problem 4	

Problem 5	
Problem 6	
Total	

Problem 1. [True or false] (16 points)

Circle TRUE or FALSE. Do not justify your answers on this problem.

(a) TRUE or FALSE: If $f(n) = (n+1)n/2$, then $f(n) \in O(n^2)$.

(b) TRUE or FALSE: If $f(n) = (n+1)n/2$, then $f(n) \in \Theta(n^2)$.

(c) TRUE or FALSE: If $f(n) = (n+1)n/2$, then $f(n) \in \Theta(n^3)$.

(d) TRUE or FALSE: $n^{1.1} \in O(n(\lg n)^2)$.

(e) TRUE or FALSE: It's possible to multiply two n-bit integers in $O(n^{1.9})$ time.

(f) TRUE or FALSE: If vertices u, v are in the same strongly connected component of a directed graph G, then it is necessarily the case that v is reachable from u in G.

(g) TRUE or FALSE: If vertices u, v are *not* in the same strongly connected component of a directed graph G, then it is necessarily the case that v is *not* reachable from u in G.

(h) TRUE or FALSE: If we run breadth-first search on a directed graph G starting from vertex s, then depending on the graph, it might visit some vertices that a depth-first search starting from s would not visit. (Assume that we use exactly the breadth-first search algorithm specified in the book.)

Problem 2. [Recurrences] (16 points)

Write the solution to the following recurrences. Express your answer using $O(\cdot)$ notation. Do not justify your answers on this problem. Do not show your work.

(a) Solve the recurrence $F(n) = F(\lceil n/2 \rceil) + O(1)$.

(b) Solve the recurrence $F(n) = 4F(\lceil n/2 \rceil) + O(1)$.

(c) Solve the recurrence $F(n) = 4F(\lceil n/2 \rceil) + O(n)$.

(d) Solve the recurrence $F(n) = 4F(\lceil n/2 \rceil) + O(n^2)$.

Problem 3. [Three-way mergesort] (18 points)

Alice suggests the following variant on mergesort: instead of splitting the list into two halves, we split it into three thirds. Then we recursively sort each third and merge them.

Mergesort3($A[0..n-1]$):
1. If $n \leq 1$, then return $A[0..n-1]$.
2. Let $k := \lceil n/3 \rceil$ and $m := \lceil 2n/3 \rceil$.
3. Return Merge3(Mergesort3($A[0..k-1]$), Mergesort3($A[k..m-1]$), Mergesort3($A[m..n-1]$)).

Merge3(L_0, L_1, L_2):
1. Return Merge(L_0, Merge(L_1, L_2)).

Assume that you have a subroutine Merge that merges two sorted lists of lengths ℓ, ℓ' in time $O(\ell + \ell')$. You may assume that n is a power of three, if you wish. Do not justify your answers on this problem. Do not show your work.

(a) What is the asymptotic running time for executing Merge3(L_0, L_1, L_2), if L_0, L_1, and L_2 are three sorted lists each of length $n/3$? Express your answer using $O(\cdot)$ notation.

(b) Let $T(n)$ denote the running time of Mergesort3 on an array of size n. Write a recurrence relation for $T(n)$.

(c) Solve the recurrence relation in part (b). Express your answer using $O(\cdot)$ notation.

(d) Is the Mergesort3 algorithm asymptotically faster than insertion sort? Circle YES or NO.

(e) Is the Mergesort3 algorithm asymptotically faster than the ordinary mergesort? Circle YES or NO.

Problem 4. [Algorithm design] (12 points)

Suppose we have t sorted arrays, each with n elements, and we want to merge them to get a single sorted array with tn elements. Your task is to fill in the blanks below to get an algorithm, ManyMerge, that solves this problem and has a running time of $O(nt \lg t)$.

You may assume that you are given an algorithm, Merge(L, L'), that merges two sorted lists of size ℓ, ℓ' into a single sorted list of size $\ell + \ell'$. You may assume that it runs in $O(\ell + \ell')$ time.

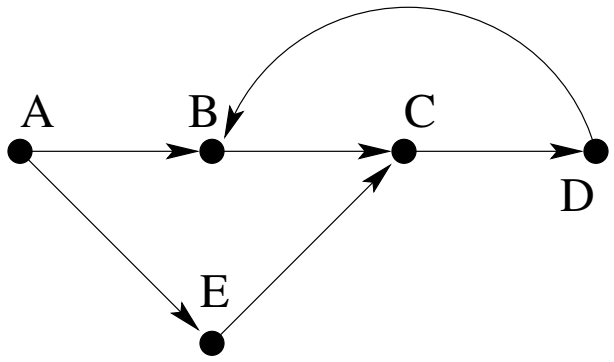
Fill in the empty boxes below, to get a correct algorithm whose running time is $O(nt \lg t)$.

ManyMerge$(L_1[0..n-1], L_2[0..n-1], \ldots, L_t[0..n-1])$:

1. If $t = 1$, then return [ ].
2. If $t = 2$, then return Merge(L_1, L_2).
3. Set $L :=$ ManyMerge([ ]).
4. Set $L' :=$ ManyMerge([ ]).
5. Return [ ].

Problem 5. [Depth-first search] (16 points)

Run depth-first search on the directed graph below, starting at vertex A. Whenever there is a choice of the order to explore vertices, use alphabetical order (so A is chosen before B, and B before C, etc.).



(a) Draw the DFS tree that results, in the space provided below. Use solid lines for tree edges, and dotted lines for non-tree edges.

(b) Label each non-tree edge in the graph above with "forward", "back", or "cross", according to whether the edge is a forward edge, back edge, or cross edge.

(c) Can the above graph be topologically sorted? Circle YES or NO. Do not justify your answer.

(d) How many strongly connected components does this graph have? Do not justify your answer.

Problem 6. [Short answer] (22 points)

Answer each question below *concisely* (one short sentence or a number should suffice). Do not justify your answer. Do not show your work.

(a) Suppose we are given a directed graph $G = (V, E)$ represented in adjacency list format, and we want to test whether G is a dag or not, using a method that is as asymptotically efficient as possible. In a sentence, what approach would you use?

(b) What's the running time of your solution in (a), using $O(\cdot)$ notation?

(c) Let $G = (V, E)$ be a directed graph with $|V| = 1000$ vertices, $|E| = 5000$ edges, and 700 strongly connected components. How many vertices does the metagraph have?

(d) Let $G = (V, E)$ be a dag, where each edge is annotated with some positive length. Let s be a source vertex in G. Suppose we run Dijkstra's algorithm to compute the distance from s to each vertex $v \in V$, and then order the vertices in increasing order of their distance from s. Are we guaranteed that this is a valid topological sort of G? Circle YES or NO.

(e) Justify your answer to part (d) as follows: If you circled YES, then give one sentence that explains the main idea in a proof of this fact. If you circled NO, then give a small counterexample (a graph with at most 4 vertices) that disproves it.

(f) Suppose we run Dijkstra's algorithm on a graph with n vertices and $O(n \lg n)$ edges. Assume the graph is represented in adjacency list representation. What's the asymptotic running time of Dijkstra's algorithm, in this case, if we use a binary heap for our priority queue? Express your answer as a function of n, and use $O(\cdot)$ notation.