CS 170 Fall 2014 — Algorithms, David Wagner — MT1

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

SIGN your name: ____

Your Student ID number: ____

Your Unix account login: cs170-____________

The room you are sitting in right now: ______________________________________

Name of the person sitting to your left: ____

Name of the person sitting to your right: ____

You may consult one double-sided sheet of notes. You may not consult other notes, textbooks, etc. Computers and other electronic devices are not permitted.

There are 12 questions. The last page is page 15.

Answer all questions. Be precise and concise. Write in the space provided. Good luck!

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

Problem 1. [True or false] (9 points)

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

(a) TRUE or FALSE: A connected undirected graph is guaranteed to have at least $|V|-1$ edges.

(b) TRUE or FALSE: A strongly connected directed graph is guaranteed to have at least $|V|-1$ edges.

(c) TRUE or FALSE: In a DAG, the number of distinct paths between two vertices is at most $|V|^2$.

(d) TRUE or FALSE: Every DAG has at least one source.

(e) TRUE or FALSE: Depth-first search on a connected undirected graph G will visit all of the vertices of G.

(f) TRUE or FALSE: After running depth-first search on a directed graph, the node with the smallest post number is part of a source component.

(g) TRUE or FALSE: Suppose we have a graph where each edge weight value appears at most twice. Then, there are at most two minimum spanning trees in this graph.

(h) TRUE or FALSE: If $f(n) = O(n^2)$ and $g(n) = O(n^2)$, then $f(n) = O(g(n))$.

(i) TRUE or FALSE: If $f(n) = O(g(n))$ and $g(n) = O(n^2)$, then $f(n) = O(n^2)$.

(j) TRUE or FALSE: Suppose we run DFS on an undirected graph and find exactly 17 back edges. Then the graph is guaranteed to have at least one cycle.

(k) TRUE or FALSE: DFS on a directed graph with n vertices and at least n edges is guaranteed to find at least one back edge.

(l) TRUE or FALSE: DFS on an undirected graph with n vertices and at least n edges is guaranteed to find at least one back edge.

(m) TRUE or FALSE: Suppose we run DFS on an undirected graph, and we discover a vertex v with $\text{pre}(v) = 1$ and $\text{post}(v) = 2|V|$. Then the graph must be connected.

(n) TRUE or FALSE: Suppose we run DFS on a directed graph, and we discover a vertex v with $\text{pre}(v) = 1$ and $\text{post}(v) = 2|V|$. Then the graph must be strongly connected.

(o) TRUE or FALSE: If the expected running time of an algorithm is $O(n)$, then its worst-case running time is also $O(n)$.

(p) TRUE or FALSE: There is an algorithm to multiply two n-bit numbers in $O(n^{\log_2 3})$ time.

(q) TRUE or FALSE: There is an algorithm to square a n-bit number in $O(n^{\log_2 3})$ time.

(r) TRUE or FALSE: If we had an algorithm to square a n-bit number in $O(n)$ time, we could multiply two n-bit numbers in $O(n)$ time.

Problem 2. [Solving recurrences] (8 points)

You don't need to justify your answer or show your work on this problem. Express your answer using $\Theta(\cdot)$ notation.

(a) What's the solution to the recurrence $T(n) = 2T(n/2) + n$?

(b) What's the solution to the recurrence $U(n) = 2U(n/2) + n \lg n$?

(c) What's the solution to the recurrence $F(n) = F(n/2) + \Theta(n)$?

(d) What's the solution to the recurrence $G(n) = 0.5G(n-2) + \Theta(1)$?

Problem 3. [Fast Fourier Transform] (4 points)

Suppose we would like to evaluate a polynomial $p(x)$ on n distinct values $x_1, x_2, \ldots, x_n$, where n is a power of two. What values of $x_1, \ldots, x_n$ should we choose, so that we can use the FFT for this purpose? Does it matter which values we pick?

Problem 4. [Short answer] (6 points)

Answer each question with "Yes" or "No". If you answer Yes, give a brief justification (one sentence). If you answer No, draw a small counterexample.

(a) Suppose G is a connected, undirected graph whose edges all have positive weight. Let M be a minimum spanning tree of this graph. Now, we modify the graph by adding 7 to the weight of each edge. Is M guaranteed to be a minimum spanning tree of the modified graph?

(b) Suppose G is an undirected graph whose edges all have positive length. Let P be a shortest path from u to v. Now, we modify the graph by adding 7 to the weight of each edge. Is P guaranteed to be a shortest path from u to v in the modified graph?

Problem 5. [Running time analysis] (12 points)

You don't need to justify your answers or show your work on this problem.

Given two 64-bit integers a,n, here is an algorithm to compute a^n:

```
Power($a,n$):
1.  If $n = 0$: return 1.
2.  Return $a \times$ Power($a, n-1$).
```

Assume throughout this problem that we don't need to worry about overflow (a^n fits into a 64-bit integer variable) and that each operation on a 64-bit integer takes $O(1)$ time.

(a) Let $T(n)$ denote the running time of Power(a,n). Write a recurrence relation for $T(n)$.

(b) What is the solution to your recurrence from part (a)? Use $\Theta(\cdot)$ notation.

You are now given another algorithm for the same problem:

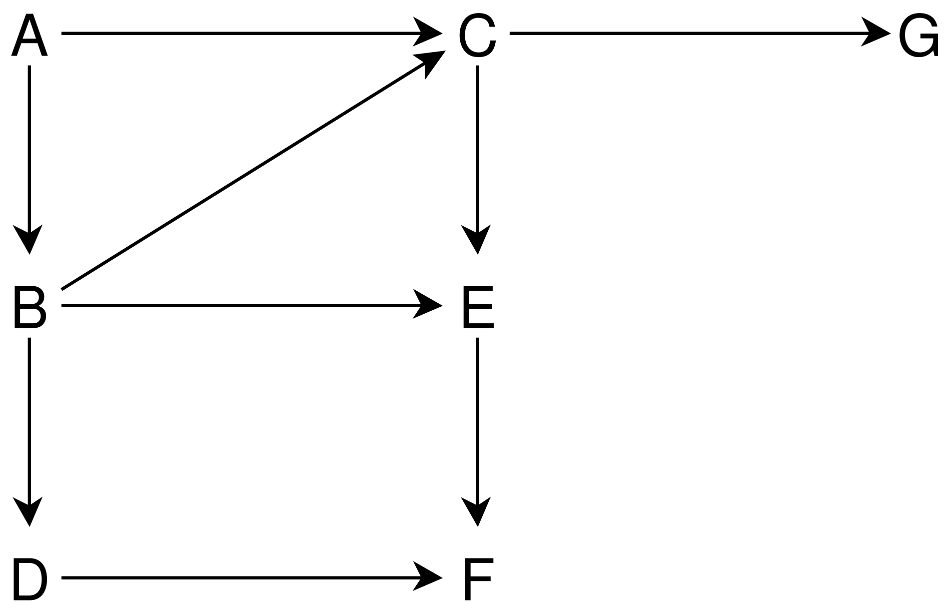
```
AltPower($a,n$):
1.  If $n = 0$: return 1.
2.  If $n = 1$: return $a$.
3.  If $n$ is even:
4.        Return AltPower($a \times a, n/2$).
5.  else:
6.        Return $a \times$ AltPower($a \times a, (n-1)/2$).
```

(c) Let $A(n)$ denote the running time of AltPower(a,n). Write a recurrence relation for $A(n)$.

(d) What is the solution to your recurrence from part (c)? Use $\Theta(\cdot)$ notation.

(e) Which would you expect to be faster, AltPower or Power?

Problem 6. [DFS] (8 points)



Perform a depth-first search on the graph above, starting from vertex A. Whenever there's a choice of vertices, pick the one that is alphabetically first.

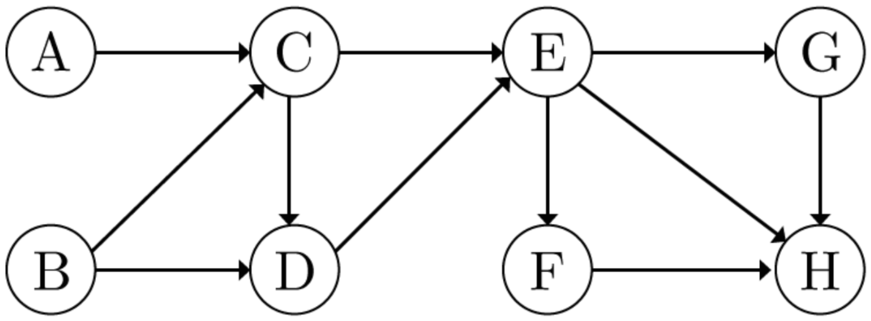
(a) Fill in the table below with the pre and post number of each vertex.

	A	B	C	D	E	F	G
pre							
post							

(b) Next, label each edge in the graph above as a tree, back, forward, or cross edge.

Problem 7. [Topological sorting] (6 points)

Consider the following graph:



(a) Which of the following orderings is not a valid topological sort of the graph?

(i) ABCDEFGH

(ii) ABCDEGFH

(iii) BACDEFGH

(iv) BACEDFGH

(b) If we run DFS on this graph, which of the following statements must be true? Circle all that must be true.

(i) post(B) > post(D)

(ii) post(G) > post(F)

(iii) post(A) > post(B)

(iv) post(D) > post(F)

Problem 8. [Shortest paths] (8 points)

Let $G = (V, E)$ be a directed graph, with non-negative edge lengths; $\ell(e)$ denotes the length of edge e.

(a) Suppose we run Dijkstra's algorithm starting from s. After it finishes, is it guaranteed that $\text{dist}(t)$ will hold the length of the shortest path from s to t? Write "yes" or "no". Do not justify your answer.

(b) If we run Dijkstra's algorithm starting from s, then after it finishes we can use the $\text{prev}(\cdot)$ pointers to find a shortest path from s to t. Concisely explain how.

(c) Is it guaranteed that the path you constructed in part (b) will have the fewest number of edges, out of all paths from s to t of length $\text{dist}(t)$? If yes, write "yes" and explain why briefly. If no, write "no" and show a small counterexample.

(d) Now suppose that exactly *one* edge of G has a negative length, and all other edges have non-negative length, and there is no negative cycle in the graph. Suppose we run Dijkstra's algorithm starting from s. After it finishes, is it guaranteed that $\text{dist}(t)$ will hold the length of the shortest path from s to t? If yes, write "yes" and explain why. If no, write "no" and show a small counterexample.

Problem 9. [All paths go through...] (8 points)

Google Maps wants to add a new feature: include a stop to Disneyworld in your route. In particular, they are looking for an algorithm for the following problem:

Input: a directed graph $G = (V,E)$, with a positive length $\ell(e)$ on each edge e; vertices s,w,t

Output: the length of the shortest path from s to t that goes through w.

They propose the following algorithm:

1. Call Dijkstra(G,ℓ,w), to get $d(w,v)$ for each $v \in V$.
2. Reverse the direction of all the edges; call the result G^r.
3. Call Dijkstra(G^r,ℓ,w), to get $d(v,w)$ for each $v \in V$.
4. Return $d(s,w) + d(w,t)$.

(a) Is their algorithm correct? If yes, write "yes" and explain why in a sentence or two. If no, write "no" and show a small counterexample.

(b) What is the asymptotic running time of their algorithm? Use $\Theta(\cdot)$ notation.

(c) Suppose G is a dag. Is G^r guaranteed to be a dag? Yes or no. Don't justify your answer.

(d) Suppose G is a dag. If we replace both calls to Dijkstra's algorithm with calls to the algorithm for computing shortest paths in a dag, will the modified algorithm be correct? Yes or no. Don't justify your answer.

(e) What is the asymptotic running time of the modified algorithm from part (d)? Use $\Theta(\cdot)$ notation.

Problem 10. [Graph subsets] (7 points)

Let $G = (V, E)$ be a connected, undirected graph, with edge weights $w(e)$ on each edge e. Some edge weights *might be negative*. We want to find a subset of edges $E' \subseteq E$ such that $G' = (V, E')$ is connected, and the sum of the weights of the edges in E' is as small as possible subject to the requirement that $G' = (V, E')$ be connected.

(a) Is it guaranteed that the optimal solution E' to this problem will always form a tree? Write yes or no. Don't justify your answer.

(b) Does Kruskal's algorithm solve this problem? If yes, explain why in a sentence or two; if no, give a small counterexample.

(c) Describe an efficient algorithm for this problem. Be concise. You should be able to describe your algorithm in one or two sentences. (You don't need to prove your algorithm correct, justify it, show pseudocode, or analyze its running time.)

Problem 11. [Graphs and Reductions] (6 points)

If S is a set of vertices in an undirected graph $G = (V,E)$, define $f(S)$ to be the length of the shortest edge between a vertex in S and a vertex not in S, i.e.,

$$f(S) = \min\{\ell(v,w) : v \in S, w \notin S, \{v,w\} \in E\}.$$

We'd like an algorithm for the following problem, with running time $O((|V|+|E|)\log|V|)$ or less:

Input: a connected, undirected graph $G = (V,E)$, with a non-negative length $\ell(e)$ on each edge e.

Output: a non-empty set S that makes $f(S)$ as large as possible, subject to the requirement that $S \neq V$.

We can solve this problem by making a small change to one of the graph algorithms we've seen in this class. Which algorithm?

What's the small change? Answer concisely (one sentence).

Problem 12. [Algorithm design] (11 points)

We are given an array $A[0..n-1]$, where $n > 1$ and all array elements are non-negative integers. Our goal is to find the maximum value of $A[i] + A[j]^2$, where the indices i, j range over all values such that $0 \le i < j < n$.

Fill in the blanks below to produce an efficient algorithm that correctly solves this problem.

```
FindMax($A[0..n-1]$):
1. If $n \le 1$, return $-\infty$.
2. Let $k := \lfloor n/2 \rfloor$.

3. Set $x :=$ FindMax(________________________).

4. Set $y :=$ ____________________________________________________________.

5. Set $z := \max(A[0], \ldots, A[k-1]) +$ ________________________________________________.
6. Return $\max(x, y, z)$.
```

(a) Write a recurrence relation for the running time of your algorithm.

(b) What is the asymptotic running time of your algorithm? Use $\Theta(\cdot)$ notation. You don't need to justify your answer.

(You do not need to prove your algorithm correct.)

(continued on the next page)

(c) Describe a $O(n)$ time algorithm for this problem. (No proof of correctness or justification of running time needed.)

Main idea:

Pseudocode: