Instructions: To solve these problems, you are allowed to consult your classmates, as well as the class textbook (Algorithms by Dasgupta, Papadimitriou, and Vazirani, which we will call DPV) and notes, but no other sources. We encourage you to collaborate with other students, while respecting the collaboration policy. Please write the names of all the other students you collaborated with on the homework. Everyone must write up their assignments separately. Staple your homework, and write your section number on it.

Please write clearly and concisely, and use rigorous, formal arguments. If you are asked to provide an algorithm the best thing to do is to provide a clear description in English (for example: “Use BFS, but with the following small change...”, or “Run BFS two times as follows...”), but you may also write some short pseudocode similar to the pseudocode in the DPV text. Homework is due at the beginning of lecture, and homework turned in later will be considered late and will use up one of your late days. Emailed copies will not be accepted.

(1) DPV Problem 0.1 (a-g) and (m-o). Just give the answers, no proofs are necessary.

(2) Consider the following pseudocode of a function which takes an integer \( n \geq 0 \) as input.

\[\text{Function } \text{foo}(n)\]
\[
\quad \text{if } n == 0 \text{ then} \\
\quad \quad \text{Return;}
\]
\[
\quad \text{for } i = 0 \text{ to } n - 1 \text{ do} \\
\quad \quad \text{Print } *; \\
\quad \text{end}
\]
\[
\quad \text{foo}(n - 1);
\]

Let \( T(n) \) be the number of times the above function prints a star (*) when called with input \( n \geq 0 \). What is \( T(n) \) exactly, in terms of only \( n \) (and not values like \( T(n - 1) \) or \( T(n - 2) \))? Prove your statement.

(3) Same as Problem 2 above, but for the following:

\[\text{Function } \text{bar}(n)\]
\[
\quad \text{Print } *; \\
\quad \text{if } n == 0 \text{ then} \\
\quad \quad \text{Return;}
\]
\[
\quad \text{for } i = 0 \text{ to } n - 1 \text{ do} \\
\quad \quad \text{bar}(i); \\
\quad \text{end}
\]

(4) You are given two connected undirected graphs, \( G_1 = (V_1, E_1) \) and \( G_2 = (V_2, E_2) \), with \( V_1 \cap V_2 = \emptyset \), as well as a node \( s \in V_1 \) and a node \( t \in V_2 \). You are also given a set of edges \( E' \) which you are able to build: each edge in \( E' \) has one endpoint in \( V_1 \) and one endpoint in \( V_2 \). Thus, building any edge of \( E' \) would connect the two graphs together, and in particular would form paths connecting \( s \) and \( t \). You are asked to determine an edge \( e \in E' \) whose addition to the graphs \( G_1 \) and \( G_2 \) would result in the shortest possible (i.e., smallest
number of edges) distance between $s$ and $t$ among all the edges in $E'$. Give a linear-time algorithm to solve this problem.

Just give the algorithm, no proofs are necessary. Recall that “linear-time” means linear in the size of the input, which for this case is the number of nodes plus the number of edges in the graphs $G_1$ and $G_2$, plus the number of edges in $E'$. *Hint:* you can use BFS as a subroutine to solve this.

(5) DPV Problem 3.7(a). Give the algorithm and argue why it is correct. You can assume the graph is connected. *Hint:* you can use BFS to solve this.

(6) DPV Problem 3.11. Your algorithm should also return the length (number of edges) of the shortest cycle containing $e$, if one exists. Just give the algorithm, no proofs are necessary. *Hint:* you can use BFS to solve this.