Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Departement of Computer Science
Markus Püschel, David Steurer

8. October 2018

Algorithms & Data Structures — Homework 4 — HS 18

Exercise Class (Room & TA): ______________________

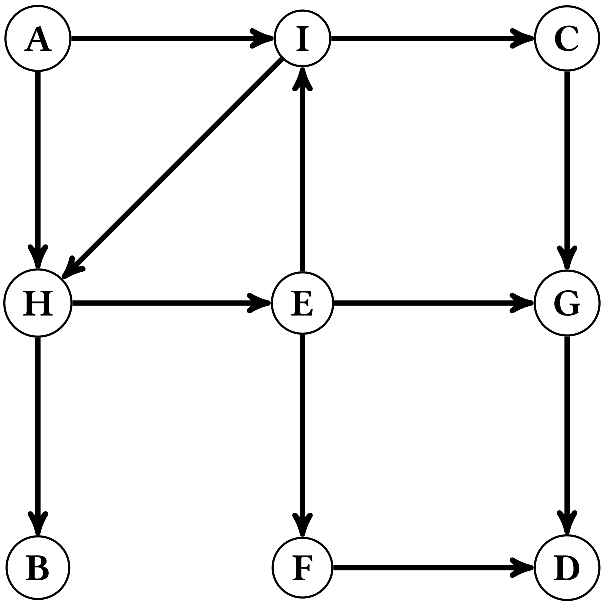
Submitted by: ______________________

Peer Feedback by: ______________________

Points: _____

Exercise 4.1 *Depth-First Search.*

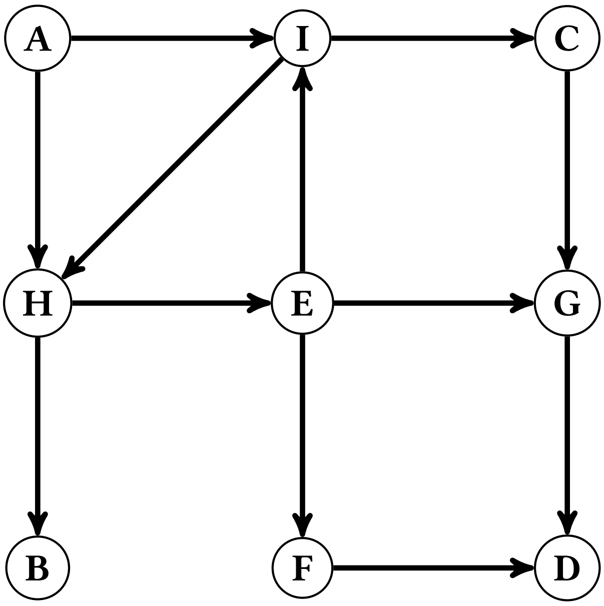
Execute a depth-first search (Tiefensuche) on the following graph starting from vertex A (using a stack, as seen in lecture). Assume that we push successor vertices (Nachfolger) on the stack in *reverse* alphabetical order. (For example, if the successors are R and U, then we first U push on the stack and then R.)



1. Mark the edges that belong to the depth-first tree (Tiefensuchbaum) with a “T” (for tree edge).
2. For each vertex, give its *pre*- and *post*-number.
3. Mark every forward edge (Vorwärtskante) not in the depth-first tree with an “F”, every backward edge (Rückwärtskante) with an “B”, and every cross edge (Querkante) with a “C”.
4. Has the above graph a topological ordering? How can we use the above execution of depth-first search in order to see this?

Exercise 4.2 *Breadth-First Search* ***(2 Points).***

On the following graph, execute a breadth-first search (Breitensuche) starting from vertex A (using a queue, as seen in the lecture). Assume that successor vertices (Nachfolger) are enqueued in alphabetical order.



1. Write down the order in which the vertices are dequeued during this execution of breadth-first search.

2. As seen in the lecture, breadth-first search can be used to determine the distances for all vertices from the start vertex. These distances partition the graph into level sets $L_0, L_1, L_2, \ldots$, where L_i is the set of all vertices with distance i from the start vertex.

   Use the above execution of breadth-first search to compute the distances from the start vertex and write down these level sets.

3. Consider the following questions about level sets $L_0, L_1, \ldots$ computed by breadth-first search in directed and undirected graphs. Justify your answer.

   - In a directed graph, can there be an edge from a level set L_i with $i \geq 2$ to a level set L_j with $j \leq i - 2$?
   - In an undirected graph, can there be an edge from a level set L_i with $i \geq 2$ to a level set L_j with $j \leq i - 2$?
   - In a directed graph, can there be an edge from a level set L_i with $i \geq 0$ to a level set L_j if $j \geq i + 2$?

4. Let G be a connected undirected graph, let s be a vertex in G, and let $L_0, L_1, \ldots$ be the level sets computed by breadth-first search starting from vertex s. Prove that G is bipartite if and only if there are no edges between two vertices in the same level set L_i.

Exercise 4.3 *Asymptotic Notation.*

1. Suppose f satisfies the condition $f(n) \geq 1$ for all $n \geq 1$. Show that if $g \leq O(f)$, then for every $D \geq 0$, we have $g(n) + D \leq O(f(n))$.
2. Let $f(n) = \frac{1}{n}$, and $g(n) = f(n) + 1$. Does $g(n) \leq O(f(n))$ hold? Justify your answer.
3. In class, we defined $O(f)$ to consist of all functions $g(n)$ such that

$$\exists C > 0.\ \forall n \geq 1.\ g(n) \leq C \cdot f(n)\,.$$

   Another definition for $O(f)$, commonly found in the literature, includes all functions that satisfy the a-priori weaker condition,

$$\exists C > 0.\ \exists n_0 \geq 1.\ \forall n \geq n_0.\ g(n) \leq C \cdot f(n)\,.$$

   (This condition is a-priori weaker, because it requires the inequality $g(n) \leq C \cdot f(n)$ to hold only for all $n \geq n_0$ instead of for all $n \geq 1$.)

   Prove that the two definitions of $O(f)$ are in fact equivalent if the function f satisfies $f(n) > 0$ for all $n \geq 1$ (which is typically the case for functions that arise as running times of algorithms).
4. Show that, if we don't require f to satisfy the condition $f(n) > 0$ for all $n \geq 1$, the above two definitions of $O(f)$ are not necessarily equivalent.

   Provide concrete functions f and g such that g satisfies the second definition of $O(f)$ but not the first.

Exercise 4.4 *Pouring water **(1 Point)**.*

We have three containers whose sizes are 15 liters, 9 liters, and 5 liters, respectively. The 15-liter container starts out full of water, but the 9-liter and 5-liter containers are initially empty. We are allowed one type of operation: pouring the contents of one container into another, stopping only when the source container is empty or the destination container is full. We want to find a shortest sequence of pourings that leaves exactly 2 liters in one of the containers.

1. Model this as a graph problem: give a precise definition of the graph involved and state the specific question about this graph that needs to be answered.
2. Find a shortest sequence of pourings which leaves exactly 2 liters in one of the containers. Prove that this sequence is actually shortest.