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

15. October 2018

Algorithms & Data Structures — Homework 4 — HS 18

Exercise Class (Room & TA): _____

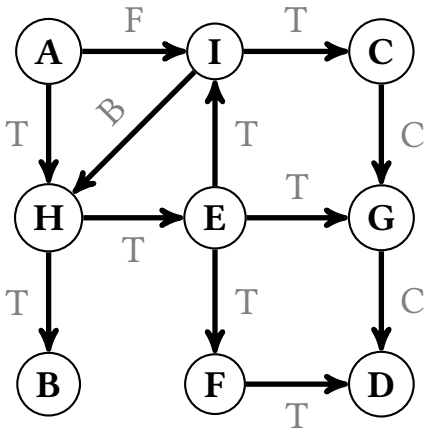
Submitted by: _____

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.)



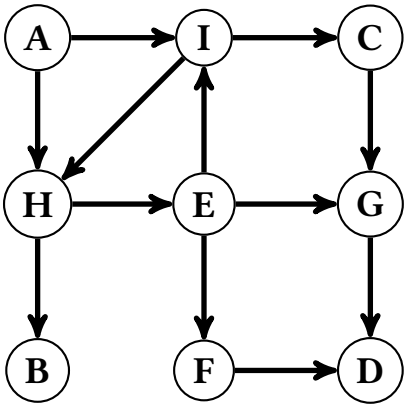
1. Mark the edges that belong to the depth-first tree (Tiefensuchbaum) with a "T" (for tree edge).
2. For each vertex, give its *pre-* and *post*-number.

   Solution: A (1,18), H (2,17), B (3,4), F (5,16), F (6,9), D (7,8), G (10,11), I(12,15), C (13,14)
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?

   Solution: The decreasing order of the post-numbers gives a topological ordering, whenever the graph is acyclic. In our graph, this is not the case, as it contains tha cycle $(H \to E \to I \to H)$. As a rule of thumb, there is no topological ordering whenever there is a backward edge in the comparison tree.

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.
   Solution: A, H, I, B, E, C, F, G, D

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.

   Solution: A: 0, H: 1, I: 1, B: 2, E: 2, C: 2, F: 3, G: 3, D: 4

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 $j \leq i - 2$?

     Solution: Consider the above graph. Add an edge from F to H. Then there is an edge from distance 3 to 1, so there can be such an edge.

   - 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 $j \leq i - 2$?

     Solution: No. Suppose there is an undirected graph with such an edge. This means that there is an edge from a vertex a in L_i to a vertex b in L_j. Because the graph is undirected, there is an edge from b to a. The distance of b is j, so the distance of the vertex a is at most $j + 1$. Therefore a cannot be in level set L_i. This is a contradiction.

   - 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 $j \geq i + 2$?

     Solution: No. Suppose there is a directed graph with such an edge. This means that there is an edge from a vertex a in L_i to a vertex b in L_j.

The level sets correspond to distances from a source vertex. Since a is in L_i, it a distance i. If there is an edge from a to b, then the distance of b can be no more than $i+1$. However, b is in the level set L_j, so by definition it has distance $j \geq i+2$. This is a contradiction.

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.

   Solution:

   - First, we prove if there are no edges between two vertices in the same level set L_i, then G is bipartite. Assume that a connected undirected graph has no edges between vertices at the same distance to a source vertex. Perform a BFS from the source vertex. Color each vertex of the graph according to its distance to the source node. Even distances (and zero) are colored red, and odd distances are colored blue.

     From the previous question (4.2.3), we know that there cannot be an edge from vertex a from level L_d to vertex b at level L_e, where $e > d+1$, and there cannot be an edge from a vertex in level L_d to a vertex in level L_e, where $e < d-1$. Therefore the only edges between levels are those edges between adjacent levels, and those levels are colored differently. There are no edges within a level, so all edges are between nodes of different colors.

     Because the graph can be colored with only two colors, it is bipartite.

   - Next we prove if G is bipartite then there are no edges between two vertices in the same level set L_i. Assume that a connected undirected graph is bipartite. Color the graph with two colors. Choose a source node and perform a BFS.

     We must argue that every vertex at some distance must be the same color. We can do this by mathematical induction.

     Base case: The only vertex at distance 0 is the source node, and it has some color.

     Inductive hypothesis: Every vertex at distance k has the same color.

     Inductive step: Assume that every vertex at distance k has the same color. Each vertex at distance $k+1$ has an edge to a vertex at distance k, so it must have the other color. By the principle of mathematical induction, every vertex at the same distance to the source node must have the same color.

     There can be no edges between vertices at the same level because the graph is bipartite, and all edges at one level have the same color.

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 $g(n) + D \leq O(f(n))$.

   Solution: When D is zero, it is trivially true. Assume $D > 0$. We know that $g \leq O(f)$. Then for some $C_0 > 0$, $\forall n \geq 1, g(n) \leq C_0 f(n)$. Add D to both sides.

   $$g(n) + D \leq C_0 f(n) + D$$

   Since $f(n) \geq 1, D \leq Df(n)$

   $$g(n) + D \leq C_0 f(n) + D \leq C_0 f(n) + Df(n) = (C_0 + D) f(n)$$

Let $C_1 = C_0 + D$. Then $\forall n \geq 1, g(n) + D \leq C_1 f(n)$

2. Let $f(n) = \frac{1}{n}$, and $g(n) = f(n) + 1$. Does $g(n) \leq O(f(n))$ hold? Justify your answer.

   Solution: No. If $g(n) \leq \mathcal{O}(f(n))$, $\exists C > 0, \forall n > 1, \frac{1}{n} + 1 \leq C\frac{1}{n}$

   But for large n, $\frac{1}{n}$ asymptotically approaches zero. The left hand side asypmtotically approaches one, and the right hand side asymptotically approaches zero, so the inequality is not true.

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).

   Solution:

   In order to prove an equivalence, we need to first assume the first property and then prove the second, and then assume the second property and prove the first.

   Assume the first property is true. $\exists C > 0, \forall n \geq 1, f(n) \leq Cg(n)$.
   Let $n_0 = 1$. Then $\exists C > 0, \forall n > n_0, f(n) \leq Cg(n)$. Thus the second definition holds.

   Assume the second property is true. $\exists C_0 > 0, \exists n_0 \geq 1, \forall n > n_0, f(n) \leq C_0 g(n)$. Let $C_1 = \max \frac{f(n)}{g(n)}, 1 \leq n \leq n_0$

   Then $\forall n, 1 \leq n \leq n_0, \frac{f(n)}{g(n)} \leq C_1$. Divide by $g(n)$. Since $g(n) > 0$,

$$f(n) \leq C_1 g(n) \forall n, 1 \leq n \leq n_0$$

   Let $C_2 = \max(C_0, C_1)$. Then:

$$f(n) \leq C_2 g(n) \forall n, 1 \leq n \leq n_0$$

   and

$$f(n) \leq C_2 g(n) \forall n \geq n_0$$

   Thus the first property is true as well.

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.

   Solution:

$$f(n) = n$$
$$g(n) = n - 1$$

The second definition holds. Let $n_0 = 2$, $C = 2$. Then $n \leq 2(n-1)$ is true as long as $n \geq 2$.

The first definition does not. We need to find a $C > 0$ such that $\forall n > 1, n \leq C(n-1)$. In particular, when $n = 1$, $1 \leq C(1-1)$. This is impossible.

Therefore the definitions are not equivalent.

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.

Solution:

1. A set of vertices is a set of all possible triples (a, b, c), such that $0 \leq a \leq 15, 0 \leq b \leq 9, 0 \leq c \leq 5$. Vertices $u = (a, b, c)$ and $v = (d, e, f)$ are connected by a directed edge (from u to v), if we can reach v from u by one pouring. We can reformulate the question in terms of this graph: Find a shortest path from the initial vertex $u_0 = (15, 0, 0)$ to some vertex $v = (a, b, c)$ such that $a = 2$, or $b = 2$, or $c = 2$.
2. To find a shortest path from $u_0 = (15, 0, 0)$ to some vertex $v = (a, b, c)$ such that $a = 2$, or $b = 2$, or $c = 2$, we start BFS from u_0. Let's write a list of groups of vertices which are at the same distances from u_0, until we find suitable v.
   - Vertices at distance $d = 1$: $(6, 9, 0)$, $(10, 0, 5)$.
   - Vertices at distance $d = 2$: $(1, 9, 5)$, $(6, 4, 5)$, $(10, 5, 0)$.
   - Vertices at distance $d = 3$: $(11, 4, 0)$, $(5, 5, 5)$.
   - Vertices at distance $d = 4$: $(11, 0, 4)$, $(5, 9, 1)$.
   - Vertices at distance $d = 5$: $(2, 9, 4)$, $(14, 0, 1)$.

   So $(2, 9, 4)$ is the closest suitable vertex. Hence the shortest sequence of pourings is

$$(15, 0, 0) \to (6, 9, 0) \to (6, 4, 5) \to (11, 4, 0) \to (11, 0, 4) \to (2, 9, 4).$$