Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Department of Computer Science
Markus Püschel
Peter Widmayer
Thomas Tschager
Tobias Pröger

15th December 2016

Algorithms & Data Structures — Solutions to Sheet 12 — AS 16

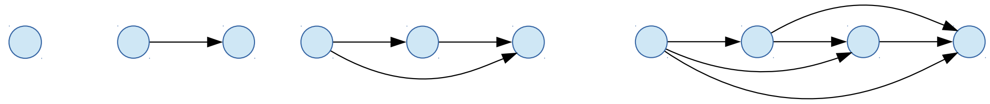
Solution 12.1 *Topological sorting and connected components.*

a) As G contains the cycles (D, B, C, D), (D, B, F, D) and (D, B, C, F, D), we have to remove at least one edge, such that G' is acyclic. If we remove the edge (D, B) (i.e., we choose $E' = \{(D, B)\}$), the graph $G' = (V, E \backslash E')$ is acyclic and, hence, has a topological order. Apparently, E' is also as small as possible, and in this case also the only possible choice (if E' would contain another edge than (D, B), then (D, B) would still be part of at least one cycle and G' would not be acyclic).

b) A topological ordering of G' is A, B, C, F, D. For G', this is even the only possible topological ordering.

c) In order to have a topological ordering, a graph has to be acyclic. The acyclic graphs with maximal number of edges for $n = 1, 2, 3, 4$ vertices are:

$n = 1$: $n = 2$: $n = 3$: $n = 4$:



We obtain the hypothesis that, in general, every acyclic graph has not more than $\sum_{i=1}^{n-1} i = n(n-1)/2$ many edges. We will prove the hypothesis by mathematical induction over n.

Base case $(n = 1)$: Every acyclic graph with a single vertex has exactly (and therefore also at most) $0 = n(n-1)/2$ many edges.

Induction hypothesis: Every acyclic graph with n vertices has at most $n(n-1)/2$ many edges.

Inductive step $(n \to n+1)$: Consider an arbitrary acyclic graph $G = (V, E)$ with $n+1$ vertices. As the graph is acyclic, it must have at least one vertex v with in-degree 0 (as shown in the lecture). If we remove this vertex and all incident edges from G, we get a new graph G' with n vertices that is also acyclic (by the removal of edges we cannot generate new cycles). By the induction hypothesis, G' has at most $n(n-1)/2$ many edges. The vertex v that has been removed from G can be connected to at most n other vertices (exactly those in G'). Therefore, the originally given graph G has at most $n(n-1)/2 + n = n(n-1)/2 + 2n/2 = n(n+1)/2$ many edges. ∎

This upper bound is indeed as good as possible, because there exists an acyclic graph with exactly that many edges. Let $V = \{v_1, \ldots, v_n\}$ be the set of vertices. Now, for every $i \in \{1, \ldots, n-1\}$ and every $j \in \{i+1, \ldots, n\}$, we create an edge (v_i, v_j). The resulting graph is acyclic (because from every vertex v_i there are only edges to vertices v_j with $j > i$, hence there is no way back to v_i), and the graph has exactly $\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$ many edges.

d) Two vertices are in the same connected component if and only if there exists a path connecting them. Therefore, a connected component is always connected. If a connected component has a minimal number of edges, it does not have cycles, as we could remove one edge on the cycle without loosing connectivity. Hence, connected components with a minimal number of edges are undirected, acyclic, connected graphs, that is trees. Trees have exactly one edge less than the number of vertices.

If a graph has k connected components that have $V_1, \ldots, V_k$ vertices (where $V_i \cap V_j = \emptyset$ and $\cup_{i=1}^{n} V_i = V$), then this graph has at least

$$\sum_{i=1}^{k}(|V_i| - 1) = \big(\sum_{i=1}^{k} |V_i|\big) - k = |V| - k = n - k \tag{1}$$

many edges.

Solution 12.2 *Depth-first search and breadth-first search.*

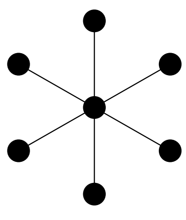
a) Depth-first search order: A, B, C, D, E, F, H, G

Breadth-first search order: A, B, F, C, H, D, G, E

b) No, because in this graph, a depth-first search from every starting vertex visits the vertices in a different order than a breadth-first search. The following orders are generated by a depth-first search and cannot be generated by a breadth-first search:

- Start at A: $A, B, C, \ldots$
- Start at B: $B, C, D, E, \ldots$
- Start at C: $C, D, E, \ldots$
- Start at D: $D, E, A, \ldots$
- Start at E: $E, A, B, \ldots$
- Start at F: $F, H, C, D, E, \ldots$
- Start at G: $G, H, C, \ldots$
- Start at H: $H, C, D, E, \ldots$

c) An example is a star-shaped graph (see below). If we start the traversal in the middle, then it clear that every breadth-first search order is equal to a depth-first search order and vice versa. If we don't start in the middle, then the next vertex is the middle one, and we are in the aforementioned situation.



d) Both breadth-first and depth-first searches have to visit every neighbor of a node at least once. Using adjacency lists, we need asymptotically exactly as many steps as the number of neighbors. In this case, the running time is in $\mathcal{O}(|V| + |E|)$, i.e., in $\mathcal{O}(|E|)$ for connected graphs.

Using an adjacency matrix, to find all the neighbors of a node we need to look through all the corresponding row/column. Since we need to do this for every node, the running time is in $\Omega(|V|^2)$.

In a very dense graph with $|E| \in \Theta(|V|^2)$, the asymptotic running time is the same. In "sparse" graphs (for example when $|E| \in \mathcal{O}(|V|)$), the use of an adjacency matrix results in a much worse asymptotic running time.

Solution 12.3 *Black Holes.*

This problem can be solved analogously to the problem of "finding a star in a set", which was presented in the first lecture.

We first observe that there can be at most one black hole. Let v be a black hole. By definition, the outdegree of v is exactly 0, so $(v, v) \notin E$. Since the indegree of v is exactly $|V| - 1$, for each $v' \in V \backslash \{v\}$ there is an edge $(v', v) \in E$, and thus the outdegree of each vertex $v' \in V \backslash \{v\}$ is at least 1. Therefore, no other v' can be a black hole.

The algorithm consists of two phases. First, using $|V| - 1$ comparisons we find a vertex v that is a possible candidate for a black hole, and then with $2|V| - 1$ comparisons we check whether v really is a black hole. Let $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$ and the corresponding adjacency matrix be $A_G = (a_{ij})$ with $a_{ij} \in \{0, 1\}$, $a_{ij} = 1 \Leftrightarrow (v_i, v_j) \in E$. To determine a candidate, we use two variables i and j, initialized with $i \leftarrow 1$ and $j \leftarrow n$. Next, we examine the entry a_{ij}. If $a_{ij} = 0$, then $(v_i, v_j) \notin E$ and v_j is certainly not a black hole (because the indegree of v_j is smaller than $n - 1$ if $(v_j, v_j) \notin E$), so we set $j \leftarrow j - 1$. If, however, $a_{ij} = 1$, then we know that $(v_i, v_j) \in E$ and v_i cannot be a black hole (because the outdegree of v_i is at least 1), so we set $i \leftarrow i + 1$. After $n - 1$ such comparisons, we have $i = j$, and v_i is our candidate for a black hole. Finally, we only check if v_i really is a black hole by calculating the indegree and the outdegree of v_i. We only need to check if $a_{ij} = 0$ for $j = 1, \ldots, n$ and $a_{ji} = 1$ for $j = 1, \ldots, n$, $j \neq i$. If this is the case, then v_i is the black hole that we searched. Otherwise, G has no black hole.

The correctness of the procedure follows from the observation that the pointer i (or j) is updated only if v_i (or v_j) *is not* a black hole. On the other hand, if v_i is a black hole, then $a_{ij} = 0$ for all $j = 1, \ldots, n$. Thus, i will never be increased (only j is reduced). Similarly, j never decreases if v_j is a black hole, because in that case we have $a_{ij} = 1$ for every $i = 1, \ldots, n$, $i \neq j$.