MTH6105 – Algorithmic Graph Theory **Spring 2024**

Problem Sheet 5 **F. Fischer**

You are expected to **attempt all exercises** before the seminar and to **actively participate** in the seminar itself.

---

1. Consider a graph G, and let $e \in E(G)$.

   (a) Show that $v_0 e_1 v_1 e_2 v_2 \dots e_m v_m e v_0$ is a shortest cycle containing e in G if and only if $v_0 e_1 v_1 e_2 v_2 \dots e_m$ is a shortest v_0–v_m-path in the graph G' with $V(G') = V(G)$ and $E(G') = E(G) \setminus \{e\}$.

   (b) Give an algorithm for finding a shortest cycle in G that contains e.

   (c) Give an efficient algorithm for finding a shortest cycle in G. Show that the algorithm is indeed efficient.

   Solution:

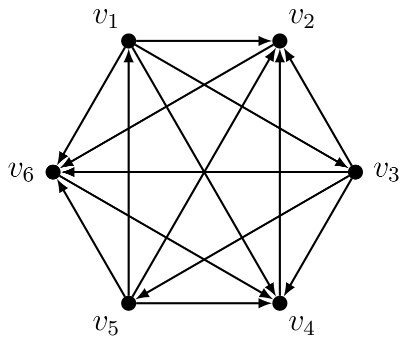
   (a) For the direction from left to right, assume that $v_0 e_1 v_1 e_2 v_2 \dots e_m v_m e v_0$ is a shortest cycle containing e in G and, for contradiction, that $v_0 e_1 v_1 e_2 v_2 \dots e_m$ is *not* a shortest v_0–v_m-path in G'. Then there exists a shorter path in G'. This path does not contain e, but together with e forms a shorter cycle in G. This is a contradiction. For the direction from right to left, assume that $v_0 e_1 v_1 e_2 v_2 \dots e_m$ is a shortest v_0–v_m-path in G' and, for contradiction, that $v_0 e_1 v_1 e_2 v_2 \dots e_m v_m e v_0$ is not a shortest cycle containing e in G. Then there exists a shorter cycle containing e in G, and removing e from this cycle yields a shorter v_0–v_m-path. This is a contradiction.

   (b) We can use breadth-first search to find a shortest u–v-path in the graph G' with $V(G') = V(G)$ and $E(G') = E(G) \setminus \{e\}$. By a similar argument as in Part (a), such a path together with the edge uv then forms a shortest cycle in G that contains uv.

   (c) To find a shortest cycle in G we can find, for every $e \in E(G)$, the shortest cycle containing e, and keep track of the shortest cycle we have found overall. For each $e \in E(G)$ we run breadth-first search on a graph no larger than G. Breadth-first search runs in time $O(|V| \cdot |E|)$, so the overall running time of our algorithm is $O(|V| \cdot |E|^2)$. This is polynomial in the size of an incidence or adjacency matrix for G, so the algorithm is efficient.

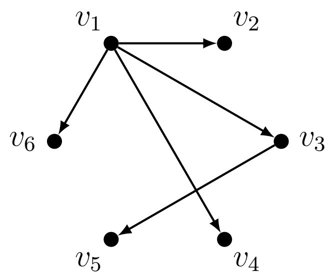
---

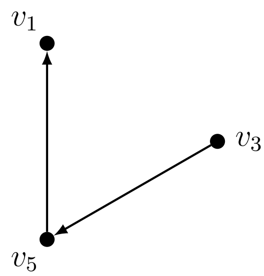
2. Find the strongly connected components of the following digraph.



Solution: Denote the graph by G. To find the strongly connected component containing a particular vertex $v \in V(G)$ we run two variants of tree search starting from v: one that adds arcs with tail in the currect tree and head outside the current tree and thus finds the set X of all vertices x such that there exists a directed v–x-path; and one that adds arcs with head in the currect tree and tail outside the current tree and thus finds the set Y of all vertices y such that there exists a directed y–v-path. The strongly connected component containing v then is $G[X \cap Y]$.

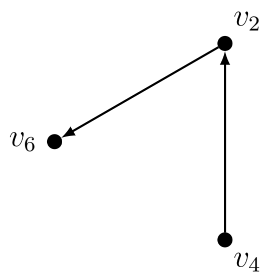
Starting from v_1, the two variants of tree search may respectively find the following trees.

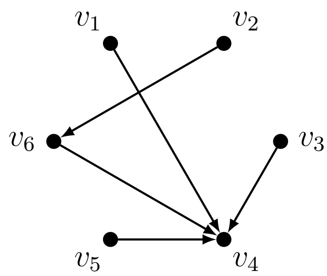




The intersection of the vertex sets of these trees is $\{v_1, v_3, v_5\}$, so $G[\{v_1, v_3, v_5\}]$ is a strongly connected components of G.

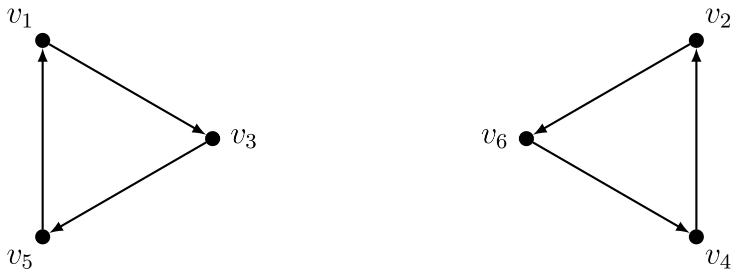
Starting from $v_4 \in V(G)\backslash\{v_1, v_3, v_5\}$, the two variants of tree search may respectively find the following trees.





The intersection of the vertex sets of these trees is $\{v_2, v_4, v_6\}$, so $G[\{v_1, v_3, v_5\}]$ is a strongly connected components of G.

Since $\{v_1, v_3, v_5\} \cup \{v_2, v_4, v_6\} = V(G)$ we have found all strongly connected components, which look as follows.



3. Consider the following spreadsheet, in which some cells contain a formula that depends on the values of other cells.

	a	b	c	d	e	f	g
1	e1+g5	a1−c5	110	a1+c1	180	f5−e1	c1+c2
2	a1+b1	a2+c4	240	a2+c2	120	f5−e2	e3+e5
3	a2+b2	a3−c3	100	a3+c1	200	f5−e3	f1+f2
4	a3+b3	a4+c2	220	a4+c2	100	f5−e4	f3+f4
5	a4+b4	a5−c1	130	a5+c5	120	g3+g4	g1+g2

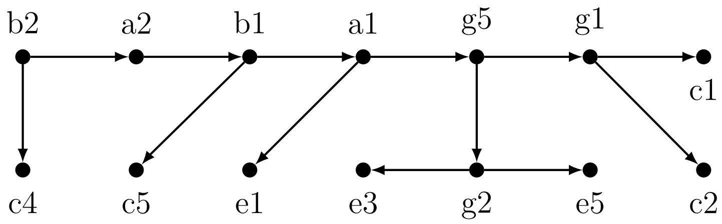
(a) Which of the values in columns c and e can be changed without changing the value of cell b2?

(b) Is it possible to compute the values of all cells in the spreadsheet? Justify your answer.

To answer these questions, you may want to consider the digraph D where $V(D)$ is the set of cells in the spreadsheet and $E(D)$ contains an arc from cell u to cell v if cell u contains a formula that depends on the value of cell v.

Solution:

(a) An entry in cell c can be changed without changing the entry in cell b2 if and only if the digraph D does *not* contain a directed b2−c-path. To find the cells c for which a directed b2−c-path *does* exist, we can use the variant of tree search that follows arcs, i.e., that in each iteration considers for inclusion arcs with tail in the current tree and head outside the current tree. This algorithm may obtain the following tree.



The cells that can be changed without changing the value of b2 are those *not* contained in the tree, i.e., c3, e2, and e4.

(b) The values of all cells can be computed if and only if they can be computed in some order, such that each value only depends on values that have already been computed. This is the case if and only if D is a directed acyclic graph. Since D contains the directed cycle $\mathrm{g3}, \mathrm{f1}, \mathrm{f5}, \mathrm{g3}$, it is not possible to compute the values of all cells.