Assignment 9 – Model Answers

Exercise 1.

(i) Specify an algorithm using depth-first search that determines the connected components of an undirected graph.

(ii) Specify an algorithm that determines the strongly connected components of a directed graph.

Solution. We use a start vertex $s \in V$, single fifo queue Q initialised to $Q = [s]$, a unary function symbol *component*, initially undefined, to capture a distinct integer value for the different connected components, a counter *count* initialised to 1, and a subset V' with initial value $V' = V$. Then we iterate the following rule:

IF $\neg$ *isempty*(Q)
THEN $v :=$ *popfront*(Q)
component$(v) :=$ *count*
$V' := V - \{v\}$
FORALL w **WITH** $(v, w) \in E \wedge w \in V'$ **DO**
pushback(Q, w)
IF *isempty*$(Q) \wedge V' \neq \emptyset$
THEN **CHOOSE** v **WITH** $v \in V'$ **IN**
pushback(Q, v)
count := *count* $+1$

In each iteration we have two possibilities:

1) The queue is not empty. Then we pop the front element v, set the label of its connected component to the current label given by the counter *count*, and push all direct neighbours w that have not yet been visited, i.e. $w \in V'$, onto the queue in some arbitrary order.
2) The queue is empty and there are still vertices for which the connected component is still undefined, i.e. $v \in V'$. We choose one of them and push them onto the queue, increment the counter *count*, as a new connected component will be built.

Exercise 2. Describe an implementation of breadth-first search on undirected graphs using a single FIFO queue containing those nodes, for which the outgoing edges still have to be scanned.

Solution. We modify the given dfs algorithm using only a single queue Q. Instead of pushing a vertex of the next layer first onto a queue Q', which becomes Q when all vertices of the current layer have been deakt with, with push the vertices directly to the end of Q.

To keep track of layers we use a unary function symbol *layer*. Initially, we only have *layer*$(s) = 0$. When processing an edge $(v, w) \in E$, where w is pushed onto Q, we set *layer*$(w) :=$ *layer*$(v) + 1$. Formally, we iterate the following rule:

IF $\neg$ *isempty*(Q)
THEN $v :=$ *popfront*(Q)
FORALL w **WITH** $(v,w) \in E \wedge$ *layer*$(w) =$ undef **DO**
pushback(Q, w)
layer$(w) :=$ *layer*$(v) + 1$

EXERCISE 3.

(i) Describe how to implement an algorithm to determine a maximum spanning tree, i.e. the sum of the edge costs shall be maximal.

(ii) What happens in the case of Prim's and Kruskal's algorithms, if negative edge costs are permitted? Is it still sensible to talk about minimum spanning trees, if negative edge costs are permitted?

SOLUTION.

(i) Let the edge-labelled graph be $G = (V, E, c)$, and let $C = \max\{c(v,w) \mid (v,w) \in E\} + 1$. Then apply an MST algorithm, e.g. the algorithm by Kruskal or Prim, to the graph $G' = (V, E, c')$ with $c'(v,w) = C - c(v,w) > 0$. Let the result be (V, E'), which is a minimum spanning tree for G'.

It is also a maximum spanning tree for G. To see this, let (V, F) be any other spanning tree. Then we have

$$\sum_{(v,w)\in E'} c'(v,w) \leq \sum_{(v,w)\in F} c'(v,w)\ ,$$

which implies

$$\begin{aligned}
\sum_{(v,w)\in F} c(v,w) &= \sum_{(v,w)\in F} (C - c'(v,w)) \\
&= (|V|-1)C - \sum_{(v,w)\in F} c'(v,w) \\
&\leq (|V|-1)C - \sum_{(v,w)\in E'} c'(v,w) \\
&= \sum_{(v,w)\in E'} (C - c'(v,w)) \\
&= \sum_{(v,w)\in E'} c(v,w)
\end{aligned}$$

(ii) As we request to obtain a spanning tree, it still makes sense to look for a minimal spanning tree, which will have a cost $\geq (|V|-1) \cdot \min\{c(v,w) \mid (v,w) \in E\}$.

If we take $C < \min\{c(v,w) \mid (v,w) \in E\}$ and replace $c(v,w)$ by $c'(v,w) = c(v,w) - C > 0$, then we can applu the algorithm of Kruskal or Prim to obtain

a minimum spanning tree (V, E') with respect to c'. As costs for all edges differe uniformly by C, this is also a minimal spanning tree with respect to c.

Furthermore, as the order of the edges remains the same for the cost functions c and c', we can get this MST by applying the algorithm directly for the original c.

EXERCISE 4.

(i) Kruskal's algorithm also works for graphs that are not connected, in which case the result is a *spanning forest*. Modify Prim's algorithm to work as well for non-connected graphs without referring to multiple calls of the algorithm.

(ii) Show that if all edge costs are pairwise different, the minimum spanning tree is uniquely defined.

(iii) A set T of edges spans a connected graph G, if (V, T) is connected. Is a minimum cost spanning set of edges necessarily a tree? Is it a tree if all edge costs are positive?

SOLUTION.

(i) Let $G = (V, E)$ have m connected components $(V_1, E_1), \ldots, (V_m, E_m)$. Choose arbitrary vertices $v_i \in V_i$ for $1 \leq i \leq m$ and add edges $E' = \{\{v_i, v_{i+1}\}, \mid 1 \leq i \leq m-1\}$. Also extend the cost function c by $c(v_i, v_{i+1}) = d > 0$ for some constant d.

Prim's algorithm will at some point add all the new edges. The resulting MST has additional costs $(m-1) \cdot d$. Omitting the additional edges gives a MST for the original graph G.

(ii) Without loss of generality we can assume that $G = (V, E)$ is connected. Let (V, E_1) be a minimal spanning tree, and let $e \in E_1$ be the edge with largest costs. Then removing e gives rise to a partition of V into disjoint vertex sets V_1 and V_2, with $e = (v_1, v_2)$ for some $v_i \in V_i$.

If there exists a different minimal spanning tree (V, E_2) not containing the edge e, then there must be an edge $e' \in E_2$ with $e' = (w_1, w_2)$ and $w_i \in V_i$. If we replace e in E_1 by e', we obtain another spanning tree with lower costs contradicting the minimality.

(iii) Take $G = (V, E)$ with $V = \{a, b, c\}$ and $E = \{\{a, b\}, \{b, c\}, \{a, c\}\}$ and costs $c(a, b) = -1$, $c(b, c) = -2$ and $c(a, c) = -3$. Clearly, $T = E$ defines a minimum cost spanning set, but it is not an MST.

If all edge costs are positive and there is a cycle in a spanning set, then removing any edge of the cycle preserves the connectivity property, but reduces edge costs. Hence, if all edge costs are positive, a minimum cost spanning set can only be a tree.