Assignment 10 – Model Answers

EXERCISE 1. Suppose the edges of a graph are presented to you only once (for example over a network connection) and you do not have enough memory to store all of them. The edges do not necessarily arrive in sorted order.

> Outline an algorithm that nevertheless computes a minimum spanning tree using space in $O(|V|)$.

SOLUTION. Maintain a possibly partial set of edges E' that is intially empty, and an array of length V for the representation of connected components.

Let $e \in E$ be a new incoming edge. If $E' \cup \{e\}$ contains a cycle—this is the case, if both endpoints v and w of e are in the same connected component—then choose the edge $e' \in E'$ in this cycle with maximal cost $c(e')$ and replace e' by e. In this way we guarantee that E' never contains more than $|V| - 1$ edges, so the space complexity is in $O(|V|)$.

When we have only one connected component, we have reached a spanning tree, so following incoming edges can at most reduce the costs. This is the same as what would have resulted, if e had been explored before e', which is what Kruskal's algorithm does. Hence the final result will be a minimum spanning tree.

EXERCISE 2. Finding the quickest routes in public transportation systems can be modelled as a shortest-path problem for an acyclic graph. Consider a bus, tram or train leaving a place p at time t and reaching its next stop p' at time t'. This connection is viewed as an edge connecting nodes (p, t) and (p', t'). Further, for each stop p and subsequent events (arrival and/or departure) at p, say at times t and t' with $t < t'$, we have the waiting link from (p, t) to (p', t').

(i) Show that the graph obtained in this way is a directed acyclic graph.

(ii) Suppose you have computed the shortest-path tree from your starting node in space and time to all nodes in the public transportation graph reachable from it. How do you actually find the route you are interested in?

SOLUTION.

(i) If there were a cycle

$$(p_1, t_1) \to (p_2, t_2) \to \cdots \to (p_n, t_n) \to (p_1, t_1) \ ,$$

then by definition we would have $t_1 < t_2 < \cdots < t_n < t_1$, which is impossible. Hence the graph is acyclic.

(ii) When creating the shortest path tree we record for each vertex its parent in the tree. Thus, following the parent links from the target stop to the current location, i.e. to the root of the tree, gives the shortest path in reverse order.

EXERCISE 3. A *tournament* is a directed graph, in which there is exactly one edge between every two vertices.

(i) How many edges does a tournament have?

(ii) How many different tournaments with n edges can be created?

(iii) Can each tournament be topologically sorted?

SOLUTION.

(i) Let $|V| = n$. As there is exactly one edge (in some direction) between any pair of vertices $v, w \in V$ with $v \neq w$, there must be $\frac{n(n-1)}{2}$ edges.

(ii) For each edge we have two possible orientations, so the number of different tournaments is $2^{\frac{n(n-1)}{2}} = \sqrt{2^{n(n-1)}}$.

(iii) A tournament may contain a cycle $v_1 \to v_2 \to \cdots \to v_k \to v_1$ with pairwise different vertices v_i. According to the definition of topological order this would imply $v_1 \leq v_2 \leq \cdots \leq v_k \leq v_1$, which is impossible. Hence not every tournament can be topologically sorted.

EXERCISE 4.

(i) In some instances of the greedy TSP algorithm it is possible to find a shorter optimal tour if the salesman is allowed to visit a city more than once. Construct an example for this.

(ii) A distance matrix is *Euclidean* iff it satisfies the triangle inequality: $d(i, j) \leq d(i, k) + d(k, j)$ for all i, j, k. Show that in this case it is never advantageous to pass through the same city several times.

(iii) Specify a greedy algorithm for the TSP with a Euclidean distance matrix, which produces tours with at most twice the length of an optimal tour.

SOLUTION.

(i) Take an edge-weighted graph with $V = \{1, 2, 3, 4, 5\}$ defined by the following incidence matrix:

$$\begin{pmatrix} 0 & 3 & 29 & 21 & 1 \\ 3 & 0 & 117 & 101 & 2 \\ 29 & 117 & 0 & 167 & 25 \\ 21 & 101 & 25 & 0 & 200 \\ 1 & 2 & 167 & 200 & 0 \end{pmatrix}$$

Then the edges $\{1, 2\}, \{2, 5\}, \{1, 5\}, \{1, 3\}, \{3, 4\}, \{1, 4\}$ define a complete tour with total costs 81, in which vertex 1 is visited twice. The greedy TSP algorithm will construct this tour, if multiple visit are permitted.

If vertex 1 is not visited twice, at least one of the remaining edges must be used, so the costrs will exceed 100.

(ii) Suppose we have a complete tour, in which one vertex v is visited multiple times. Then there are edges e_1, e_2, e_3, e_4 that are in the tour and incident to v. Let $e_3 = \{v, w_1\}$ and $e_4 = \{v, w_2\}$. If we replace e_3 and e_4 by the edge $e_5 = \{w_1, w_2\}$, the costs do not increase, as $d(w_1, w_2) \leq d(v, w_1) + d(, w_2)$, and the double visit is omitted from the tour.

(iii) First, create a MST (V, E') using the greedy algorithm by Kruskal or Prim. Then E' defines a tour through G following each edge in E' exactly twice:procedd in depth-first way and baxcktrack, when there is no more incident edge in E' that has not yet been explored.

As every edge is used exactly twice, the cost of this tour is $2 \sum_{e \in E'} c(e)$. Then for every backtracking path apply the replacement from (ii), which may only reduce the costs.

If we have an optimal tour with edges in E'', then the omission of a single edge e from E'' defines a spanning tree, and we must have $\sum_{e \in E'} c(e) \leq \sum_{e \in E''} c(e)$. Hence the costs for the constructed tour are at most $2 \sum_{e \in E''} c(e)$ as claimed.