GENOME-SCALE ALGORITHM DESIGN
by Veli Mäkinen, Djamal Belazzougui, Fabio Cunial and Alexandru I. Tomescu
Cambridge University Press, 2015
www.genome-scale.info

Exercises for Chapter 4. Graphs

4.1 Find a family of DAGs in which the number of topological orderings is exponential in the number of vertices.

4.2 Find a family of DAGs with a number of distinct paths exponential in the number of vertices.

4.3 Show that a directed graph G is acyclic (does not contain a directed cycle) if and only if for every subset of vertices $S \subseteq V(G)$, there exists a vertex $v \in S$ such that no out-neighbor of v belongs to S. Conclude that a DAG must have at least one sink.

4.4 Show that a directed graph G is acyclic if and only if it admits a topological ordering (the forward implication is Theorem 4.1). Conclude that we can check in $O(m)$ time whether a directed graph is acyclic.

4.5 Let G be a DAG with precisely one source s and one sink t. Show that for any $v \in V(G) \setminus \{s,t\}$ there exists a path from s to v and a path from v to t.

4.6 Let G be a DAG with a unique source. Prove that G is connected.

4.7 Explain in detail how the connectivity assumption is exploited in the proof of Theorem 4.3.

4.8 Show how the procedure in Theorem 4.3 can be implemented in time $O(m)$.

4.9 Consider a directed graph G in which for every pair of vertices $u,v \in V(G)$ exactly one of $(u,v) \in E(G)$ or $(v,u) \in E(G)$ holds (such a graph is called a *tournament*). Show that there exists a path in G passing through all vertices exactly once (such a path is called *Hamiltonian*).

4.10 Let G be an undirected graph. Show that either G or its complement $\overline{G}$ is connected.

4.11 Give an algorithm running in time $O(n+m)$ that checks whether an undirected graph G is connected, and if not, lists all of its connected components.

4.12 A connected undirected graph G is called *2-connected* (or *bi-connected*) if removing any vertex from G (along with its incident edges) results in a connected graph. Give an algorithm running in time $O(m)$ for checking whether a graph is 2-connected.

4.13 Refine the algorithm from Exercise 4.12 so that it also outputs the 2-connected components of G (that is, the maximal 2-connected subgraphs of G).

4.14 Given an undirected graph G, consider the graph $B(G)$ of 2-connected components of G: the vertices of $B(G)$ are the 2-connected components of G, and we add an edge between two vertices of $B(G)$ if the corresponding 2-connected components of G have a vertex in common. Show that $B(G)$ is a tree.

4.15 Consider a directed graph G, a vertex $s \in V(G)$, and $c : E(G) \to \mathbb{Q}$ so that no cycle of G has negative cost. Show that if G has only two 2-connected components C_1 and C_2, sharing a vertex v, such that $s \in V(C_1)$, then we can solve the shortest-path problem on G by solving it independently on C_1 and C_2, by appropriately initializing the dynamic programming algorithm on C_2 at vertex v.

4.16 Consider a DAG G, a vertex $s \in V(G)$, a cost function $c : E(G) \to \mathbb{Q}$, a partition $\mathcal{S} = \{S_1, \ldots, S_k\}$ of $V(G)$, and an integer $t \le k$. Give a dynamic programming algorithm for computing a shortest path from s to any other vertex of G that changes the partite sets of $\mathcal{S}$ at most t times. What is the complexity of your algorithm? What if G is not acyclic?

4.17 Consider a DAG G, a partition $\mathcal{S} = \{S_1, \ldots, S_k\}$ of $V(G)$, and an integer $t \le k$. We say that a path $P = u_1, u_2, \ldots, u_\ell$ in G is *t-restricted* if every maximal subpath of P of vertices from the same set of $\mathcal{S}$ has at least t vertices. (In other words, if P starts using vertices from a set $S_i \in \mathcal{S}$, then it must do so for at least t vertices.) Give a dynamic programming algorithm that is additionally given a vertex $s \in V(G)$ and a cost function $c : E(G) \to \mathbb{Q}$, and finds a t-restricted shortest path from s to any other vertex of G. What is the complexity of your algorithm? What if G is not acyclic?

4.18 Given a directed graph $G = (V, E)$, $n = |V|$, $m = |E|$, and a cost function $c : E \to \mathbb{Q}$, we say that the *length* of a cycle $C = v_1, v_2, \ldots, v_t, v_{t+1} = v_1$ in G, denoted $l(C)$ is t, its *cost*, denoted $c(C)$, is $\sum_{i=1}^{t} c(v_i, v_{i+1})$, and its *mean cost* is $\mu(C) = c(C)/l(C)$. Denote by $\mu(G)$ the minimum mean cost of a cycle of G, namely

$$\mu(G) = \min_{C \text{ cycle of } G} \mu(C).$$

(*a*) For each $v \in V(G)$, and each $k \in \{0, \ldots, n\}$, let $d(v, k)$ be the minimum cost of a path in G with exactly k edges, ending at v (where $d(v, 0) = 0$ by convention). Show that the bi-dimensional array d can be computed in time $O(nm)$.

(*b*) Show that

$$\mu(G) = \min_{v \in V} \max_{0 \le k \le n-1} \frac{d(v, n) - d(v, k)}{n - k}, \tag{1}$$

by showing the following facts. Consider first the case $\mu(G) = 0$, and show that

- for any $v \in V$, there exists a $k \in \{0, \ldots, n-1\}$ such that $d(v, n) - d(v, k) \ge 0$, thus, the right-hand side of (1) is greater than or equal to 0;
- each cycle of G has non-negative cost: if C is a cycle of G, then there exists a vertex v on C such that for every $k \in \{0, \ldots, n-1\}$, it holds that $d(v, n) - d(v, k) \le c(C)$; and
- a cycle C of minimum mean cost $\mu(C) = 0$ also has $c(C) = 0$; use the above bullet to show that the right-hand side of (1) is equal to 0.

Conclude the proof of (1) by considering the case $\mu(G) \ne 0$. Show that

- if we transform the input (G, c) into (G, c') by subtracting $\mu(G)$ from the cost of every edge, then the minimum mean cost of a path of (G, c') is 0, and the paths of minimum mean cost of (G, c) are the same as those of (G, c');

- since relation (1) holds for (G, c'), then it holds also for the original input (G, c).

(c) Use (a) and the proof of (b) to conclude that a minimum mean cost cycle C in G can be found in time $O(nm)$.