Round Trip with the Norwegian Trekking Association (DNT)

The student representative committee at the Department of Informatics has decided to organize a round trip in a beautiful Norwegian mountain landscape for all IFI students. This coincides with the launch of a new app by the Norwegian Trekking Association, together with public datasets, making it possible to make a well-informed route selection. Since most students at IFI have a fondness for graphs, it is inevitable that they will use the dataset to build a graph that captures enough information to algorithmically determine the best round trip.

The graph built is a directed graph $G = (V, E)$, where each node $u \in V$ represents a location, and an edge $(u, v) \in E$ represents a hiking path from u to v. No nodes have an edge to themselves.

(a) It turns out that the graph has a very special property: *Each node is part of at most one cycle*. Each cycle represents a *round trip*.

Provide an algorithm that finds all the round trips in the graph and state the runtime complexity. You may call algorithms known from the curriculum.

(b) One of the round trips from (a) is selected as the destination, where t is a node in the round trip. The trip starts at a *departure point*, which is easy to reach by public transport. A departure point is identified by nodes with in-degree 0. The last thing remaining in the planning is to choose a suitable departure point to start the trip in

order to reach t.

Fortunately, the dataset also contains information about the average time a hiker takes along a hiking path. This is given as a weight function w, such that for each edge $(u, v) \in E$, $w(u, v)$ indicates the time taken as a positive integer. We will use the weights to determine which departure point is best suited to reach the destination.

The input is a directed and weighted graph $G = (V, E)$ with weight function w, a set of departure points U, and a node t in the destination. The output is the node $s \in U$ such that the distance between s and t is as low as possible.

Provide an algorithm that effectively finds the departure point with the lowest distance to t in the destination, and state the runtime complexity. You may call algorithms known from the curriculum.