1.7 Road Trip

Andre Monteiro

Term 3, 2025

Problem

This task involves creating a Road Trip R within Austria.

R is a path through Austria where you arrive at a town by plane, visit one or more towns via roads and then leave by plane.

For example:

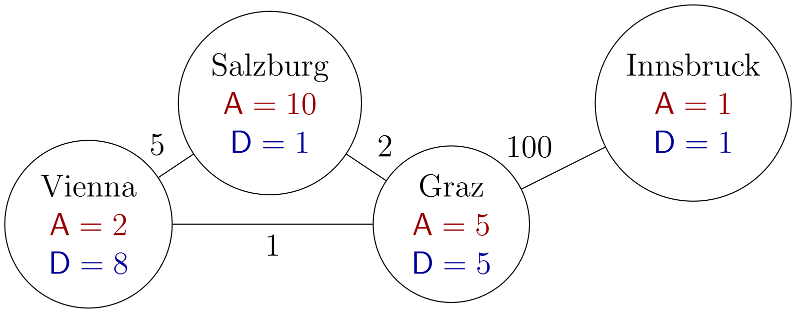

Figure 1

In Figure 1, the cheapest road trip is to enter at Vienna, drive to Graz then Salzburg, and leave via Salzburg. The total cost is

$$\mathsf{Arrive}(\text{Vienna}) + \mathsf{Toll}(\text{Vienna}, \text{Graz}) + \mathsf{Toll}(\text{Graz}, \text{Salzburg}) + \mathsf{Depart}(\text{Salzburg}) = 2 + 1 + 2 + 1 = 6.$$

While it would be cheaper to fly into Innsbruck then immediately fly out, this is not a road trip as no roads are taken. Finally, a road trip does not have to include all towns.

Solution

The intention of this problem, is to solve it via reduction, and have a resultant time complexity of:

$$O(n \log(m))$$

Together with a shortest-path graph problem, this points to Dijkstra's Algorithm as the solution. Before we can apply it, however, we need to design a graph that can be processed by the algorithm.

To be specific, we want to use Dijkstra's to find the shortest path from a **single source** to all other vertices in a graph with non-negative edge weights. The key point here, is that we need all values to appropriately be modelled as vertices, e.g. have a node for arrival and depature weights.

As such, we begin the algorithm by first inserting a node that has vertices connected to all vertices that have an arrival and/or depature cost, to model these weights accurately.

Using the previous example:

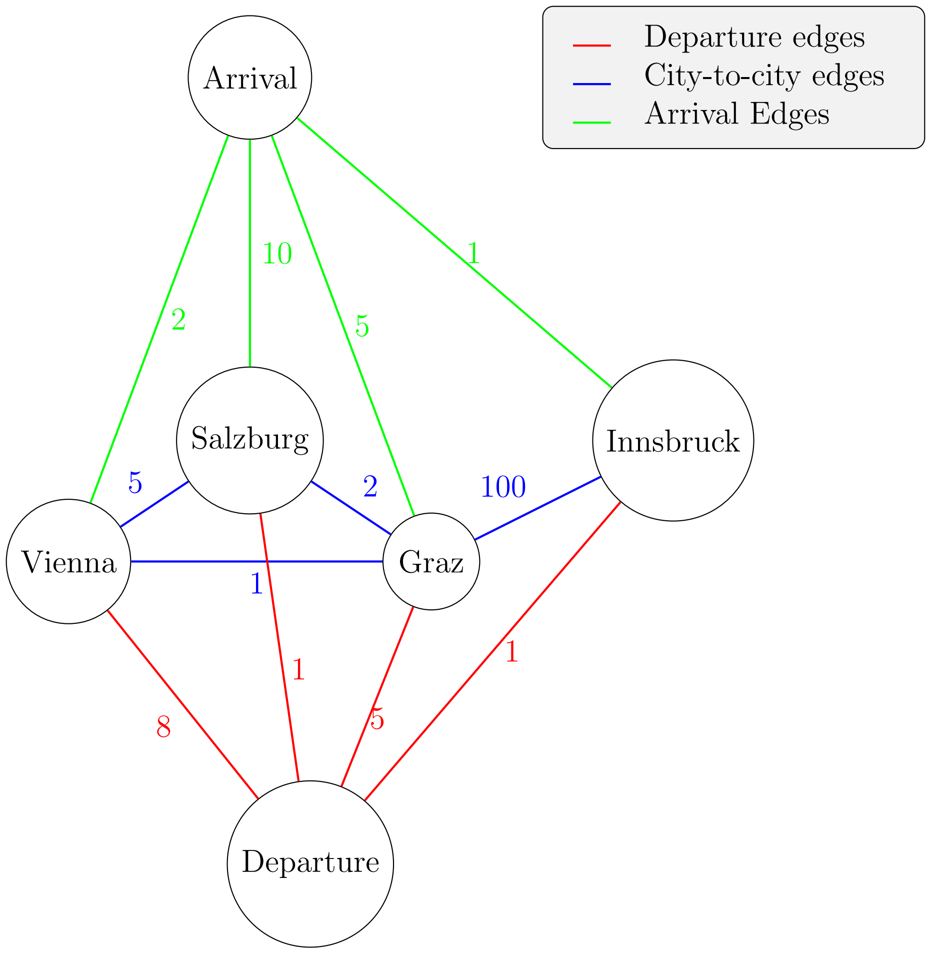

Figure 2

Time complexity:

Insertion for Arrival vertices:	$O(m)$
Insertion for Departure vertices:	$O(m)$
Total insertion:	$O(m) + O(m) = O(m)$

A small improvement we can make when considering aymptomatic complexity, is to create the vertices for Arrival and Departure as directed edges, and therefore slightly reduce time:

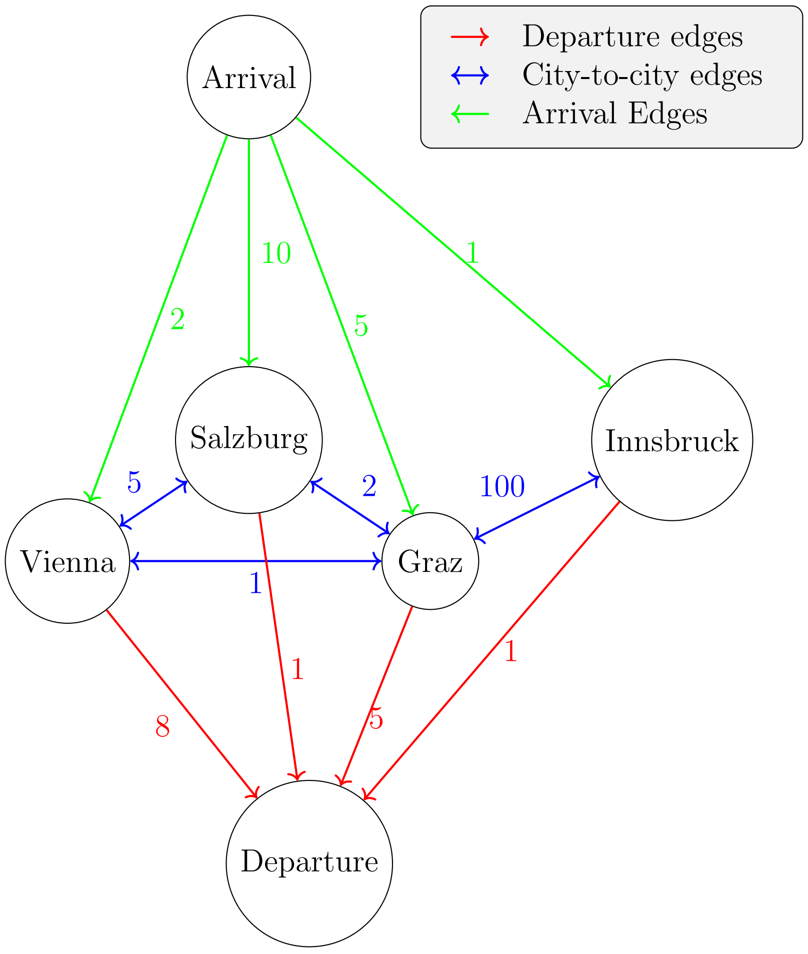

Figure 3

After adding the two reference vertices for Arrival and Departure, we now can apply Dijkstra's! However, we find that one illegal path still exists, that will be taken:

$$\mathsf{Arrive}(\text{Innsbruck}) + \mathsf{Depart}(\text{Innsbruck}) = 1 + 1 = 2.$$

We need to account for the (1) road taken rule. In order to do so, we now introduce a new layer of vertices, that act as a gateway into the rest of the network, seperating Arrival from Departure by at least one "road", e.g. now the fastest path is accurately

$$\mathsf{Arrive}(\text{Vienna}) + \mathsf{Toll}(\text{Vienna}, \text{Graz}) + \mathsf{Toll}(\text{Graz}, \text{Salzburg}) + \mathsf{Depart}(\text{Salzburg}) = 2 + 1 + 2 + 1 = 6.$$

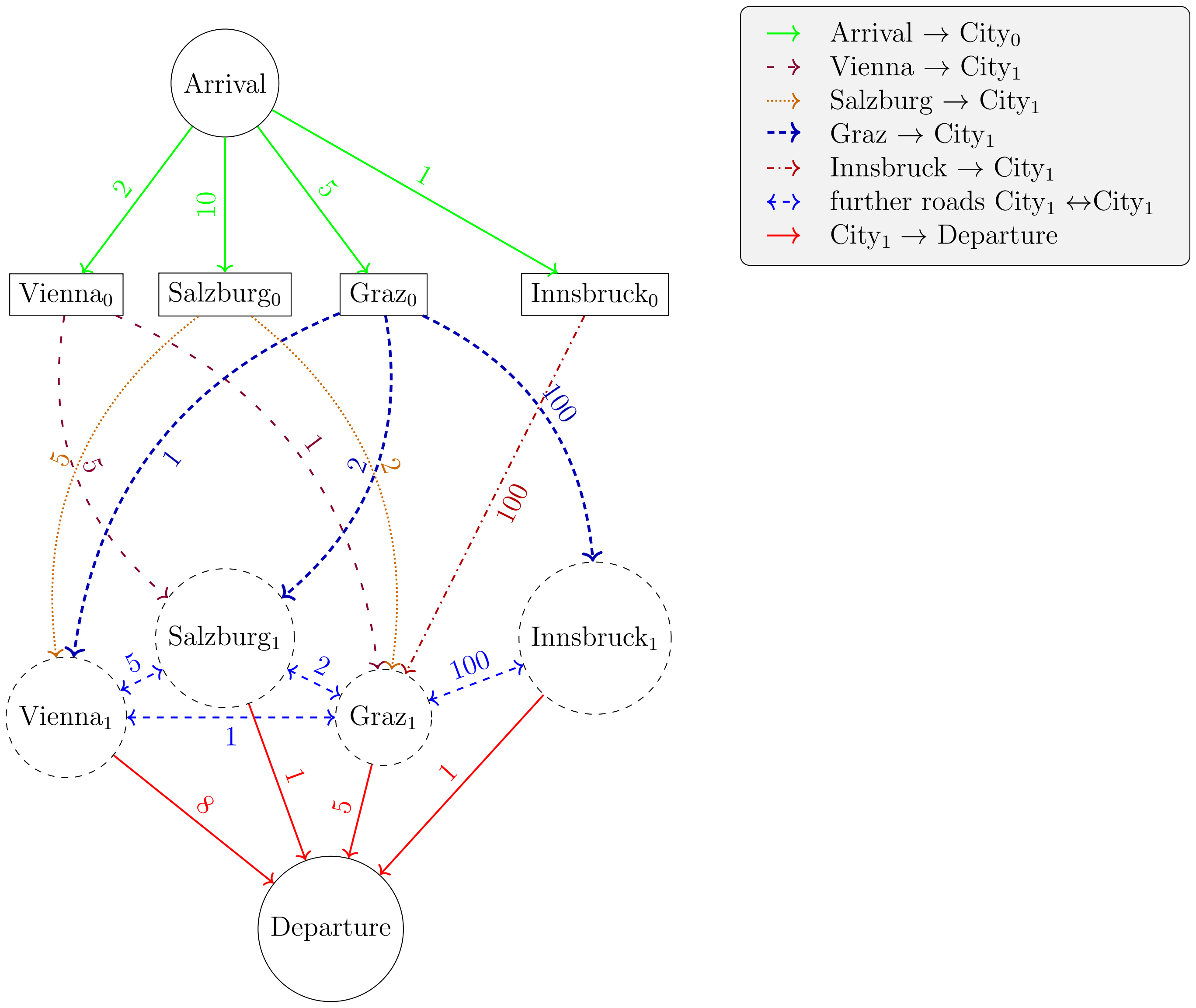

Figure 4: Layered graph with new "before road" (_0) and "after road" (_1) vertices.

Time Complexity

The time complexity of adding this new layer of vertices is

Vertices:	
Vertice vertices:	$2n$ (one City_0 and one City_1 for each original city)
Dep and Arr:	2
Total vertices:	$2n + 2 = O(n)$

Edges:	
Edges for Arrival/Departure:	$2n$ (Arrival $\to$ City_0, City_1 $\to$ Departure)
Edges per original road:	$4m$ (bidirectional original edge + $\text{City}_0 \to \text{City}_1$ edge endpoints)
Total edges:	$2n + 4m = O(m + n)$

Final Time Complexity

Total vertices:	$2n + 2 = O(n)$
Total edges:	$2n + 4m = O(m + n)$
Shortest path using Dijkstra:	$O((m + n) \log n)$
Simplification for connected graph:	$m \geq n - 1 \implies O((m + n) \log n) = O(m \log n)$

As we know that the amount of edges will always be at least double than the amount of vertices (worst case, every vertice requires at worst an edge to Arr and Dep), we can conclude that the total time is:

$$\boxed{O(m \log n)}$$