CS2040S: Data Structures and Algorithms

Recitation 10

Goals:

- Problem casting
- Identify and break down graph problems
- Exploit graph properties and algorithms

Only as Good as The Widest Link

Introduction

Google Maps team has hired you to improve their driving directions service in the US. As things are currently implemented, a user can query for the shortest path between any two cities. However, sometimes users care about other things, such as ensuring that they always maintain convenient access to a gas station along the road during their trip. For instance, they might request to be no further than 2 miles from the nearest gas station at any point along their recommended route.

Suppose US has a policy that mandates adjacent gas stations along the same road to be located apart by a *minimum* of 1 mile and a *maximum* of k miles. Moreover, two adjacent gas stations is always separated by an integer number of miles apart. For instance, there are never two adjacent gas stations at 2.5 miles apart. You may assume that k is relatively small compared to the number of cities and the number of roads.

For two cities a and b, we define the *best* route from a to b as the route which minimizes (out of all routes) the maximum separation distance between gas stations (in a single route). We refer to the maximum separation distance between adjacent gas stations along this *best* route as the *minimax* distance. If the minimax distance is 10, then it means that a driver travelling from location a to location b is guaranteed a route along which he/she will never be more than 10 miles from the nearest roadside gas station.

Consider the road network example in Figure 1, the maximum distance between adjacent gas stations is provided for each stretch of road. Suppose the start location is a and the destination location is c. Then the route $[a \to s \to e \to f \to c]$ has a gas station at most every 7 miles apart. The route $[a \to d \to g \to h \to e \to f \to c]$ also has a gas station at most every 7 miles apart and is therefore an equally best route (according to our minimax metric). By contrast, if the start location is a and the destination location is e, then the best route is $[a \to d \to g \to h \to e]$ where the minimax distance is 4 miles.

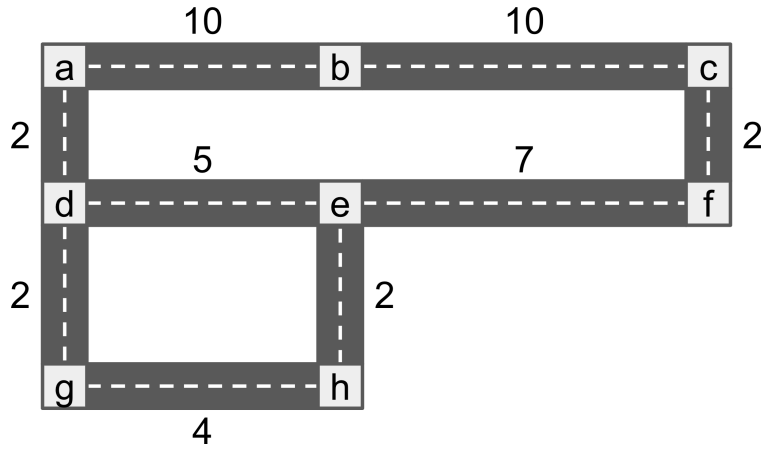

Figure 1: Letters denote cities, numbers along roads denote maximum separation (in miles) between adjacent gas stations along it.

Problem Specification

Input

Your input to the problem is a road map (much like the one in Figure 1), where each road is labelled with the maximum distance between adjacent gas stations along it. For simplicity, each city is identified by an *integer* rather than a name string. The map is provided as an *adjacency list*, i.e. an array in which each entry in the array represents a city and contains a list of roads that run out of the city. Each entry in the list (i.e. a road) consists of a pair containing the city to which the road connects to, and the maximum distance between adjacent gas stations along that road. You may assume that all the roads are two-way, and that there is a gas stations in every city (so we are only concerned about gas stations along the roads).

Requirement

Your solution should comprise of two stages:

Preprocessing stage: Initialize the service by *preprocessing* the given road map and storing it in a data structure (DS) for querying.

Query stage: Service user requests by *querying* the DS for the maximum distance between adjacent gas stations along the *best* routes recommended to them (i.e the minimax distances).

Specifically, your solution must implement the following two functions to realize each stage respectively:

1. `void preprocessMap(Vector<Vector<ii>> map)`: Reads in the map and initialize the chosen DS
2. `int queryMinimax(int start, int destination)`: Return the minimax *distance* corresponding to the best route for the given start-to-destination request (Note that your algorithm *does not* have to actually track and return the best route itself)

Note that there are many possible solutions to this problem. Different solutions have different trade-offs in terms of preprocessing cost versus querying cost: some solutions may have a very fast preprocessing step but is very slow at answering queries; other solutions may have a very slow preprocessing step but is very fast at answering queries. In this case, we would prefer queries to be relatively fast. However we also want to minimize the time spent on preprocessing the map which may be very, very big.

Problem 1. Come up with at least two solutions and present them in pseudocode. Explain why your solutions work and provide their time and space complexities. What are the tradeoffs between your solutions?