CPSC 535 Advanced Algorithms
Project 1: Savvy Traveler

35 points

Prof. Doina Bein, CSU Fullerton

dbein@fullerton.edu

Introduction

In this project you will design and implement one algorithm related to weighted undirected graphs, more specifically the nodes are arranged along a line. You will design the algorithm, describe the algorithm using clear pseudocode, and implement it using C/C++/C#/Java/Python, compile, test it, and submit BOTH the report (as a PDF file) and the source files. The execution should take less than one hour.

Electric Car Traveler

We all know that electric cars do not have as large a range as gas cars, so they need periodic recharge. Assume that one needs to travel a large distance, that cannot be done in one charge, so one needs to stop to recharge and continue. But some of the chargers may not function, in case you need to drive back to previous stop and re-charge there. To simplify, let's assume that you drive along I-10 from Santa Monica, CA to Tallahassee, FL Your stops are Phoenix, Tucson, Las Cruces, El Paso, San Antonio, Houston, Baton Rouge, New Orleans, Gulfport, Mobile, Tallahassee (https://en.wikipedia.org/wiki/Interstate_10#:~:text=Interstate%20Highway%20System,-Main&text=Major%20cities%20connected%20by%20I,Mobile%2C%20Tallahassee%2C%20and%20Jacksonville.). You are in Tucson and the charge station doesn't work, so you need to go back to Phoenix to recharge.

Given a capacity *C* in miles that represents the maximum number of miles your electric car can drive, *n* cities and (*n*-1) distances between two consecutive cities, design an algorithm that outputs the list L of cities where one need to stop and charge the car such that:

(1) L is if minimum length among all possible list of cities
(2) the starting city, which is the first city, is the first element of L
(3) The destination city, which is the last city, is the last element of L
(4) If j and k are two consecutive cities in L, then when the car is in city j, the car is able to drive to city k and back to city j, in case the charge station in city k is broken.

The capacity *C* in miles is not fixed, but one can assume that is a positive integer between 250 and 350.
The number of cities *n* is not fixed, but you can assume that it is greater than 3 and less than 20.
The distance between cities is a positive integer and always less than C/2 and greater than 10.

I need to be able to test your program with different values for C, n, and distances, within the restrictions mentioned above.

Three examples are provided below.

Example 1

Input: The graph below in Figure 1, the capacity is C = 300 miles, the starting city = A and the destination city = H, the miles between cities are shown with blue ink.

We need to compute the list of stops starting with A and ending in H such that the number of stops is minimized, in case the charge station in a stop city is broken, one can make it back to the previous city.

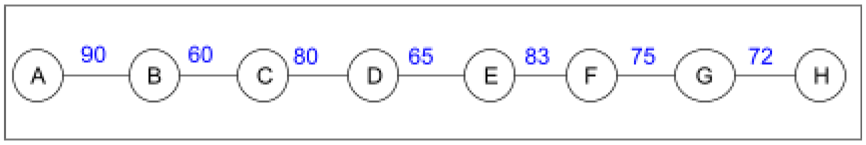

Figure 1: Example 1

Output: list L = {A, D, G, H}

Example 2

Input: The graph below in Figure 2, the capacity is C = 300 miles, the starting city = A and the destination city = H, the miles between cities are shown with blue ink.

We need to compute the list of stops starting with A and ending in H such that the number of stops is minimized, in case the charge station in a stop city is broken, one can make it back to the previous city.

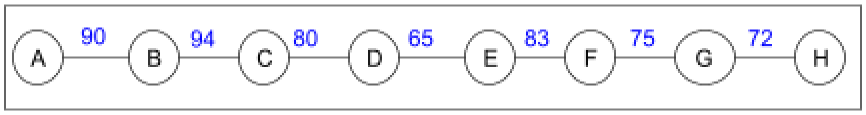

Figure 2: Example 2

Output: list L = {A, C, D, F, H}

Example 3

Input: The graph below in Figure 3, the capacity is C = 300 miles, the starting city = A and the destination city = H, the miles between cities are shown with blue ink.

We need to compute the list of stops starting with A and ending in H such that the number of stops is minimized, in case the charge station in a stop city is broken, one can make it back to the previous city.

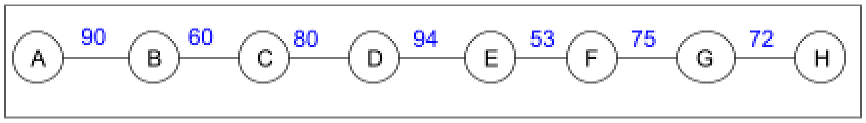

Figure 3: Example 3

Output: list L = {A, C,F,H}

Grading rubric

The total grade is 35 points. Your grade will be comprised of three parts, Form, Function, and Report:

- Function refers to whether your code works properly (18 points).
- Form refers to the design, organization, and presentation of your code. The instructor will read your code and evaluate these aspects of your submission (6 points):
    - README.md completed clearly = 2 points
    - Style (whitespace, variable names, comments, helper functions, etc.) = 2 points
    - Language Craftsmanship (appropriate handling of encapsulation, memory management, avoids gross inefficiency and taboo coding practices, etc.) = 2 points
- Report (11 points) divided as follows:
    - Summary of report document (2 points)
    - Pseudocode of the algorithm (6 points)
    - Four screenshots: one for the group members and three for the three sample input files (1 point each, total 3 points)

Obtaining and Submitting Code

This document explains how to obtain and submit your work:
GitHub Education / Tuffix Instructions

Here is the invitation link for this project:
https://classroom.github.com/a/MiejatOj

Implementation

You are provided with the following files.

1. README.md contains a brief description of the project, and a place to write the names and CSUF email addresses of the group members. You need to modify this file to identify your group members.
2. LICENSE contains a description of the MIT license.

What to Do

First, add your group members' names to README.md. Then write clear pseudocode for the rewriting algorithm, describe what parameters are needed to execute your program, and submit it as a PDF report.Your report should include the following:

1. Your names, CSUF-supplied email address(es), and an indication that the submission is for project 1.
2. A full-screen screenshot with your group member names shown clearly. One way to make your names appear in Atom is to simply open your README.md.
3. The pseudocode for the algorithms
4. A brief description on how to run the code.
5. Three snapshots of code executing for the three given examples.

Then implement your algorithm in C/C++/C#/Java/Python. Submit your PDF by committing it to your GitHub repository along with your code.