CS/ECE 374 A ✧ Fall 2019

Homework 7

Due Tuesday, October 29, 2019 at 8pm

---

1. You new job at Object Oriented Parcel Service is to help direct delivery drivers through the city of Gridville. You are given a complete street map, in the form of a graph G, whose vertices are intersections, and whose edges represent streets between those intersections. Every street in Gridville runs in a straight line either north-south or east-west, and there are no one-way streets. One specific vertex s of G represents the OOPS warehouse.

   To increase fuel economy, decrease delivery times, and reduce accidents, OOPS imposes the following strict policies on its drivers.[1]

   - U-turns are forbidden, except at dead ends, where they are obviously required.
   - Left turns are forbidden, except where the road turns left, with no option to continue either straight or right.
   - Drivers must stop at every intersection.
   - Drivers must must park as close as possible to their destination address.

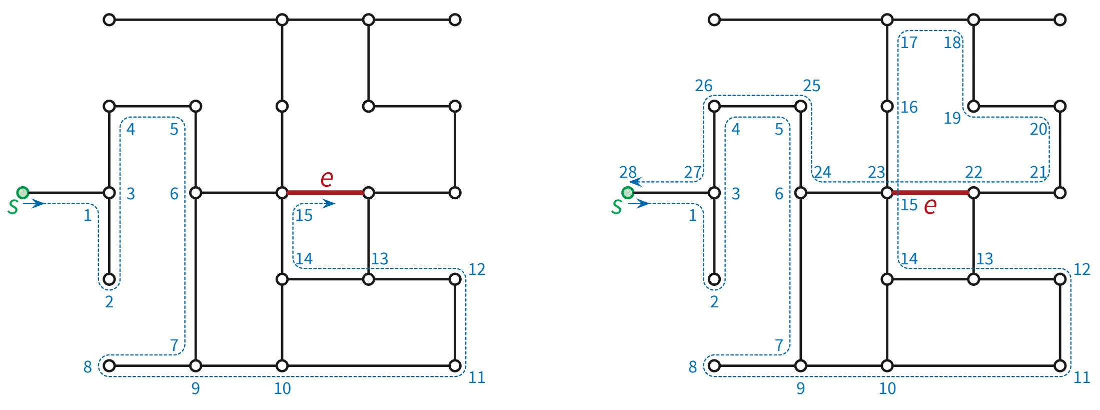
   Your job is to find routes from the OOPS warehouse to other locations in Gridville, with the smallest possible number of stops, that satisfy OOPS's driving policies. A destination is specified by an *edge* of G.

   (a) Describe and analyze an algorithm to find a legal route with the minimum number of stops from the OOPS warehouse to an arbitrary destination address. The input to your algorithm is the graph G, the start vertex s, and the destination edge; the output is the number of stops on the best legal route (or ∞ if there is no legal route).

   (b) After submitting your fancy new algorithm to your boss, you gently remind her that trucks have to return to the warehouse after making each delivery. Describe and analyze an algorithm to find a legal route with the minimum number of stops, from the OOPS warehouse, to an arbitrary destination address, and then back to the warehouse. The input to your algorithm is the graph G, the start vertex s, and the destination edge; the output is the number of stops on the best legal route (or ∞ if there is no legal route).

   For example, given the map on the next page, your algorithm for part (a) should return 15, and your algorithm for part (b) should return 28. Both optimal routes start with a forced right turn, followed by a forced U-turn, because turning left at a T intersection is forbidden. Notice that the optimal route to the destination is *not* a prefix of the optimal route to the destination and back.

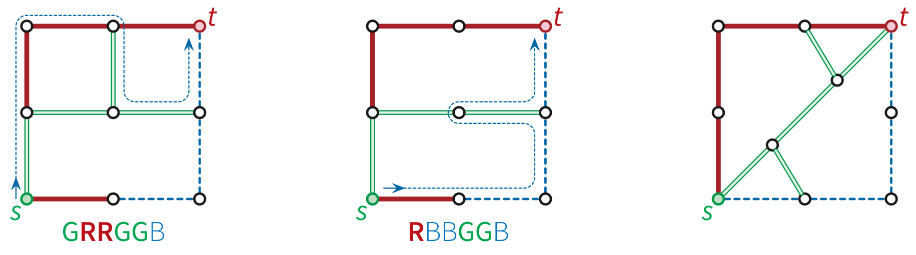
[1] OOPS maintains GPS trackers on every truck. If a driver ever breaks any of these rules, the tracker immediately shuts down and locks the truck, trapping the driver until a manager arrives, unlocks the truck, fires the driver, and installs a new replacement driver. OOPS managers are notoriously lazy, so most drivers keep enough food and water in their trucks to last several days.



Optimal delivery routes for OOPS drivers.

2. Suppose you are given an undirected graph G in which every edge is either red, green, or blue, along with two vertices s and t. Call a walk from s to t *awesome* if the walk does not contain three consecutive edges with the same color.

   Describe and analyze an algorithm to find the length of the shortest awesome walk from s to t. For example, given either the left or middle input below, your algorithm should return the integer 6, and given the input on the right, your algorithm should return ∞.



3. You are helping a group of ethnographers analyze some oral history data. The ethnographers have collected information about the lifespans of n different people, all now deceased, arbitrarily labeled with the integers 1 through n. Specifically, for some pairs (i, j), the ethnographers have learned one of the following facts:

   (a) Person i died before person j was born.

   (b) Person i and person j were both alive at some moment.

   The ethnographers are not sure that their facts are correct; after all, this information was passed down by word of mouth over generations, and memory is notoriously unreliable. So they would like you to determine whether the data they have collected is at least internally consistent, meaning there could have been people whose births and deaths consistent with their data.

   Describe and analyze an algorithm to answer the ethnographers' problem. Your algorithm should either output possible dates of birth and death that are consistent with all the stated facts, or it should report correctly that no such dates exist.

Solved Problem

4. Professor McClane takes you out to a lake and hands you three empty jars. Each jar holds a positive integer number of gallons; the capacities of the three jars may or may not be different. The professor then demands that you put exactly k gallons of water into one of the jars (which one doesn't matter), for some integer k, using only the following operations:

   (a) Fill a jar with water from the lake until the jar is full.

   (b) Empty a jar of water by pouring water into the lake.

   (c) Pour water from one jar to another, until either the first jar is empty or the second jar is full, whichever happens first.

   For example, suppose your jars hold 6, 10, and 15 gallons. Then you can put 13 gallons of water into the third jar in six steps:

   - Fill the third jar from the lake.
   - Fill the first jar from the third jar. (Now the third jar holds 9 gallons.)
   - Empty the first jar into the lake.
   - Fill the second jar from the lake.
   - Fill the first jar from the second jar. (Now the second jar holds 4 gallons.)
   - Empty the second jar into the third jar.

   Describe and analyze an efficient algorithm that either finds the smallest number of operations that leave exactly k gallons in any jar, or reports correctly that obtaining exactly k gallons of water is impossible. Your input consists of the capacities of the three jars and the positive integer k. For example, given the four numbers 6, 10, 15, and 13 as input, your algorithm should return the number 6 (the length of the sequence of operations listed above).

   Solution: Let A, B, C denote the capacities of the three jars. We reduce the problem to breadth-first search in the following directed graph:

   - $V = \{(a,b,c) \mid 0 \le a \le A \text{ and } 0 \le b \le B \text{ and } 0 \le c \le C\}$. Each vertex corresponds to a possible ***configuration*** of water in the three jars. There are $(A+1)(B+1)(C+1) = O(ABC)$ vertices altogether.
   - The graph has a directed edge $(a,b,c) \to (a', b'c')$ whenever it is possible to move from the first configuration to the second in one step. Specifically, there is an edge from (a,b,c) to each of the following vertices (except those already equal to (a,b,c)):
     - $(0,b,c)$ and $(a,0,c)$ and $(a,b,0)$ — dumping a jar into the lake
     - (A,b,c) and (a,B,c) and (a,b,C) — filling a jar from the lake
     - $\begin{Bmatrix} (0, a+b, c) & \text{if } a+b \le B \\ (a+b-B, B, c) & \text{if } a+b \ge B \end{Bmatrix}$ — pouring from jar 1 into jar 2
     - $\begin{Bmatrix} (0, b, a+c) & \text{if } a+c \le C \\ (a+c-C, b, C) & \text{if } a+c \ge C \end{Bmatrix}$ — pouring from jar 1 into jar 3

- $\left\{\begin{array}{ll}(a+b,0,c) & \text{if } a+b \le A\\ (A, a+b-A, c) & \text{if } a+b \ge A\end{array}\right\}$ — pouring from jar 2 into jar 1
- $\left\{\begin{array}{ll}(a,0,b+c) & \text{if } b+c \le C\\ (a, b+c-C, C) & \text{if } b+c \ge C\end{array}\right\}$ — pouring from jar 2 into jar 3
- $\left\{\begin{array}{ll}(a+c,b,0) & \text{if } a+c \le A\\ (A, b, a+c-A) & \text{if } a+c \ge A\end{array}\right\}$ — pouring from jar 3 into Jar 1
- $\left\{\begin{array}{ll}(a,b+c,0) & \text{if } b+c \le B\\ (a, B, b+c-B) & \text{if } b+c \ge B\end{array}\right\}$ — pouring from jar 3 into jar 2

Since each vertex has at most 12 outgoing edges, there are at most $12(A+1)\times(B+1)(C+1) = O(ABC)$ edges altogether.

To solve the jars problem, we need to find the ***shortest path*** in G from the start vertex $(0,0,0)$ to any target vertex of the form $(k,\cdot,\cdot)$ or $(\cdot,k,\cdot)$ or $(\cdot,\cdot,k)$. We can compute this shortest path by calling ***breadth-first search*** starting at $(0,0,0)$, and then examining every target vertex by brute force. If BFS does not visit any target vertex, we report that no legal sequence of moves exists. Otherwise, we find the target vertex closest to $(0,0,0)$ and trace its parent pointers back to $(0,0,0)$ to determine the shortest sequence of moves. The resulting algorithm runs in $O(V+E) = \boldsymbol{O(ABC)}$ ***time***.

We can make this algorithm faster by observing that every move either leaves at least one jar empty or leaves at least one jar full. Thus, we only need vertices (a,b,c) where either $a=0$ or $b=0$ or $c=0$ or $a=A$ or $b=B$ or $c=C$; no other vertices are reachable from $(0,0,0)$. The number of non-redundant vertices and edges is $O(AB+BC+AC)$. Thus, if we only construct and search the relevant portion of G, the algorithm runs in $\boldsymbol{O(AB+BC+AC)}$ ***time***. ■

Rubric: 10 points: standard graph reduction rubric (see next page)

- Brute force construction is fine.

−1 for calling Dijkstra instead of BFS

- max 8 points for $O(ABC)$ time; scale partial credit.

Standard rubric for graph reduction problems. For problems out of 10 points:

+ 1 for correct vertices, *including English explanation for each vertex*
+ 1 for correct edges
  - −½ for forgetting "directed" if the graph is directed
+ 1 for stating the correct problem (in this case, "shortest path")
  - − "Breadth-first search" is not a problem; it's an algorithm!
+ 1 for correctly applying the correct algorithm (in this case, "breadth-first search from $(0,0,0)$ and then examine every target vertex")
  - −½ for using a slower or more specific algorithm than necessary
+ 1 for time analysis in terms of the input parameters.
+ 5 for other details of the reduction
  - If your graph is constructed by naive brute force, you do not need to describe the construction algorithm; in this case, points for vertices, edges, problem, algorithm, and running time are all doubled.
  - Otherwise, apply the appropriate rubric, *including Deadly Sins*, to the construction algorithm. For example, for a solution that uses dynamic programming to build the graph quickly, apply the standard dynamic programming rubric.