1. Describe an algorithm to compute the minimum cost to get from your friend's house at vertex s to the Transit Authority at vertex t.

Solution: We reduce to shortest paths in a new directed graph $G' = (V', E')$ with positive edge lengths $\ell' : E' \to \mathbb{R}^+$ as follows.

- $V' := V \times \{A, B, C\}$.
  Vertex (v, a) means we're at station v while appearing to be a member of System a.
- $E' := \{(u,a)\to(v,a) \mid u\to v \in E \text{ and } a \in \{A,B,C\}\} \cup$
  $\{(v,A)\to(v,B) \mid v \in V\} \cup \{(v,B)\to(v,C) \mid v \in V\}$.
- $\ell'((u,a)\to(v,a)) := \$(u\to v)/2$ if $system(u\to v) = a$.
- $\ell'((u,a)\to(v,a)) := \$(u\to v)$ if $system(u\to v) \neq a$.
- $\ell'((v,a)\to(v,b)) := 0$ for all edges $(v,a)\to(v,b) \in E'$.

The minimum cost to get from s to t in G is equal to the distance from (s,A) to (t,C) in G'. Therefore, we run Dijkstra's algorithm from (s,A).

Our algorithm runs in $O(|E'| \log |V'|) = \boldsymbol{O(|E| \log |V|)}$ ***time***. ■

Rubric: 10 points: standard graph reduction rubric.

2. Let $G = (V, E)$ be a directed graph, $\Sigma = \{0, 1, \ldots, 9\}$ be the alphabet of decimal digits, and $\ell : V \to \Sigma$ be an assignment of digit *labels* to each vertex.

   (a) Describe an algorithm that, given the directed graph $G = (V, E)$, the labeling function $\ell : V \to \Sigma$, and two vertices $s, t \in V$, returns whether there is an (s, t)-walk W such that $\sigma(W)$ contains the substring 374.

Solution: We reduce to reachability in a new directed graph $G = (V', E')$ as follows.

- $V' := V \times \{\varepsilon, 3, 37, 374\}$.
  Vertex (v, ε) means we've walked from s to v, vertex (v, x) with $x \in \{3, 37\}$ means we've walked from s to $v \in V$ and that our walk contains a suffix W with $\sigma(W) = x$, and vertex $(v, 374)$ means we've done a walk W from s to $v \in V$ where $\sigma(W)$ contains the substring 374.
- $E' := \{(u, \varepsilon) \to (v, \varepsilon) \mid u \to v \in E\}$
  $\cup \{(u, \varepsilon) \to (v, 3) \mid u \to v \in E \text{ and } \ell(v) = 3\}$
  $\cup \{(u, 3) \to (v, 37) \mid u \to v \in E \text{ and } \ell(v) = 7\}$
  $\cup \{(u, 37) \to (v, 374) \mid u \to v \in E \text{ and } \ell(v) = 4\}$
  $\cup \{(u, 374) \to (v, 374) \mid u \to v \in E\}$

There is an (s, t)-walk W such that $\sigma(W)$ contains the substring 374 in G if and only if (s, ε) can reach $(t, 374)$ in G'. Therefore, we run WhateverFirstSearch$((s, \varepsilon))$ and return whether $(t, 374)$ gets marked.

Our algorithm runs in $O(|V'| + |E'|) = \boldsymbol{O(|V| + |E|)}$ ***time***. ■

Rubric: 3 points: standard graph reduction rubric.

Solution: As stated in part (b), the set of strings L containing the substring 374 is regular. Let $M = (Q, \delta, s', A)$ be any DFA that accepts L. We run the $O(|Q|(|V| + |E|))$ time algorithm given below for part (b) on G, s, t, and M. Machine M is fixed, so $|Q| = O(1)$, and ***the running time of the current algorithm is*** $\boldsymbol{O(|V| + |E|)}$. ■

Rubric: 3 points. Explicitly describing the DFA M is unnecessary for full credit.

(b) Describe an algorithm that, given the directed graph $G = (V, E)$, the labeling function $\ell : V \to \Sigma$, and two vertices $s, t \in V$, *and a DFA* $M = (Q, \delta, s', A)$ *over* Σ returns whether there is an (s, t)-walk W such that $\sigma(W) \in L(M)$. The running time of your algorithm should be expressed in terms of $|V|$, $|E|$, and $|Q|$.

Solution: We reduce to reachability in a new directed graph $G = (V', E')$ as follows.

- $V' := V \times Q$.
  Vertex (v, q) means we've done a walk W from s to $v \in V$ where $\delta^*(s', \sigma(W)) \in A$.
- $E' := \{(u, p) \to (v, q) \mid u \to v \in E \text{ and } q = \delta(p, \ell(v))\}$.

There is an (s, t)-walk W such that $\sigma(W) \in L(M)$ if and only if (s, s') can reach some (t, r) with $r \in A$ in G'. Therefore, we run WhateverFirstSearch$((s, s'))$ and return whether any such vertrex (t, r) gets marked.

Our algorithm runs in $O(|V'| + |E'|)) = \boldsymbol{O\left(|Q|(|V| + |E|)\right)}$ ***time***. ■

Rubric: 7 points: standard graph reduction rubric.

3. ***Practice only. Do not submit solutions.***

Describe and analyze an algorithm to solve arbitrary acute-angle mazes. You are given a connected undirected graph G, whose vertices are points in the plane and whose edges are line segments; you are also given two vertices Start and Finish. Your algorithm should return TRUE if G contains a walk from Start to Finish that has only acute angles, and FALSE otherwise. Assume you have a subroutine that can determine in $O(1)$ time whether two segments with a common vertex define a straight, obtuse, right, or acute angle.

Solution: Let $G = (V, E)$ be the input graph. Imagine moving a token along a valid walk through G. At any time during the walk, the current ***state*** of the token is described by its current vertex and its previous vertex (if any). More formally, we reduce the angle-maze problem to a reachability problem in a new directed graph $G' = (V', E')$ as follows.

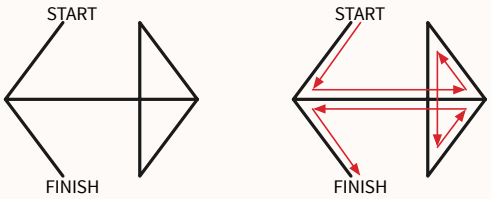
- $V' = \{(u, v) \mid uv \in E\} \cup \{\text{Start}\}$ — Each vertex (u, v) indicates that the token is currently at vertex v and it was previously at vertex u. These are *ordered* pairs; G' has *two* distinct vertices for each edge in G. In addition, we retain the Start vertex from G, because the token initially has no "previous" location.

  There are $2E + 1 = O(E)$ vertices altogether.
- There are two types of edges:
  - Starting edges $\{(\text{Start}, (\text{Start}, v)) \mid (\text{Start}, v) \in E\}$ — At the beginning, the token can follow any edge out of the Start vertex.
  - Regular edges $\{((u, v), (v, w)) \mid \angle uvw \text{ is straight or acute}\}$ — At every vertex in the walk after Start, the token can either continue straight or make an acute-angle turn.

  Edges are directed. The total number of edges is $O(EV)$, because there is at most one edge in G' for each $uv \in E$ and each $w \in V$.
- ***The maze is solvable if and only if the Start vertex can reach any vertex of the form (v, Finish) in G'.***
- We can solve this problem using whatever-first search in G'. Specifically, we mark every vertex in G' that is reachable from Start, and then scan through all vertices (v, Finish) to see if any is marked.

Our overall algorithm runs in $O(|V'| + |E'|) = \boldsymbol{O(|E||V|)}$ ***time***. ■

Non-solution: It is not possible to solve the problem by running any version of WFS on the original input graph, because the solution walk can revisit the same vertex *up to four* times, but WFS always computes *simple paths*. See the maze below and its unique solution.



♣

Rubric: 10 points: standard graph-reduction rubric

- −1 for Dijkstra
- −1 for "$O(|E|^2)$ time" or "$O(|V|^3)$ time" instead of $O(|E||V|)$ time.