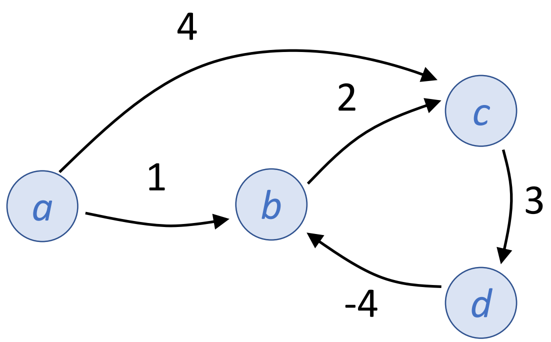
SECTION 5.7

Using dynamic programming to find shortest paths

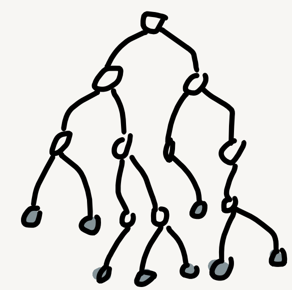


I'd like to find a minimum-weight path from a to d.
Can I use dynamic programming for this?

3.1 The Bell… …mming

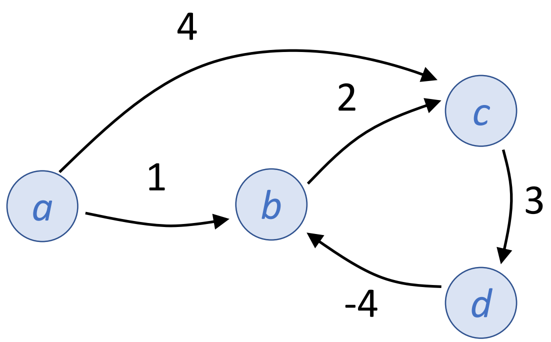
Let $v(x)$ be the to…

$$v(x) = \begin{cases} \text{term}\ldots \\ \max\limits_{a \in A} \ldots \end{cases}$$



How can I frame my task as "find an optimal sequence of actions"?

- What are the actions?
- What is the value/cost that I'm optimizing?



let's turn the "shortest path" problem into a problem with a deadline.

let $F_t(v, n)$ = minweight among all paths $v \rightsquigarrow t$ that have $\le n$ edges.

e.g. $F_d(v, 1) = \begin{cases} \text{if } v = c: & 3 \\ \text{if } v = a, b: & \infty \\ \text{if } v = d: & 0 \end{cases}$

I'll consider d to be a path from d to d with 0 edges and weight 0

$F_d(v, 2) = \begin{cases} \text{if } v = b: & 5 \\ \text{if } v = a: & 7 \\ \text{if } v = c: & 3 \\ \text{if } v = d: & 0 \end{cases}$

paths of length 2 (for $v = b$ and $v = a$)

path of length 1 (for $v = c$)

path of length 0 (for $v = d$)

General case:

$$F_t(v, n) = \min\left(\min_{w:\, v \to w} \left\{ \text{weight}(v \to w) + F_t(w, n-1) \right\},\ F_t(v, n-1) \right)$$

Boundary condition:

$F_t(v, 0) = \begin{cases} \text{if } v = t: & 0 \\ \text{if } v \neq t: & \infty \end{cases}$

path "v" of length 0 is valid, has weight 0

There are no paths from v to t of length 0.

Theorem

Let g be a directed graph where each edge is labelled with a weight.
Assume g has no –ve weight cycles.

Then, $F_t(s, |V|-1)$ is the minimum weight from s to t over paths of *any* length.

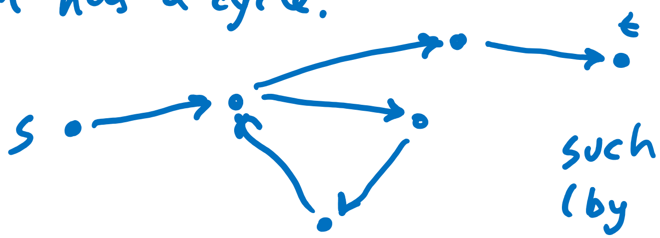
in words, to find minweight path, it's sufficient to look only at paths with $\leq |V|-1$ edges

Algorithm

To find a minweight path from s to t, just compute $F_t(s, |V|-1)$
then reconstruct the optimal programme as usual, by replaying the optimal actions.

EXERCISE. Add in a detection subroutine (similar to Bellman-Ford) that detects whether g satisfies the assumption.

Proof of theorem
From the set of minweight paths from s to t, pick one with the least number of vertices. Suppose it has $> |V|$ vertices. Then some vertex is repeated, so the path has a cycle.



such a cycle has weight ≥ 0
(by precondition of theorem)

so if we cut it out we get a path that's shorter (fewer vertices) and at least as good. #

So, the path has $\leq |V|$ vertices, $\therefore$ has $\leq |V|-1$ edges.

Thus, between s and t, minweight over all paths $s \to t$ = minweight over all paths $s \to t$ of $\leq |V|-1$ edges = $F_t(s, |V|-1)$.

$$F_t(v,n) = \min\Big(F_t(v,n-1),\ \min_{w:v\to w}\{\text{weight}(v\to w) + F_t(w,n-1)\}\Big)$$

$$F_t(v,0) = \begin{cases} 0 & \text{if } v = t \\ \infty & \text{if } v \neq t \end{cases}$$

Algorithm

To find a minweight path from s to t, just compute $F_t(s,|V|-1)$
then reconstruct the optimal programme as usual.

Running time

To fill in a row, $O(V+E)$

To fill in the table, $O(V^2+VE)$

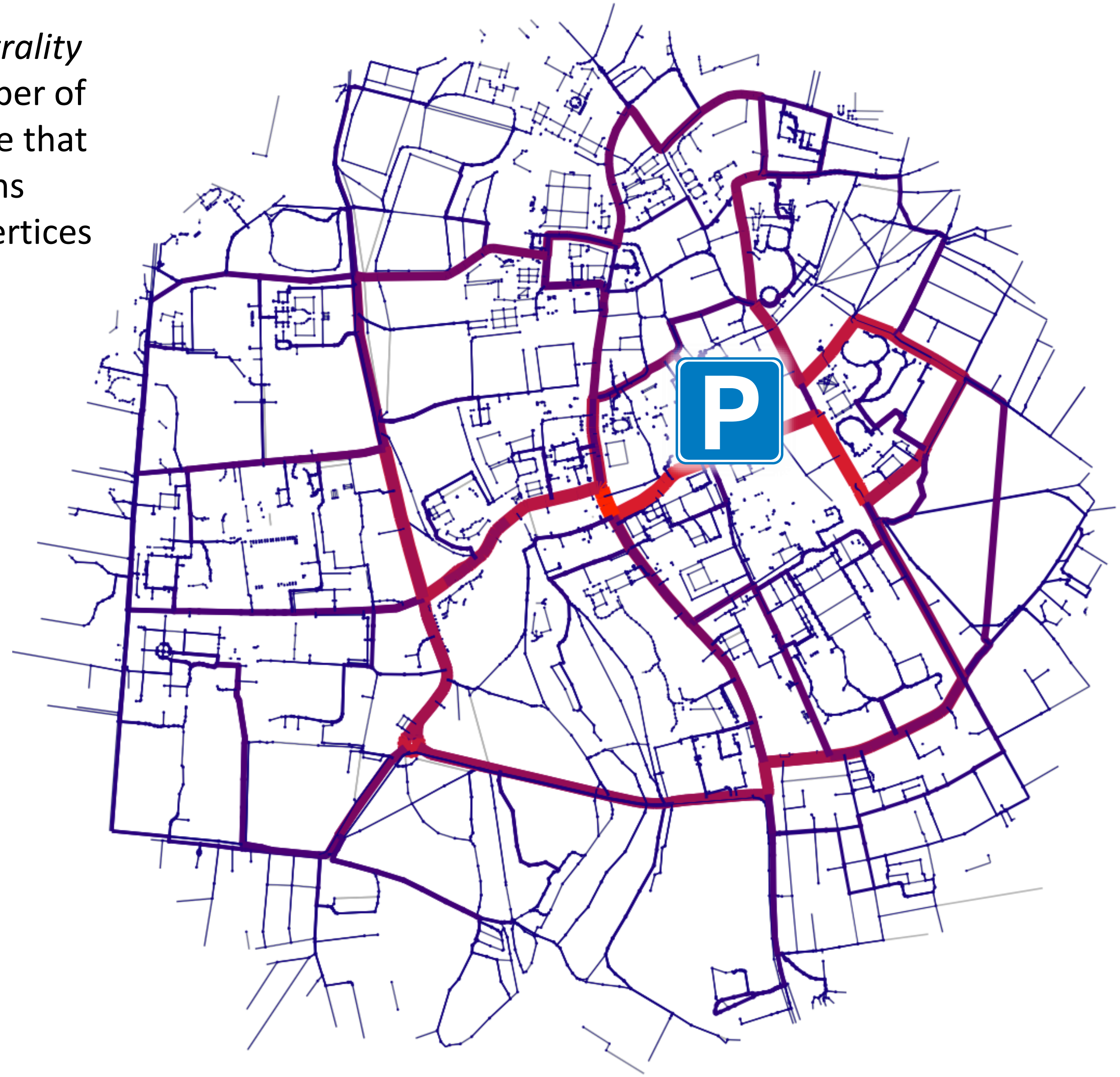
		Tree  $E = V-1$	Fully connected  $E = V(V-1)$	Intermediate $E = \Theta(V^{\alpha})$, $\alpha \in [1,2]$
Dijkstra if all weights ≥ 0	$O(E + V\log V)$	$O(V\log V)$	$O(V^2)$	$O(V^{\alpha} + V\log V)$ This term dominates for $\alpha > 1$.
Dijkstra if some weights < 0	??? (might not even terminate)			
Bellman-Ford	$O(VE)$	$O(V^2)$	$O(V^3)$	$O(V^{1+\alpha})$
dynamic prog.	$O(V^2 + VE)$	$O(V^2)$	$O(V^3)$	$O(V^{1+\alpha})$

SECTION 5.8

Finding all-to-all shortest paths

Definition

The *betweenness centrality* of an edge is the number of shortest paths that use that edge, considering paths between all pairs of vertices in the graph



What's the cost of finding all-to-all minimum weights?

	cost	cost if $\lvert E\rvert = \lvert V\rvert^{\alpha}$, $\alpha \in [1,2]$
$V \times$ Dijkstra for weights ≥ 0	$V \times O(E + V \log V)$	$O(V^{1+\alpha} + V^2 \log V)$
$V \times$ Bellman-Ford	$V \times O(VE)$	$O(V^{2+\alpha})$
$V \times$ dyn.prog.	$V \times O(V^2 + VE)$	$O(V^{2+\alpha})$
dynamic prog. with matrix trick	$O(V^3 \log V)$	$O(V^3 \log V)$
Johnson	same as Dijkstra, but works with –ve edge weights	

See Discrete Maths lecture 14 for how to write the Bellman recursion as a matrix multiplication.

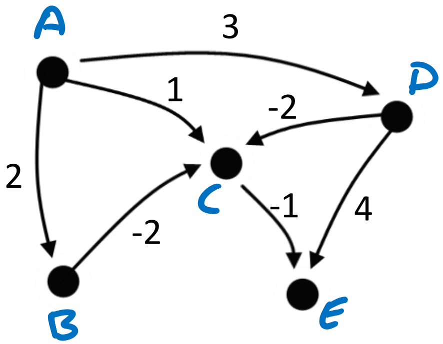
The $(n \times n)$-matrix $M = \mathrm{mat}(R)$ of a finite directed graph $([n], R)$ for n a positive integer is called its *adjacency matrix*.

The adjacency matrix $M^* = \mathrm{mat}(R^{\circ *})$ can be computed by matrix multiplication and addition as M_n where

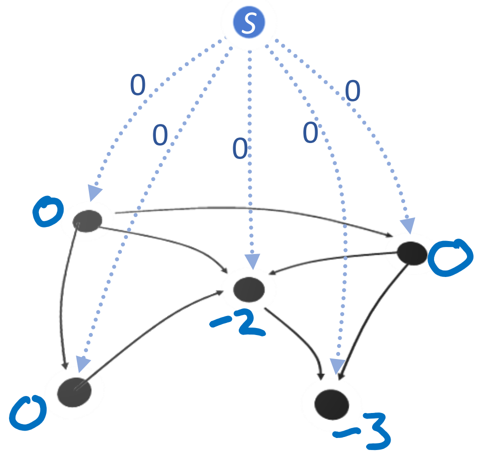
$$\begin{cases} M_0 = I_n \\ M_{k+1} = I_n + (M \cdot M_k) \end{cases}$$

This gives an algorithm for establishing or refuting the existence of paths in finite directed graphs.

Johnson’s algorithm

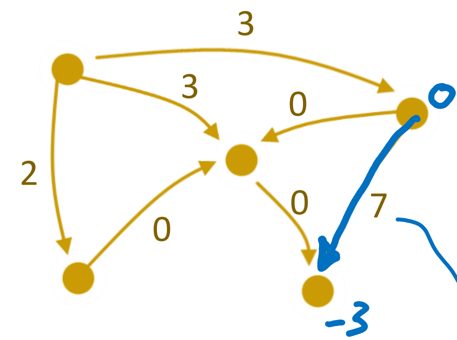


0. The graph where we want all-to-all minweights
Denote the edge weights by $w(u \to v)$



1. The augmented graph
Add a new vertex s, and run Bellman-Ford to compute minimum weights from s,

$$d_v = \text{minweight}(s \text{ to } v)$$



2. The helper graph
Define a new graph with modified edge weights

$$\underline{w'(u \to v)} = d_u + w(u \to v) - d_v$$

$w' = 0 + 4 - (-3) = 7$

3. Run Dijkstra to get all-to-all distances in the helper graph, $\text{distance}'(u \text{ to } v)$

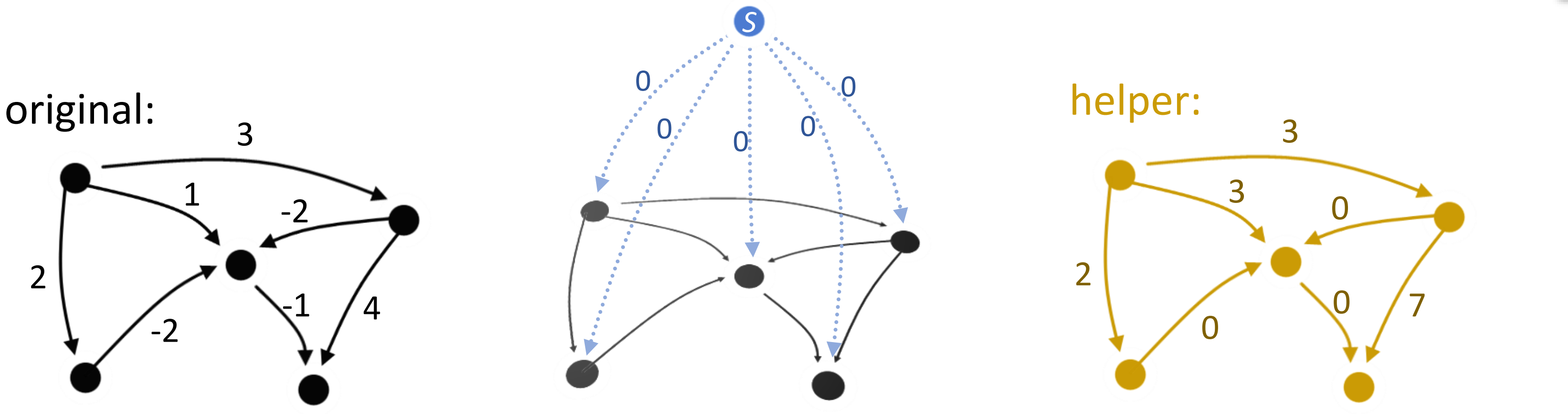
CLAIM: $w' \geq 0$ on all edges.

4. Translation

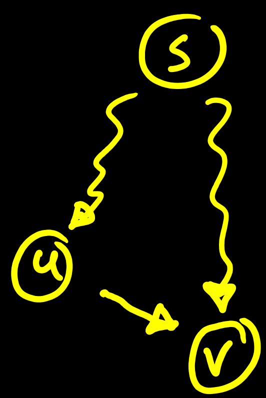
$$\text{minweight}(p \text{ to } q) = \text{distance}'(p \text{ to } q) - d_p + d_q$$

CLAIM. This computes correct minweights in the original graph.

Lemma. The edge weights in the helper graph are all ≥ 0



edge weights $w(u \to v)$ $\quad d_v = \text{minweight}(s \text{ to } v)$ $\quad w'(u \to v) = d_u + w(u \to v) - d_v$



Consider paths from s in the augmented graph.

We know, from edge relaxation, that

$$d_v \leq d_u + w(u \to v)$$

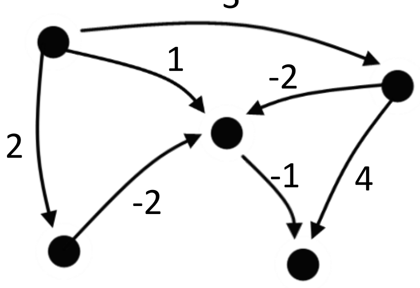
Rearranging,

$$\underbrace{d_u + w(u \to v) - d_v}_{w'(u \to v)} \geq 0$$

Lemma. The translation step computes correct minweights:

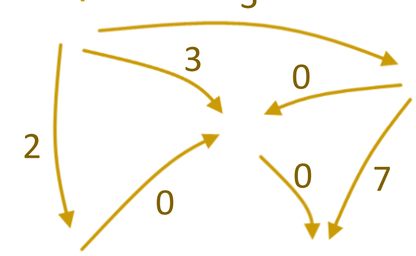
$$\text{minweight}(p \text{ to } q) = \text{distance}'(p \text{ to } q) - d_p + d_q$$

original:



edge weights $w(u \to v)$

helper:



$w'(u \to v) = d_u + w(u \to v) - d_v$

Stronger claim:

For every path $p \rightsquigarrow q$,

weight in original = weight in helper $- d_p + d_q$

Dijkstra finds least-weight paths in helper graph. Because the ordering of paths is the same, it finds least-weight paths in original graph.

Proof: Consider any path $p = v_0 \to v_1 \to \cdots \to v_k = q$

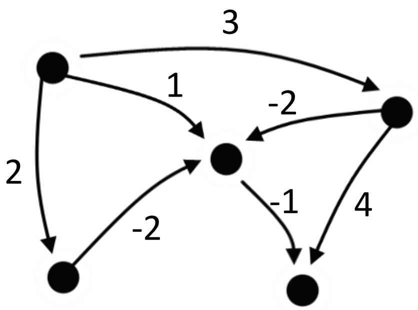
weight in original: $w(v_0 \to v_1) + w(v_1 \to v_2) + \cdots + w(v_{k-1} \to v_k)$

weight in helper: $w'(v_0 \to v_1) + w'(v_1 \to v_2) + \cdots + w'(v_{k-1} \to v_k)$

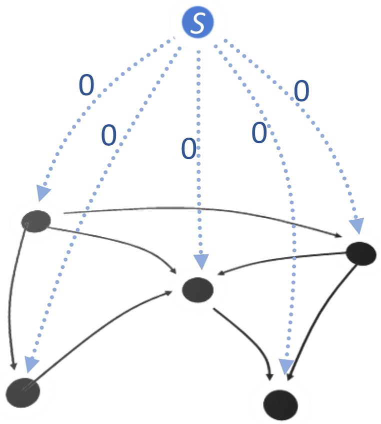
$$= d_{v_0} + w(v_0 \to v_1) - \cancel{d_{v_1}} + \cancel{d_{v_1}} + w(v_1 \to v_2) - \cancel{d_{v_2}} \cancel{+ \cdots} + \cancel{d_{v_{k-1}}} + w(v_{k-1} \to v_k) - d_k$$

$$= d_{v_0} + w(v_0 \to v_1) + w(v_1 \to v_2) + \cdots + w(v_{k-1} \to v_k) - d_k$$

$$= d_p + \text{weight in original} - d_q \quad \blacksquare$$



0. The graph where we want all-to-all minweights
Denote the edge weights by $w(u \to v)$

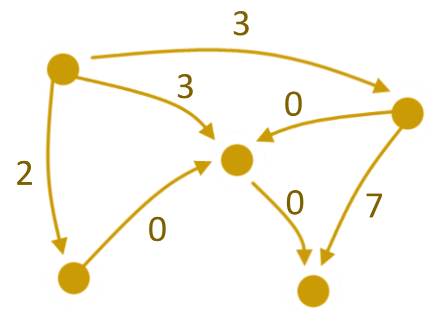


1. The augmented graph
Add a new vertex s, and run Bellman-Ford to compute minimum weights from s,

$$d_v = \text{minweight}(s \text{ to } v)$$

Cost to setup augmented graph: $O(V)$

Cost of Bellman-Ford: $O(VE)$



2. The helper graph
Define a new graph with modified edge weights

$$w'(u \to v) = d_u + w(u \to v) - d_v$$

Cost $O(E)$

3. Run Dijkstra to get all-to-all distances in the helper graph, $\text{distance}'(u \text{ to } v)$

cost $V \times O(E + V \log V)$

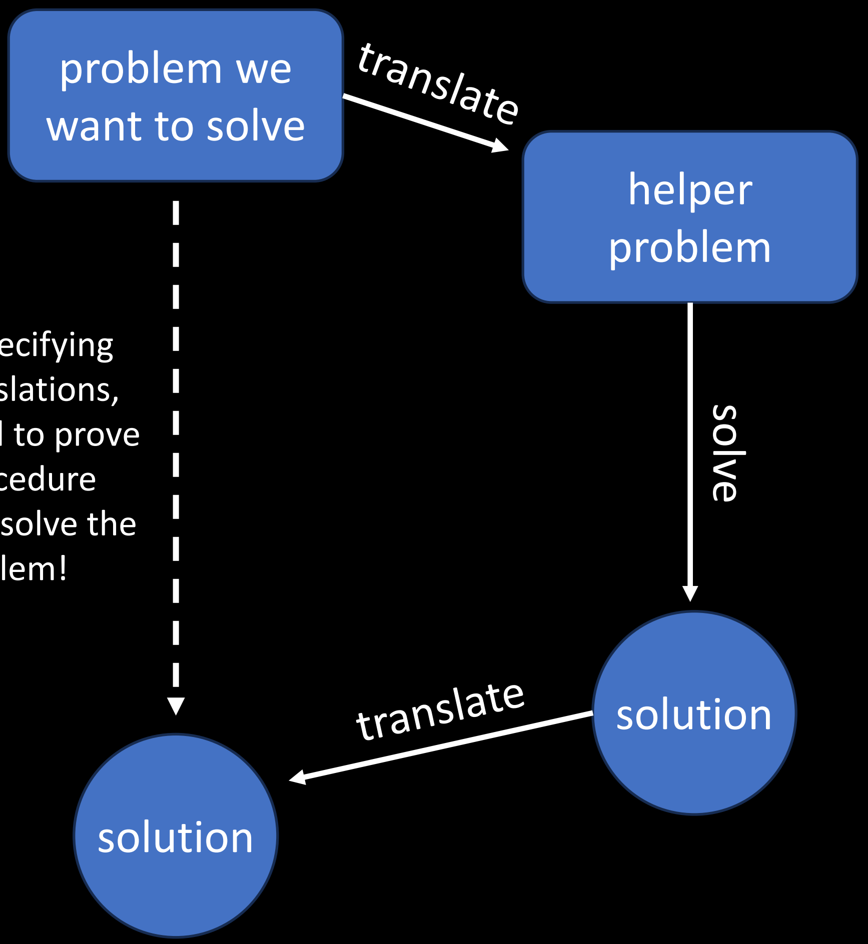
4. Translation Cost $O(V^2)$

$$\text{minweight}(p \text{ to } q) = \text{distance}'(p \text{ to } q) - d_p + d_q$$

This Dijkstra step dominates the costs.

The total cost is $O(VE + V^2 \log V)$, the same as V runs of Dijkstra.

Johnson’s algorithm is an example of the *translation strategy.*



As well as specifying the two translations, we also need to prove that this procedure does indeed solve the original problem!