Dijkstra's is pretty efficient and can find shortest paths for us, which is great, but it would be nice if we could have negative weight edges as well. Dijkstra's fails on negative weights because it can't "look ahead".

Example:

\[ d(A) = 0 \]
\[ d(C) = 5 \rightarrow \text{the next shortest path, and now } d(C) \text{ is fixed} \]
\[ d(B) = 6 \]

But \( d(C) \) should be 3!

The way we fix this is by giving up the ability to know when a vertex is done. Instead of "extending the shortest path" (a greedy choice) we take a step back and look at all paths of length 1, 2, 3, ..., \( N-1 \) (a dynamic programming approach).

Let \( D_i^s(u) \) be the shortest distance from \( s \) (the source vertex) to \( u \) for all \( u \), using up to \( i \) edges. We define the base case

\[ D_0^s(u) = \begin{cases} 0 & u = s \\ \infty & u \neq s \end{cases} \]

Then recursively

\[ D_{i+1}^s(v) = \min(D_i^s(u) + w(u,v)), \]

\[ \text{weight function} \]

Note that paths found for Dijkstra clearly use at most \( i+1 \) edges.
This example is nice and every change goes down and right.
The down is necessary, but the right is not. Moreover, if we changed $w(E, B)$ to $-4$, we would have a negative weight cycle. How do we detect this? Well, these cycles will always improve the shortest path, so if we find a better path with 1 NL edges, an impossibility without such cycles, then we have a negative weight cycle.

Before we turn this into code, notice that because each arrow points down, it doesn’t matter what order we process the edges in $E$! This makes our job easy.
BF (G, S, w)

Let \( D_0^i : V \rightarrow \mathbb{R} \) for each \( i \)
Initialize \( D_0^i (s) = 0 \) and \( D_i^u = \infty \) for \( u \neq s \).

For \( i = 1 \) to \( |V|-1 \)
  For each \( (u,v) \in E \)
    \( D_{i+1}^v (v) = \min \left( D_i^v (v), D_i^u (v) + w(u,v) \right) \)

For each \( (u,v) \in E \)
  If \( D_{|V|-1}^u (v) > D_{|V|-1}^v (u) + w(u,v) \)
    Throw an error

Return \( D_{|V|-1} \)

The runtime of BF is \( O(VE) \) regardless of input. This can be improved in the best case by observing that if \( D_{i+1}^u = D_i^u \) for some \( i \), no further improvements can be made. In this case, we can return early. This gives us a BC runtime of \( O(E) \).

Ex: Arbitrage

The practice of taking advantage of differing market prices to profit. For example, you might buy a stock on the NYSE for $20 but sell it for $20.01 on the LSE. You made 11 cents per share. Congrats!

Currency is a better place to practice arbitrage because there are so many, all of which can be immediately traded with each other at some ratio. Market inefficiencies may take time to propagate changes in valuations, leaving you able to profit. Consider

\[
\text{GBP} \xrightarrow{1.15} \text{USD} \xrightarrow{1.12} \text{EUR} \xrightarrow{1.11} \text{AUD} \xrightarrow{2.11} \text{GBP.}
\]

You can make money by cycling this trading sequence. This happens when trading ratios satisfy \( \frac{1}{i} > 1 \). How can we use BF to determine when this is possible? (assign \( w(u,v) = -\log \frac{u}{v} \))