The real Bellman-Ford algorithm
COMS20010 (Algorithms II)

John Lapinskas, University of Bristol

Bellman-Ford: A reminder

Algorithm: GoodPath

Input : A weighted digraph $G = ((V, E), w)$ with no negative-weight cycles, two vertices $s, t \in V(G)$, and an integer $k \geq 0$.

Output : A shortest walk from s to t in G with at most k edges, or `None` if none exists.

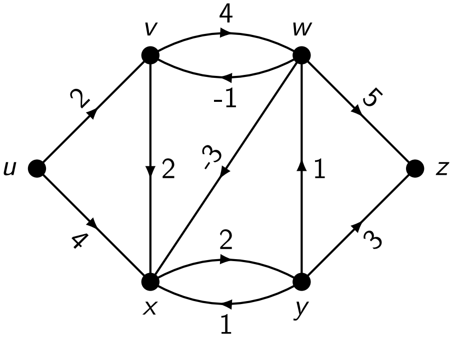
begin
    if $s = t$ **then**
        Return the empty walk.
    else if $k = 0$ **then**
        Return `None`.
    Write $N^+(s) = \{v_1, \dots, v_d\}$, where $d \geq 1$.
    Let $P_i \leftarrow \textsc{GoodPath}(G, v_i, t, k-1)$ for all $i \in [d]$.
    if $P_i =$ `None` *for all* $i \in [d]$ **then**
        Return `None`.
    Return whichever walk is shortest in $\{sv_iP_i : i \in [d], P_i \neq \texttt{None}\}$.

Memoised, this takes $O(|V|^3)$ time and space, since we need to store the result of $\Omega(|V|^2)$ function calls.

By making the algorithm iterative and being a little smarter, we can drop this to $O(|V||E|)$ time and $O(|V|)$ space. This is why it's often a good idea to de-memoise!

Iterative Bellman-Ford: An example

Say we are trying to find shortest paths from every vertex to z.

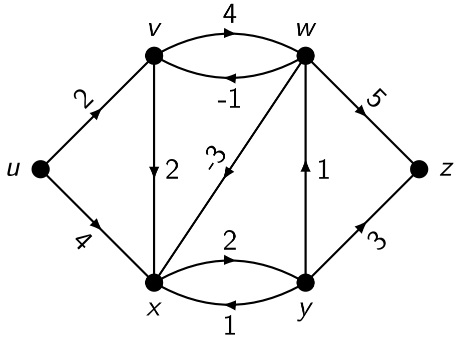


$k \backslash s$	u	v	w	x	y	z
5						
4						
3	$uxyz$	$vxyz$	$wxyz$	xyz	yz	z
2	$\emptyset$	vwz	wz	xyz	yz	z
1	$\emptyset$	$\emptyset$	wz	$\emptyset$	yz	z
0	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	z

This is getting ugly. These paths are taking up a lot of space, and we need to recalculate their lengths each time.

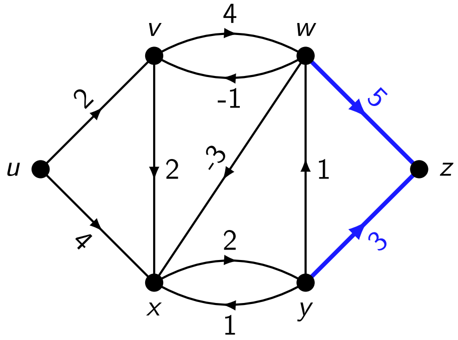
Why not just store the **first edge** of each path, along with its length?

Iterative Bellman-Ford: A better approach



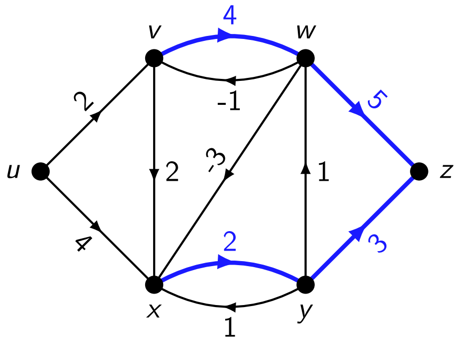
$k \backslash s$	u	v	w	x	y	z
5						
4						
3						
2						
1						
0	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(z, 0)$

Iterative Bellman-Ford: A better approach

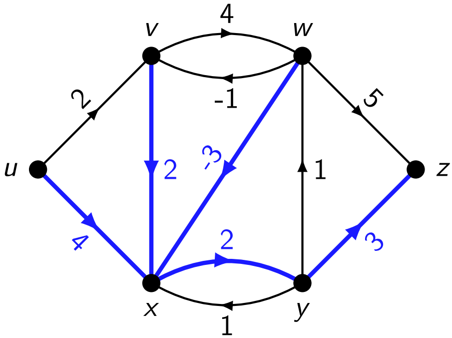


$k \backslash s$	u	v	w	x	y	z
5						
4						
3						
2						
1	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(wz, 5)$	$(\emptyset, \infty)$	$(yz, 3)$	$(z, 0)$
0	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(z, 0)$

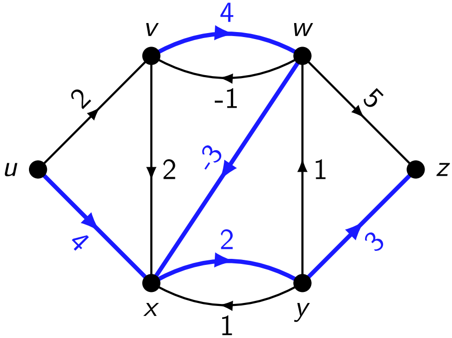
Iterative Bellman-Ford: A better approach



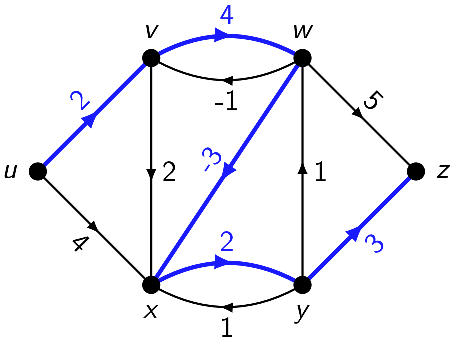
$k \backslash s$	u	v	w	x	y	z
5						
4						
3						
2	$(\emptyset, \infty)$	$(vw, 9)$	$(wz, 5)$	$(xy, 5)$	$(yz, 3)$	$(z, 0)$
1	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(wz, 5)$	$(\emptyset, \infty)$	$(yz, 3)$	$(z, 0)$
0	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(z, 0)$



$k \backslash s$	u	v	w	x	y	z
5						
4						
3	$(ux, 9)$	$(vx, 7)$	$(wx, 2)$	$(xy, 5)$	$(yz, 3)$	$(z, 0)$
2	$(\emptyset, \infty)$	$(vw, 9)$	$(wz, 5)$	$(xy, 5)$	$(yz, 3)$	$(z, 0)$
1	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(wz, 5)$	$(\emptyset, \infty)$	$(yz, 3)$	$(z, 0)$
0	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(z, 0)$



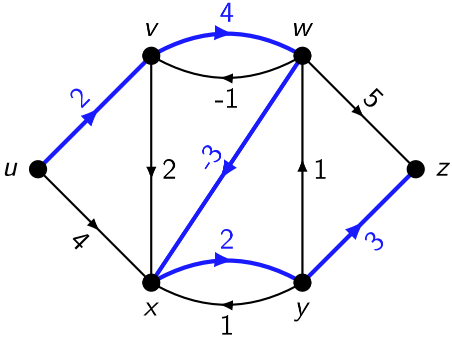
k \ s	u	v	w	x	y	z
5						
4	$(ux, 9)$	$(vw, 6)$	$(wx, 2)$	$(xy, 5)$	$(yz, 3)$	$(z, 0)$
3	$(ux, 9)$	$(vx, 7)$	$(wx, 2)$	$(xy, 5)$	$(yz, 3)$	$(z, 0)$
2	$(\emptyset, \infty)$	$(vw, 9)$	$(wz, 5)$	$(xy, 5)$	$(yz, 3)$	$(z, 0)$
1	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(wz, 5)$	$(\emptyset, \infty)$	$(yz, 3)$	$(z, 0)$
0	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(z, 0)$



$k \backslash s$	u	v	w	x	y	z
5	$(uv, 8)$	$(vw, 6)$	$(wx, 2)$	$(xy, 5)$	$(yz, 3)$	$(z, 0)$
4	$(ux, 9)$	$(vw, 6)$	$(wx, 2)$	$(xy, 5)$	$(yz, 3)$	$(z, 0)$
3	$(ux, 9)$	$(vx, 7)$	$(wx, 2)$	$(xy, 5)$	$(yz, 3)$	$(z, 0)$
2	$(\emptyset, \infty)$	$(vw, 9)$	$(wz, 5)$	$(xy, 5)$	$(yz, 3)$	$(z, 0)$
1	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(wz, 5)$	$(\emptyset, \infty)$	$(yz, 3)$	$(z, 0)$
0	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(z, 0)$

So e.g. $d(u, z) = 8$, via the path $uvwxyz$.

Iterative Bellman-Ford: A better approach



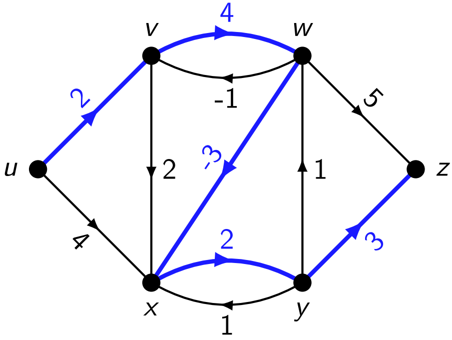
$k \backslash s$	u	v	w	x	y	z
5	$(uv, 8)$	$(vw, 6)$	$(wx, 2)$	$(xy, 5)$	$(yz, 3)$	$(z, 0)$
4	$(ux, 9)$	$(vw, 6)$	$(wx, 2)$	$(xy, 5)$	$(yz, 3)$	$(z, 0)$
3	$(ux, 9)$	$(vx, 7)$	$(wx, 2)$	$(xy, 5)$	$(yz, 3)$	$(z, 0)$
2	$(\emptyset, \infty)$	$(vw, 9)$	$(wz, 5)$	$(xy, 5)$	$(yz, 3)$	$(z, 0)$
1	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(wz, 5)$	$(\emptyset, \infty)$	$(yz, 3)$	$(z, 0)$
0	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(z, 0)$

So e.g. $d(u, z) = 8$, via the path *uvwxyz*.

We only actually use the $k = 0$ row of the table in calculating the $k = 1$ row, which we only use in calculating the $k = 2$ row...

Let's free the memory after we're done with it. Now we use $O(|V|)$ space!

Iterative Bellman-Ford: A better approach



$k \backslash s$	u	v	w	x	y	z
5	$(uv, 8)$	$(vw, 6)$	$(wx, 2)$	$(xy, 5)$	$(yz, 3)$	$(z, 0)$
4	$(ux, 9)$	$(vw, 6)$	$(wx, 2)$	$(xy, 5)$	$(yz, 3)$	$(z, 0)$
3	$(ux, 9)$	$(vx, 7)$	$(wx, 2)$	$(xy, 5)$	$(yz, 3)$	$(z, 0)$
2	$(\emptyset, \infty)$	$(vw, 9)$	$(wz, 5)$	$(xy, 5)$	$(yz, 3)$	$(z, 0)$
1	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(wz, 5)$	$(\emptyset, \infty)$	$(yz, 3)$	$(z, 0)$
0	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(\emptyset, \infty)$	$(z, 0)$

So e.g. $d(u, z) = 8$, via the path $uvwxyz$.

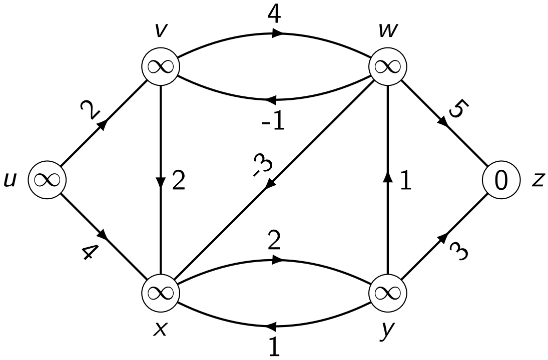
We can be even more cunning, and store **only the current row**.

So when we try and retrieve the value from our table for $s = v$, $k = 3$ (say), we might get the value for $s = v$, $k = 4$ instead if we already updated it. But this is OK — in fact, it means we sometimes find shorter paths faster!

Iterative Bellman-Ford: An even better approach

Here's what this looks like in practice:

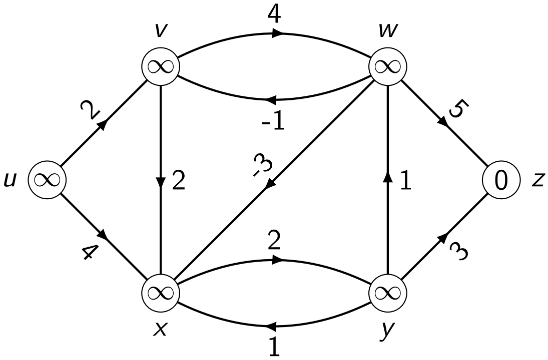
Initialise at $k = 0$.



Iterative Bellman-Ford: An even better approach

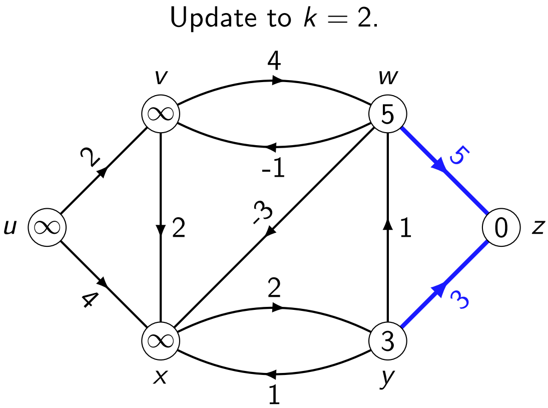
Here's what this looks like in practice:

Update to $k = 1$.



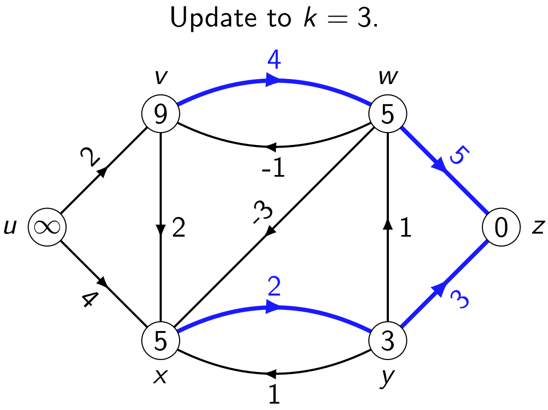
Iterative Bellman-Ford: An even better approach

Here's what this looks like in practice:



Iterative Bellman-Ford: An even better approach

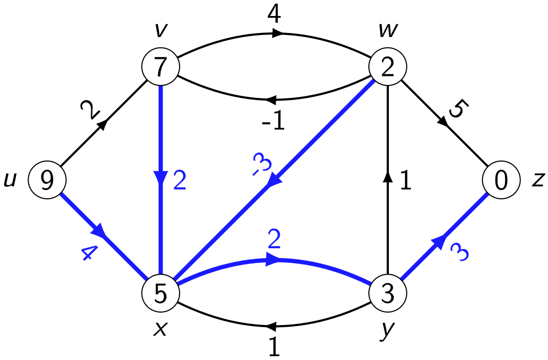
Here's what this looks like in practice:



Iterative Bellman-Ford: An even better approach

Here's what this looks like in practice:

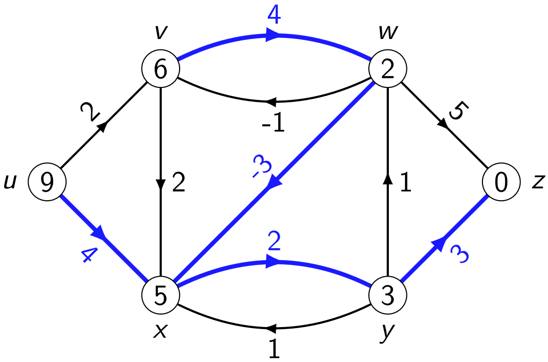
Update to $k = 4$.



Iterative Bellman-Ford: An even better approach

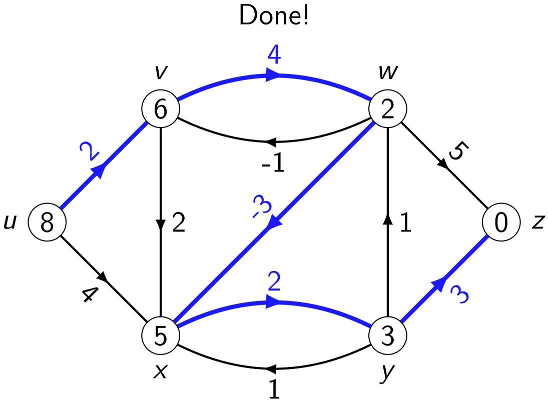
Here's what this looks like in practice:

Update to $k = 5$.



Iterative Bellman-Ford: An even better approach

Here's what this looks like in practice:



Now let's put this in pseudocode...

The real Bellman-Ford algorithm

Algorithm: BELLMANFORD

Input : A weighted digraph $G = ((V, E), w)$ with no negative-weight cycles and vertices $s, t \in V(G)$.

Output : A shortest path from s to t, or `None` if none exists.

begin
  Let $\texttt{dist}[v] \leftarrow \infty$ for all $v \in V \setminus t$, $\texttt{dist}[t] \leftarrow 0$.
  Let $\texttt{edge}[v] \leftarrow$ `None` for all $v \in V$.
  for $i = 1$ *to* $|V| - 1$ **do**
    for u *in* V **do**
      for v *in* $N^+(u)$ **do**
        if $\texttt{dist}[u] > w(u, v) + \texttt{dist}[v]$ **then**
          $\texttt{dist}[u] \leftarrow w(u, v) + \texttt{dist}[v]$ and $\texttt{edge}[u] \leftarrow (u, v)$.
  $v \leftarrow s$. **while** $v \neq t$ **do**
    If $\texttt{edge}[v] =$ `None`, return `None`.
    Else writing $\texttt{edge}[v] = (v, w)$, output (v, w) and set $v \leftarrow w$.

This now takes $O(|V| \sum_{u \in V} d^+(u)) = O(|V||E|)$ time, by the handshaking lemma, and $O(|V|)$ space. Using edge, you can also output every *other* shortest path to t in $O(|V|^2)$ time.

Other useful pathfinding algorithms

Bellman-Ford as described gives you all shortest paths to a sink.

You can also adapt it to give you all shortest paths from a source, like Dijkstra. (See problem sheet.)

What if you want shortest paths from **all** sources to **all** sinks, though? Repeatedly applying Dijkstra gives you $O(|V||E|\log|V|)$ time for non-negative edge weights.

You can match this running time even with negative edge weights using **Johnson's algorithm**.

Also, a lot of the time you're not working blind — you have some idea of "which direction is best", e.g. if you're pathfinding in a video game. In this case you should use a heuristic-guided algorithm like **A* search**, which often runs much faster than Dijkstra or Bellman-Ford.