Shortest Paths: Bellman-Ford Algorithm

Here’s a version of the Bellman-Ford algorithm that runs in time $\Theta(nm)$:

1. $\text{BellmanFord}(G=(V,E), s \in V)$
2. $W[n][n]$: new array
3. set $W[0][s]=0$, set $W[0][j]=\infty$ for $j$ not $s$
4. for $i = 1$ to $n$
5.   for each $j$ in $V$
6.     set $W[i][j] = W[i-1][j]$
7.   for each edge $(u,j)$ in $V$
8.     newPath = $c(u,j) + W[i-1][u]$
9.     if newPath < $W[i][j]$
10.    set $W[i][j] = (\text{newPath}, u)$
11. return $W[n-1]$

In both this and our original version of the algorithm, we consider all possible ways to reach node $j$ in at most $i$ steps, and record the best option.

In our original version, we accomplish this by going through all nodes $j$ one at a time, considering all of the edges into node $j$ at the same time (in the min, or the innermost for loop in the original version).

In this updated version, we accomplish this by initially recording our solution for $i - 1$ edges as the best solution (so far) for $i$ edges. We then go through all edges in the graph one at a time, where if edge $(u,j)$ yields a better path than the best so far we’ve recorded on iteration $i$ for node $j$, we update. The end result is the same: by the end of iteration $i$, we’ve considered, for each node $j$, each edge that leads into $j$. We’re just doing this in a different order that lets us take better advantage of the structure of an adjacency list.