Recitation 12

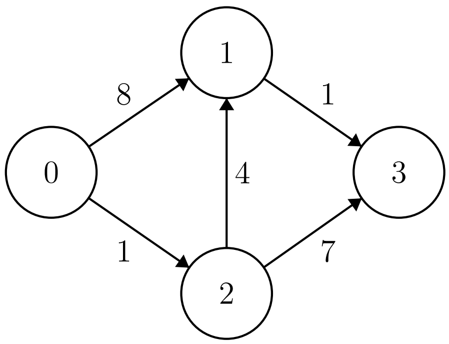
Dynamic Programming

12.1 Announcements

- *Midterm 2* is on **Friday**.

12.2 Can We Solve SSSP With Dynamic Programming?

Task 12.1. *Let $\delta(v, k)$ be the shortest path distance between the source and v using at most k edges. For the example graph shown, fill in the table below with values $\delta(v, k)$ using 0 as the source. If a vertex v is unreachable from the source using at most k edges, then write $\delta(v, k) = \infty$.*



k \ v	0	1	2	3
0	0	∞	∞	∞
1	0	8	1	∞
2	0	5	1	8
3	0	5	1	6

Task 12.2. *What are the values $\delta(v, 0)$ for every v?*

If v is the source, then 0. Otherwise, ∞, because we aren't allowed to use any edges!

Task 12.3. *Write $\delta(v, k)$ in terms of $\delta(\cdot, k-1)$. (Intuition: if we know shortest path distances using at most $k-1$ edges, can we easily calculate all shortest path distances which use at most k edges? We only need to extend each shortest path by one edge.)*

We're given some specific v and k. Consider all of v's *in-neighbors*, that is, vertices u for which there is an arc (u, v). For each one of these, we know $\delta(u, k-1)$. Therefore the values $\delta(u, k-1) + w(u, v)$ are possible distances to v which use at most k edges. The best of these is simply the minimum. Let's call this $b(v, k)$, for the "best incoming":

$$b(v, k) = \min_{u:(u,v)\in E} \big\{\delta(u, k-1) + w(u, v)\big\}.$$

Now, is it correct to say that $\delta(v, k) = b(v, k)$? What if $\delta(v, k-1)$ is smaller? This value is the length of a path of at most $k-1$ edges, so we may need to reuse it when considering paths of at most k edges. This yields the following definition for $\delta(v, k)$.

$$\begin{aligned}\delta(v, k) &= \min\big(\delta(v, k-1), b(v, k)\big)\\ &= \min\left(\delta(v, k-1), \min_{u:(u,v)\in E} \big\{\delta(u, k-1) + w(u, v)\big\}\right).\end{aligned}$$

Task 12.4. *Assuming the graph contains no negative cycles, prove the following statement:*

> *For each vertex, there exists a shortest path to it from the source using at most $|V| - 1$ edges.*

What does this statement tell us about using $\delta(v, k)$ to solve the SSSP problem?

Consider a shortest path to some vertex v. If this path contains at least $|V|$ edges, then one vertex on the path must be repeated (pigeonhole), and therefore the path contains a cycle. The total weight of this cycle cannot be strictly positive, since then we could remove the cycle to obtain a shorter path. Thus this cycle must have a total weight of 0. We can remove the cycle to obtain a path of fewer edges, but same total weight. Repeat as necessary to obtain a shortest path to each vertex which uses at most $|V| - 1$ edges.

Because of this, we can solve SSSP by calculating $\delta\big(v, |V| - 1\big)$ for each vertex v.

Task 12.5. *Using what's been established above, write a top-down dynamic programming solution to the SSSP problem. Note that your result will essentially be a top-down version of the well-known* Bellman-Ford *algorithm.*

Algorithm 12.6. *Top-down Bellman-Ford*

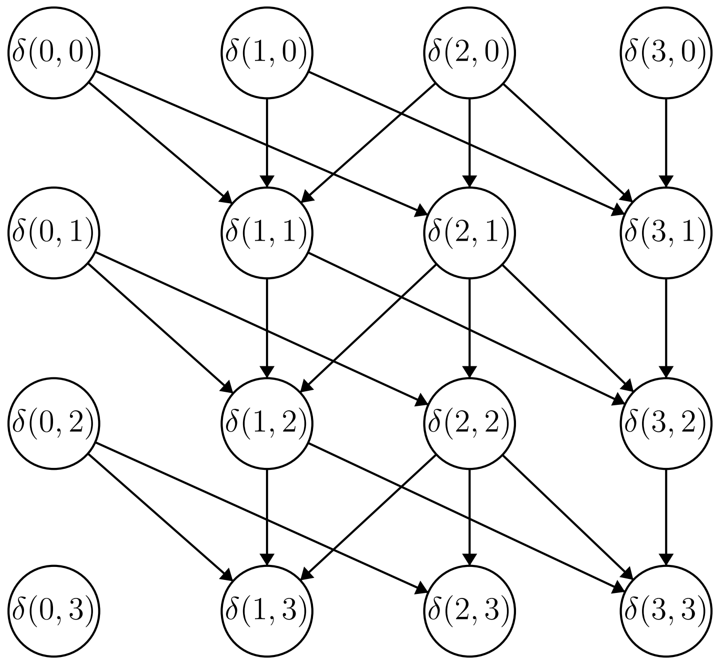
fun `TopDownBellmanFord` $(G = (V, E), s)$ $=$
  let
    % Subproblem: the shortest distance from s to v using k or fewer hops
    fun $\delta(v, k)$ $=$
      if $v = s$ **then** 0
      else if $k = 0$ **then** ∞
      else $\min\left(\delta(v, k-1), \min_{u:(u,v)\in E}\left\{\delta(u, k-1) + w(u, v)\right\}\right)$
  in
    $\{v \mapsto \delta(v, |V| - 1) : v \in V\}$
  end

12.2.1 Cost Analysis

Task 12.7. *Describe the DAG of dependencies between subproblems $\delta(v, k)$; i.e., each vertex is a pair (v, k), and there is an arc from (v', k') to (v, k) if we need to know the value $\delta(v', k')$ in order to calculate $\delta(v, k)$.*

Draw the dependency DAG for the example graph given above.

Except for $v = s$, each $\delta(v, k)$ is dependent upon $\delta(v, k-1)$ as well as $\delta(u, k-1)$ for every u which is an in-neighbor of v.



Task 12.8. *Identify a bottom-up ordering of the Bellman-Ford DAG which maximizes parallelism while respecting subproblem dependencies. Use this bottom-up ordering to determine the work and span of Bellman-Ford in terms of n, the number of vertices, and m, the number of edges. You may assume constant-time access to subproblems you have already computed.*

We can solve the subproblems in increasing order of k. For each value of k, we can solve a whole "row" in parallel: round i of the algorithm computes $\{v \mapsto \delta(v, i) : v \in V\}$ in parallel.

What are the work and span of calculating a single $\delta(v, k)$? Notice that we have to take the minimum over items in $\{u | (u, v) \in E\}$. Let's write $\deg^-(v)$ for the size of this set; i.e., the in-degree of v. Then calculating $\delta(v, k)$ has $O(\deg^-(v))$ work and $O(\log(\deg^-(v)))$ span.

Each round is parallel, hence the span of any particular round is

$$O\left(\max_v \log(\deg^-(v))\right) = O(\log n).$$

There are n rounds, and so the span is $O(n \log n)$.

The work of any particular round is $O(m)$: for each vertex we pay proportional to the indegree of that vertex, and the sum of indegrees is equal to m.

$$O\left(\sum_v \deg^-(v)\right) = O(m).$$

Once again, there are n rounds, so the total work is $O(nm)$.

12.2.2 Is Bellman-Ford A Useful Algorithm?

Task 12.9. *Recall that the work bound of Dijkstra's algorithm as presented in class was $O(m \log n)$. For certain priority queue implementations, this can be reduced even to $O(m + n \log n)$. Bellman-Ford is clearly much slower. In what situations might you want to use Bellman-Ford instead of Dijkstra's algorithm?*

Dijkstra's algorithm may fail on graphs which contain negative edges. As presented above, Bellman-Ford works for any graph–even those with negative edges–which does not contain a negative cycle (which is fine, because SSSP isn't defined for negative cycles).

Task 12.10. *Describe a simple modification to Bellman-Ford's algorithm which can be used to detect the presence of a negative cycle.*

Notice that, for every vertex v on the negative cycle, even as k increases past n, $\delta(v, k)$ will continue to decrease. Thus, if there exists a v such that $\delta(v, n) < \delta(v, n - 1)$, then the graph contains a negative cycle.

Task 12.11. *Describe an optimization for Bellman-Ford which (assuming there are no negative cycles) causes it terminate after ℓ rounds, where ℓ is the maximum number of edges used in any of the shortest paths.*

Notice that for every v, $\delta(v, \ell + 1) = \delta(v, \ell)$. Thus we can terminate as soon as the δ's stop changing.

.