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 $\delta(v, k) = \infty$.

![Graph Image]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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.)

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?

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.

Built: April 4, 2016
12.2. CAN WE SOLVE SSSP WITH DYNAMIC PROGRAMMING?

12.2.1 Cost Analysis

**Task 12.6.** 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.

**Task 12.7.** 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.

12.2.2 Is Bellman-Ford A Useful Algorithm?

**Task 12.8.** 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?

**Task 12.9.** Describe a simple modification to Bellman-Ford’s algorithm which can be used to detect the presence of a negative cycle.

**Task 12.10.** Describe an optimization for Bellman-Ford which (assuming there are no negative cycles) causes it terminate after $\ell$ rounds, where $\ell$ is the maximum number of edges used in any of the shortest paths.