Single Source Shortest Path Algorithms

Definitions

Definition 1 (Greedy algorithm). A greedy algorithm is one which always makes the choice that looks best at the moment—the locally optimal choice—in order to find the best globally optimal solution. Greedy algorithms do not always yield optimal solutions, but for many problems they do.

Definition 2 (Shortest path). A shortest path from vertex $s$ to vertex $t$ is a directed path from $s$ to $t$ with the property that no other such path has a lower total edge weight.

Definition 3 (Negative Weight Cycle). A negative weight cycle is a cycle with weights that sum to a negative number.

Dijkstra’s Algorithm

Dijkstra’s algorithm finds the shortest path between two given vertices in a weighted graph, assuming that the graph’s edge weights are non-negative. The running time of the algorithm is $O(E \log V + V \log V)$ when the graph is implemented using adjacency lists. The pseudo-code for the algorithm is given below.

Pseudocode

Dijkstra($G, s$)

1. for each vertex $v \in V_G$
2. $dist[v] = \infty$
3. $parent[v] = \text{NIL}$
4. $dist[s] = 0$
5. $Q = V_G$
6. while $Q \neq \emptyset$
7. $u = \text{Extract-Min}(Q)$
8. for each vertex $v \in G. Adj[u]$
9. if $dist[v] > dist[u] + w(u, v)$
10. $dist[v] = dist[u] + w(u, v)$
11. $parent[v] = u$

Runtime

The running time of Dijkstra’s algorithm has two components, $E \log V$ and $V \log V$. Let us first consider the $V \log V$ term: this component derives from the maximum size ($V$) of the heap used to store vertices, and the running time of heap operations such as INSERT and REMOVE_MIN is $O(\log V)$.

The $E \log V$ term has to do with the relaxation step of Dijkstra’s algorithm. Each edge examined may result in a relaxation of the neighboring node in the heap; in other words, an update key operation that is $O(\log V)$. We know that the number of vertices examined in line 8 above is bounded by the total degree of all vertices, as each vertex is added and popped exactly once from the min-heap. This value is $2|E|$ by the Handshake lemma, so in the worst case we have $2|E|$ decrease-key operations, for a total of $O(E \log V)$.

This bound is good for easily proving our run-time, but it is not tight. Each edge $(u, v)$ can only cause one relaxation, not two as the handshake lemma suggests. This is because $(u, v)$ is explored only when node
$u$ is popped from the min-heap. This means that when $(u, v)$ is explored from node $v$ node $u$ has already been removed, so it’s key cannot be decreased.

Example

Trace through Dijkstra’s on this graph, starting at A.

Dijkstra’s algorithm produces the following state:

<table>
<thead>
<tr>
<th>Node</th>
<th>Distance from A</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>0</td>
</tr>
<tr>
<td>$B$</td>
<td>2</td>
</tr>
<tr>
<td>$C$</td>
<td>1</td>
</tr>
<tr>
<td>$D$</td>
<td>3</td>
</tr>
<tr>
<td>$E$</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Node</th>
<th>Parent node</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>NULL</td>
</tr>
<tr>
<td>$B$</td>
<td>$A$</td>
</tr>
<tr>
<td>$C$</td>
<td>$A$</td>
</tr>
<tr>
<td>$D$</td>
<td>$C$</td>
</tr>
<tr>
<td>$E$</td>
<td>$D$</td>
</tr>
</tbody>
</table>
Problems

Problem 1
Does Dijkstra’s Algorithm work with negative weights? Why or why not?

Problem 2
True or false: Dijkstra’s algorithm will not terminate if run on a graph with negative edge weights.

Problem 3
True or false: If we double the weights of all the edges in a graph, then Dijkstra’s algorithm will produce the same shortest path. What about squaring?

Problem 4
Explain why Dijkstra’s algorithm is a greedy algorithm.

Problem 5
Find the shortest path between vertices E and G.