Dijkstra's Shortest Path Algorithm

Write down your guesses.

Find shortest path costs from node C to all other nodes.
Repeat.

Cost?

Why?

Which node is second?

Which node is it's cost?

Which node should we start with?

Dijkstra's Algorithm
How can you succinctly represent all shortest paths?

<table>
<thead>
<tr>
<th>Node</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>6</td>
<td>4</td>
<td>0</td>
<td>7</td>
<td>5</td>
</tr>
</tbody>
</table>

What is the path to C?

Diagram Algorithm
What might the algorithm look like that efficiently calculates this result?

<table>
<thead>
<tr>
<th>Node</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>6</td>
</tr>
<tr>
<td>B</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>5</td>
</tr>
<tr>
<td>F</td>
<td>3</td>
</tr>
</tbody>
</table>

Diagram:

- Node A connected to nodes B and C with costs 6 and 4, respectively.
- Node B connected to node D with cost 2.
- Node C connected to node E with cost 5.
- Node E connected to node F with cost 3.

Dijkstra's Algorithm
Return dist, prev

\[ \text{prev}[u] = u \]
\[ \text{dist}[u] = \text{dist}[v] + \text{r}(u,r) \]

\[
\text{if } \text{dist}[v] < \text{dist}[u] + \text{r}(u,r) : \\
\text{for } u \in \text{E} : \\
\quad u = \text{pa}, \text{pop}() \\
\text{while } \text{pa} \text{ empty} : \\
\quad \text{pa} = \text{makequeue}(v, \text{dist}) \\
\quad \text{dist}[v] = 0 \\
\quad \text{prev}[v] = v \\
\quad \text{dist}[u] = \infty \\
\text{for } u \in \text{E} : \\
\quad \text{# } G = (V, E) \text{, } \text{r}(u,v) = \text{cost of } (u,v) \text{. } S = \text{start node} \\
\text{function dikstra(}G, s) :}
\]

Dijkstra's Algorithm
Dijkstra's Algorithm

Is it correct?

Divide nodes into 2 groups:
1. Already processed, best path and cost found.
2. Best path so far found, but not known to be best overall.

Invariant:
At the top of the while loop, PQ holds group 2, group 1 has been popped.

Claims:
- After a node is removed from PQ, it doesn't need to be updated, because all paths that are shorter have already been extended.
- The algorithm won't update it.
- Any nodes in group 2 that touch the removed node will be updated if the node is the predecessor.
Go back to algorithm and examine the runtime.

Run time:

Big O notation: Algorithm
Dijkstra's Algorithm

Runtime

Initialization loop $O(1\text{V})$

Make priority queue $O(\text{make queue})$

While loop $O(1\text{V})$ repetitions

$\Rightarrow$ Pop $O(1\text{V} \cdot \text{pop})$

Edge for loop $O(1\text{E})$ repetitions over algorithm

$\Rightarrow$ Change Key $O(1\text{E} \cdot \text{change key})$

Total Runtime = $O(1\text{V}) + O(\text{make queue}) + O(1\text{V} \cdot \text{pop}) + O(1\text{E} \cdot \text{change key})$

Using binary tree min heap implementation:

$\Rightarrow O((1\text{V} + 1\text{E}) \log 1\text{V})$
with the node that is being processed at that iteration. Label each column.

<table>
<thead>
<tr>
<th>Node</th>
<th>Processed</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>3</td>
</tr>
<tr>
<td>F</td>
<td>4</td>
</tr>
</tbody>
</table>

Problem 6. (6 points):

![Graph](image)