Skyline Graph

This note is about analyzing "Grid of tries" (Chapter 12.5.3 in the textbook) with skyline graph.

Definitions

**dst_len, src_len**: length of the destination prefix or the source prefix

**Skyline graph**: A graph with dst_len and src_len as vertical and horizontal axis. Given a pair of (dst ip, src ip), all and only the matching nodes (including those have exact-matching rules or not) will be on the graph. A node’s coordination is (src_len, dst_len). Every horizontal line in the skyline graph represents a src trie.

**Domination rule**: In a skyline graph, if a rule is not a prefix in both dst and src fields of any other rule, then the rule is a domination rule.

**Domination node**: In a skyline graph, if a node has the longest dst_len among all nodes with a particular src_len, then the node is a domination node.

Property of the domination node: A domination node cannot be a prefix in both dst and src fields of any other node in a skyline graph.

To prove it, you can assume a domination node is at (s1, d1), and there exists a node at (s2, d2), s2 > s1, d2 > d1.

If there is a node is at (s2, d2), there must be nodes at (0, d2), (1, d2), ... , (s1, d2), ... , (s2 - 1, d2), because a node must be on a src trie.

The node at (s1, d2) now should be the domination node, which is a contradiction.

**Skyline**: the algorithm’s search path in the src tries

Some properties of the skyline:

- The first node in the skyline always has src_len = 0. Because you always search from the root of a src trie.
- The first node in the skyline always has the longest possible dst_len. Because the algorithm first go to the deepest possible dst trie node.
- dst_len is non-increasing. Because:
If next node is on the same trie, dst_len remains the same.
If following the switch pointer, you will switch to a src trie corresponding to an ancestor dst trie node, so dst_len must decrease.
• src_len will increase by 1 every step of the skyline. Because to go along a switch pointer or to find the child both will go 1 level down the src trie.

Example

Using the example in the book, a skyline graph looks like this:

Claim 1

It is impossible that a skyline miss a domination node in a skyline graph.

The first node of the skyline must be a domination node, because the algorithm will start at the src trie with the longest dst_len.

Then, by definition, every step, the algorithm first attempts to search its child. If its child is a matching node, it must be a domination node, because it has the longest possible dst_len for a particular src_len. If its child is not a matching node, the algorithm will use a switch pointer to find the lowest ancestor, and the lowest ancestor has the longest dst_len for a
particular src_len, so it is also the domination node.

Claim 2

It is possible that a skyline miss a domination rule in a skyline graph.

Example

<table>
<thead>
<tr>
<th>Rules</th>
<th>Dst</th>
<th>Src</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>0*</td>
<td>0*</td>
</tr>
<tr>
<td>R2</td>
<td>00*</td>
<td>*</td>
</tr>
<tr>
<td>R3</td>
<td>*</td>
<td>00*</td>
</tr>
<tr>
<td>R4</td>
<td>00*</td>
<td>01*</td>
</tr>
</tbody>
</table>

In this rule set, priority is defined as R1 > R2 > R3 > R4.

The above rule set will produce a grid-trie (rules in "[]" are storedRules):

If dst = 00, src = 00, then the skyline graph will look like this:
As shown above, R1 cannot be reached, but R1 is a domination rule. If a storedRule is only stored in every exact-matching rule node, the algorithm will eventually output R2 instead of the correct answer R1.

If a storedRule is stored in every node, the algorithm is correct:
Claim 3

The algorithm can cover all the matching rules if a \textit{storedRule} is stored in every node.

Suppose an exact-matching rule node's src\_len is $s1$. The domination node with src\_len = $s1$ will consider this rule. Since a skyline does not miss any domination node, the algorithm can cover all the matching rules.