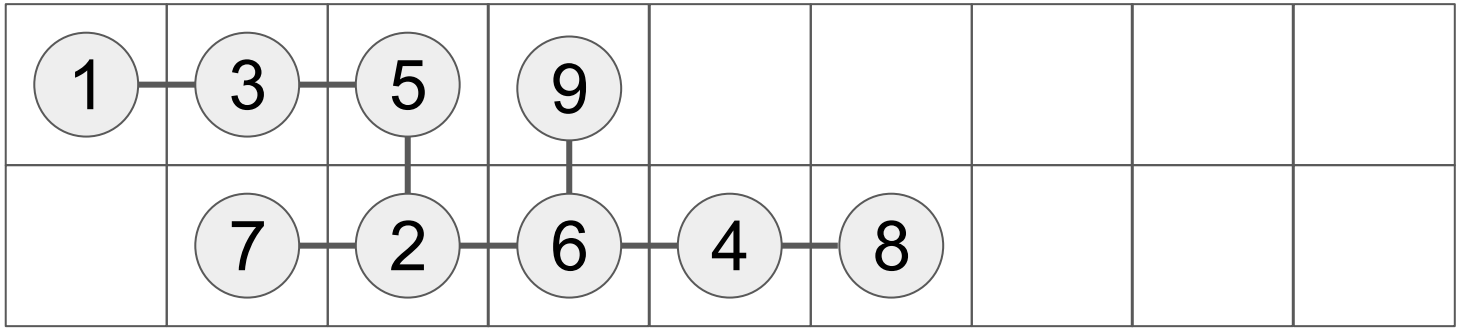
# **Problem E**
# Embedding Enumeration

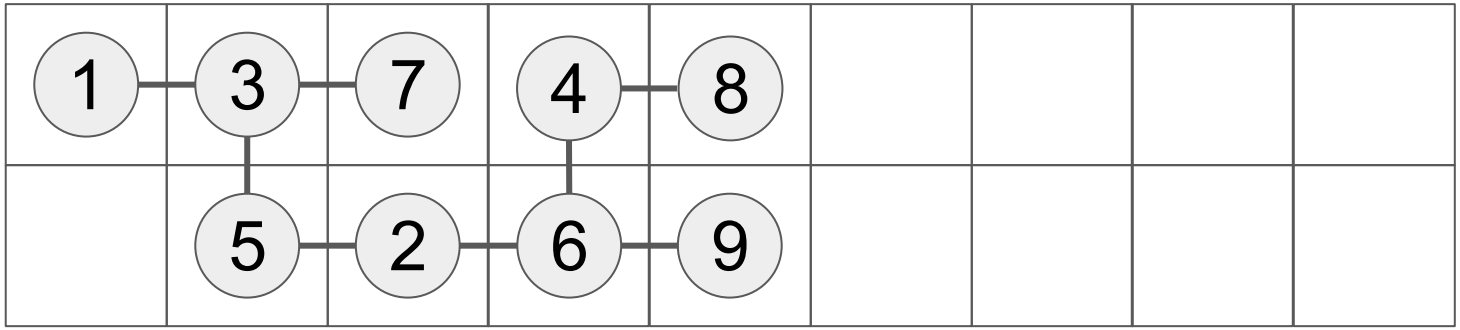
Submits: 1
Accepted: ?

Author: Luka Kalinovčić

Problem: Given a tree, count the number of ways to embed it in a 2 by N grid, such that two nodes connected by an edge are adjacent in the grid. Node 1 has to be in top-left cell.



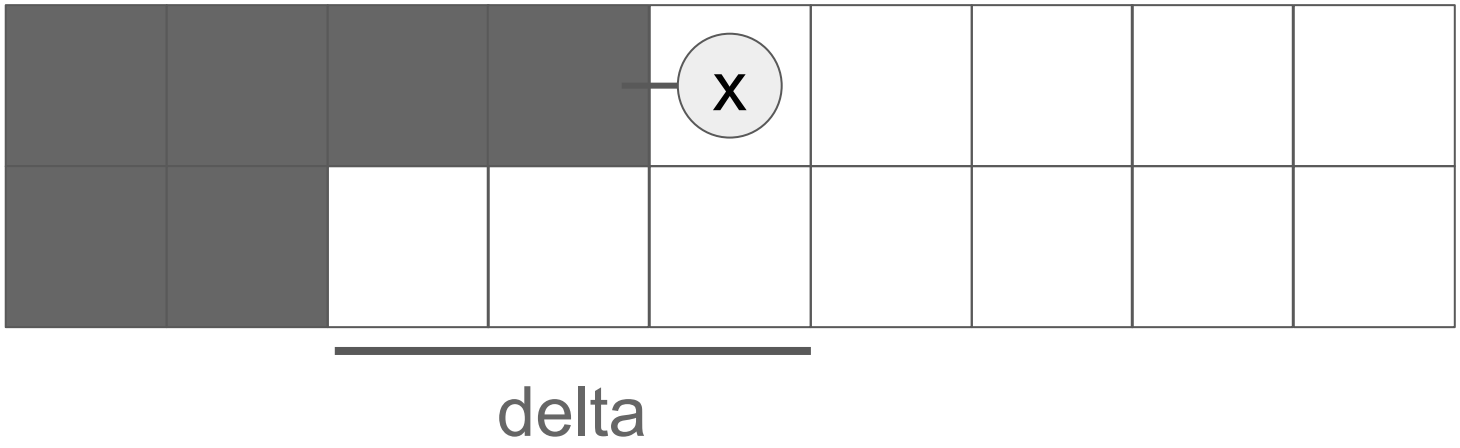
Problem: Given a tree, count the number of ways to embed it in a 2 by N grid, such that two nodes connected by an edge are adjacent in the grid. Node 1 has to be in top-left cell.




Observation: When we root the tree at node 1, it has to be a binary tree. Otherwise, we have a node with degree greater than three, which can't be embedded.

Let's build a dynamic programming solution that enumerates all embeddings. We can describe the state as (x, delta).

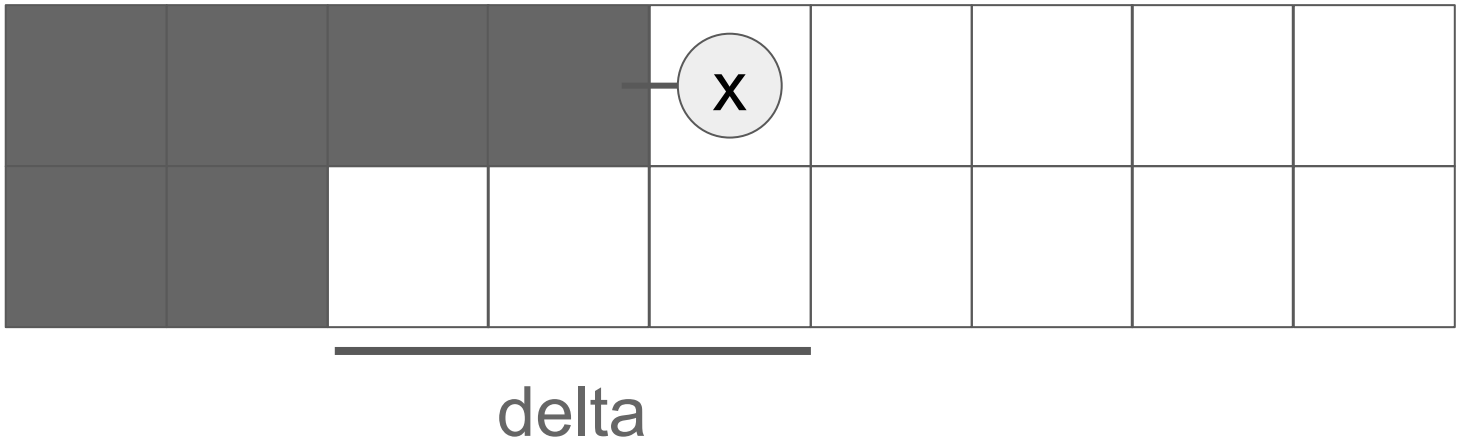
At this state, we have embedded all nodes except for those in x's subtree. Node x is embedded at the last cell of the longer of the two rows, and the delta is the difference in length between the two rows.



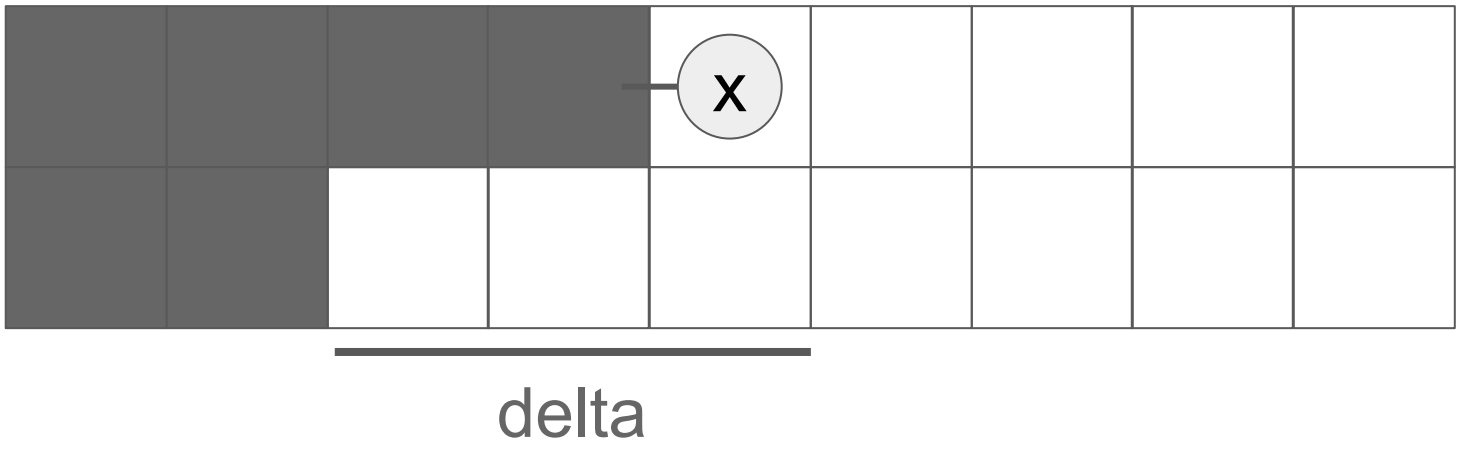
To make the transition, we'll try every possible assignment of node x's children to neighboring cells.

If assignment assigns a node y to a cell below x, we also try every possible assignment of y's children to neighboring cell.

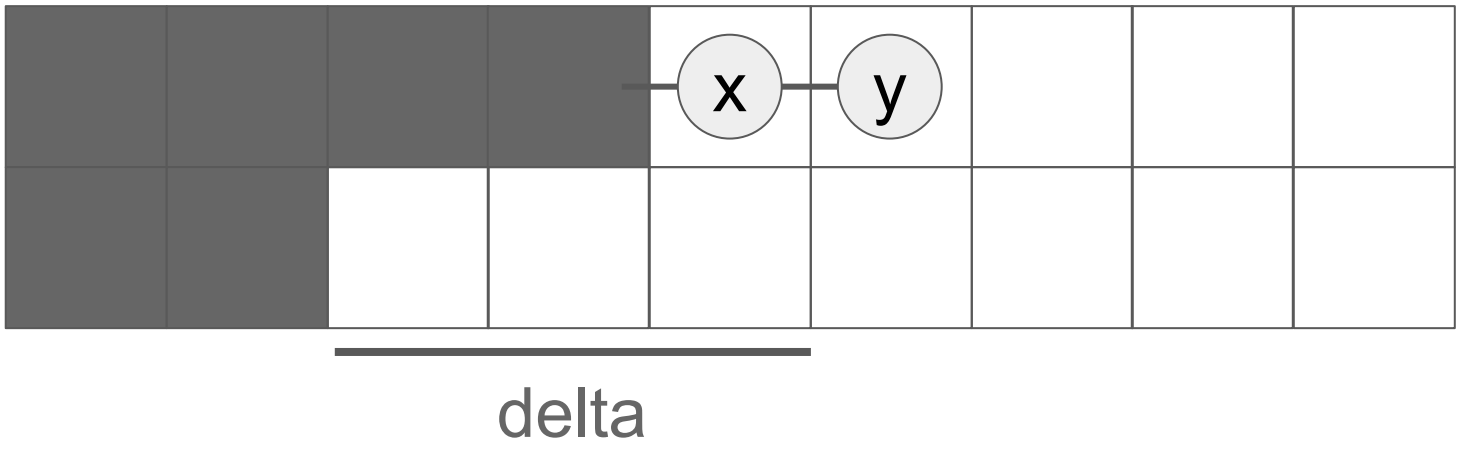
Let's analyze possible outcomes of such assignments.



Trivial case: x has no children. We've found one valid embedding.

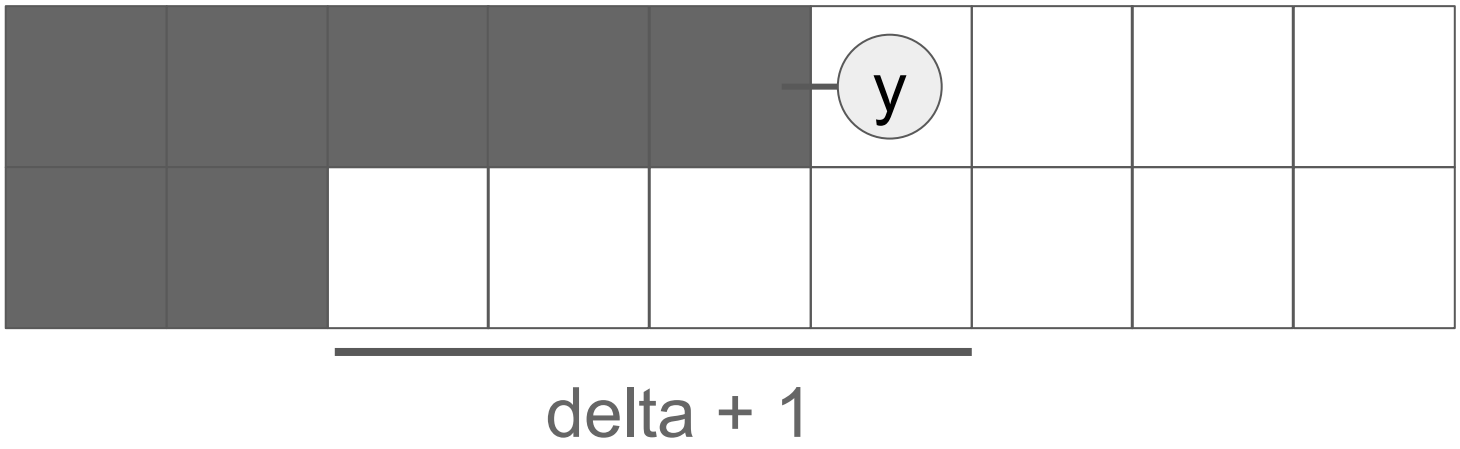


Node x has one child node y that was assigned to the right cell.

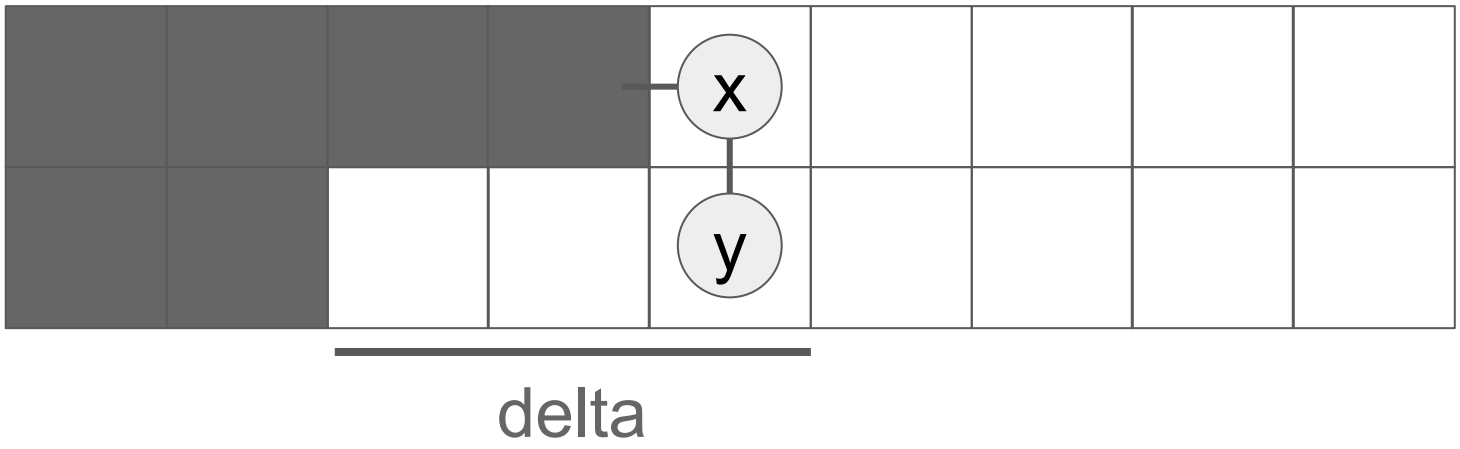


Node x has one child node y that was assigned to the right cell.

We transition to state (y, delta + 1)

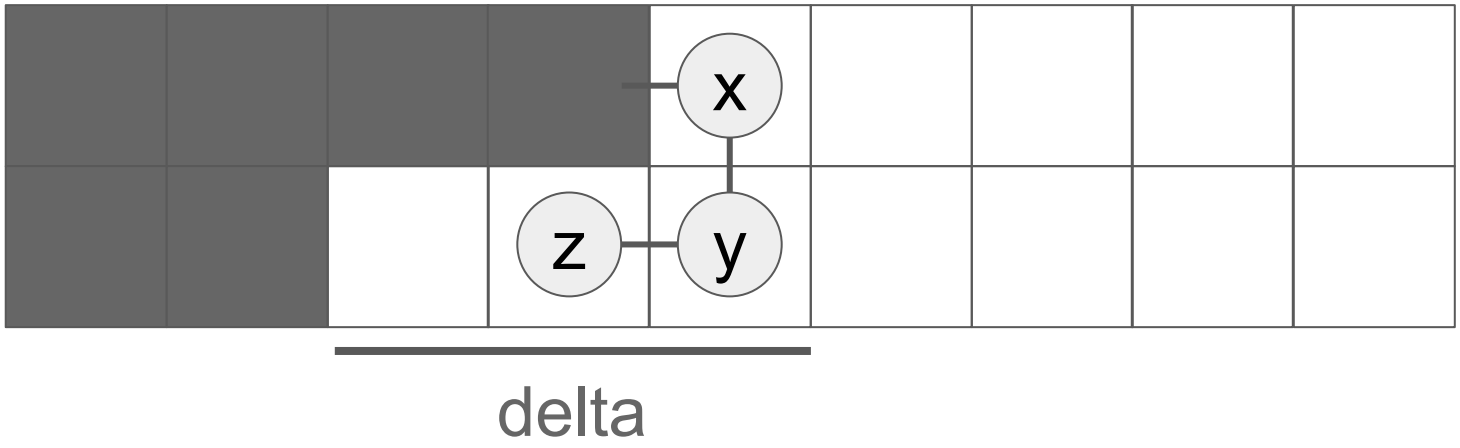


Node x has one child node y that was assigned to the bottom cell. We also assign y's children to neighboring cells.



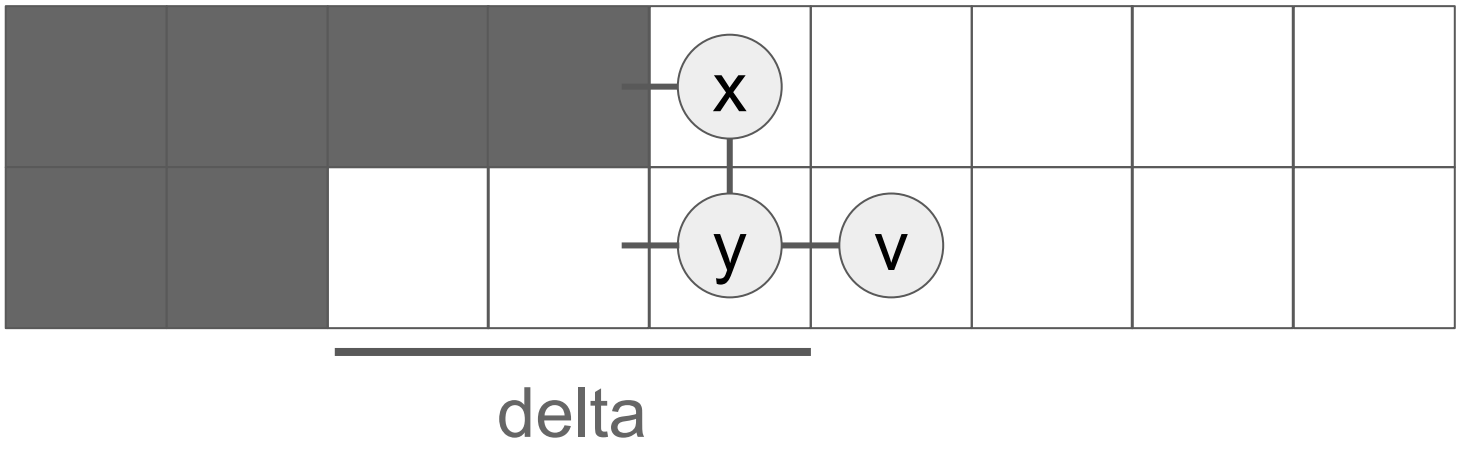
Node x has one child node y that was assigned to the bottom cell. We also assign y's children to neighboring cells.

If it there is a child node z assigned to the left, we know that its subtree has form a simple chain of length up to (delta - 1). Otherwise we can't make a valid embedding from this assignment.



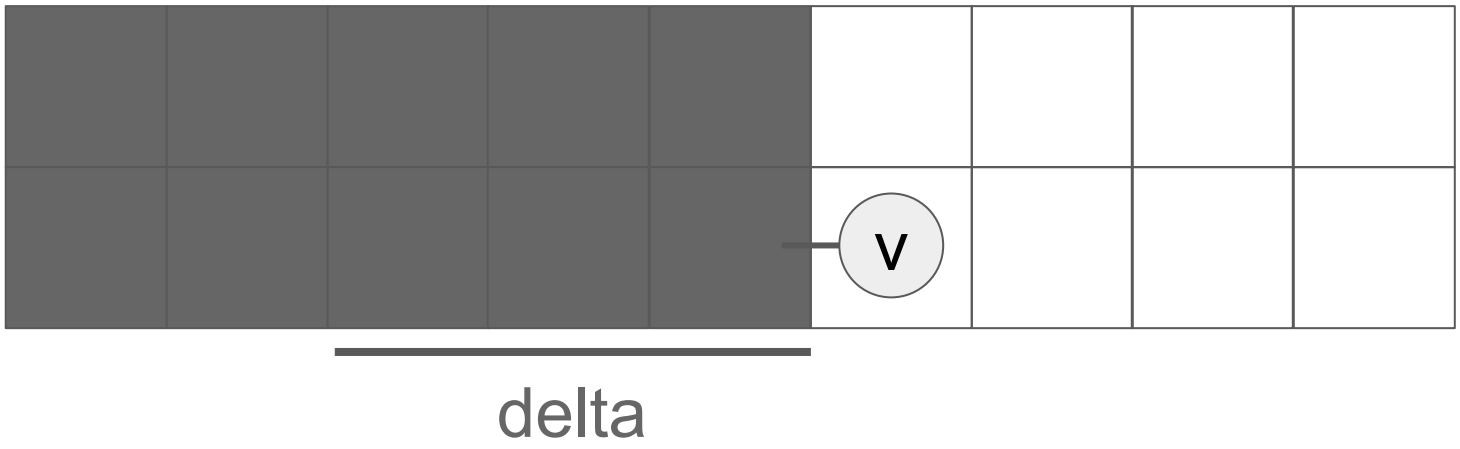
Node x has one child node y that was assigned to the bottom cell. We also assign y's children to neighboring cells.

If it there is a child node v assigned to the right, we transition to state (v, 1).



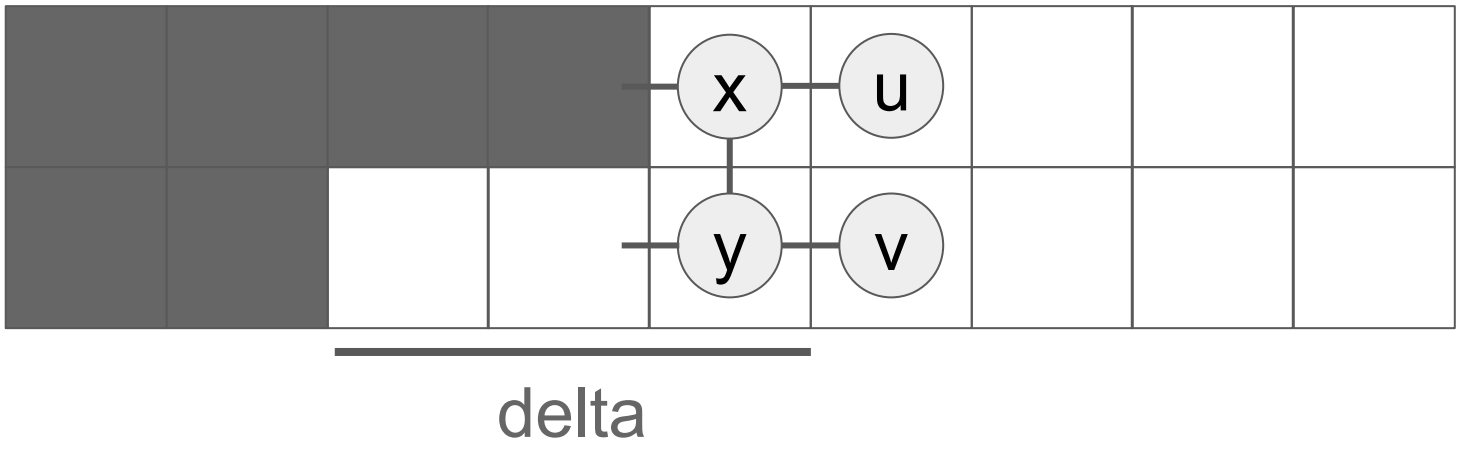
Node x has one child node y that was assigned to the bottom cell. We also assign y's children to neighboring cells.

If it there is a child node v assigned to the right, we transition to state (v, 1).

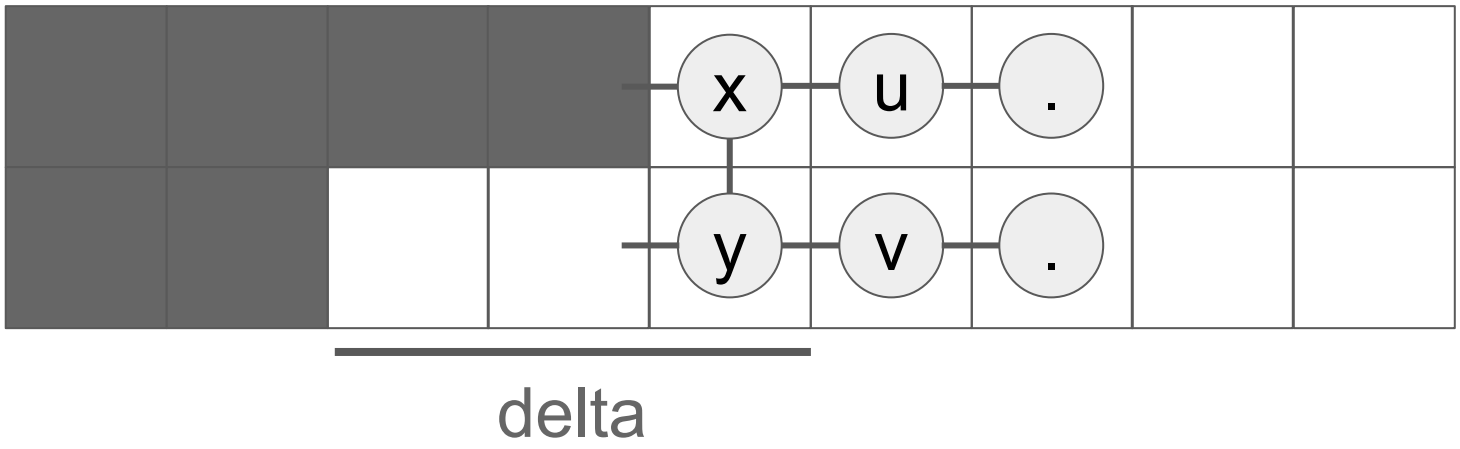


In the general case, x has two children y and u, and y has a child v assigned to the lower right cell.

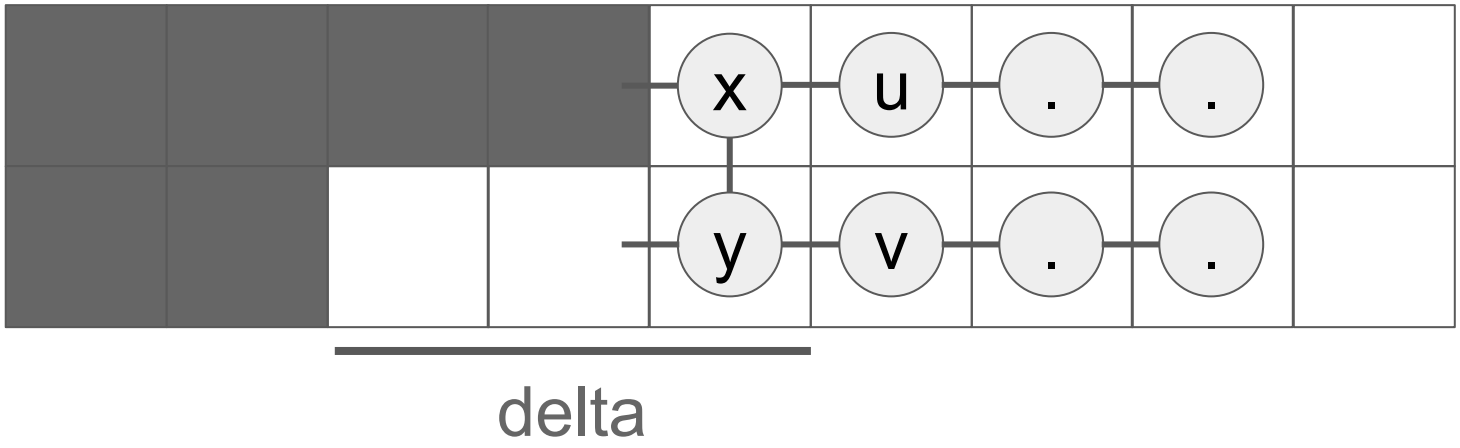
We now have two nodes, u and v, whose subtrees are not yet embedded, so we can't transition to any simple state just yet.



We keep appending children to the right until one of the chain runs out of nodes (or we encounter a node with two children which would make this assignment invalid).

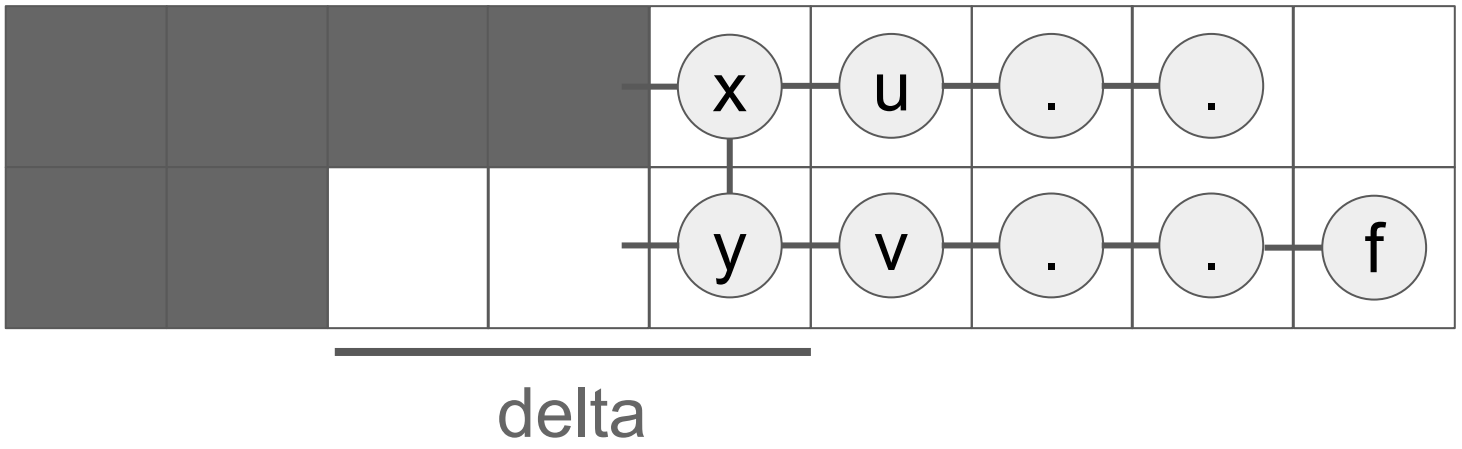


We keep appending children to the right until one of the chain runs out of nodes (or we encounter a node with two children which would make this assignment invalid).



We keep appending children to the right until one of the chain runs out of nodes (or we encounter a node with two children which would make this assignment invalid).

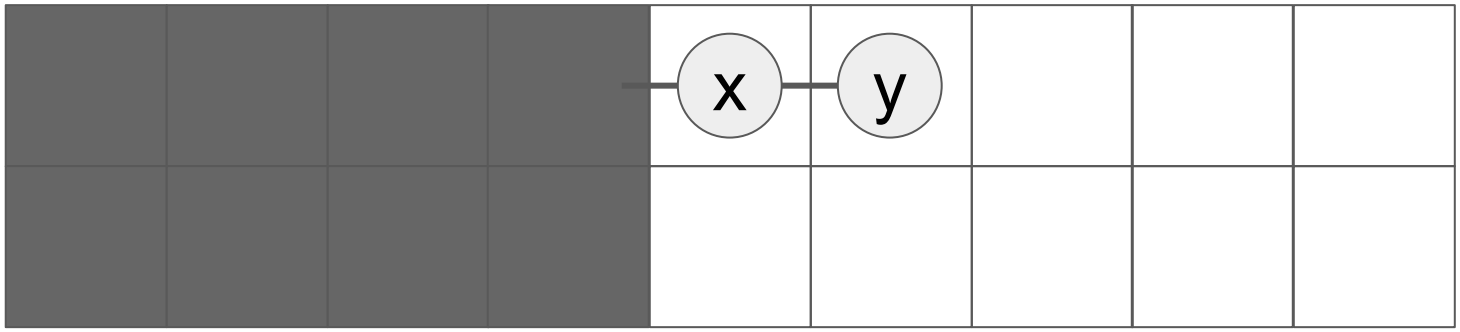
Once that happens, we can transition to state (f, 1).



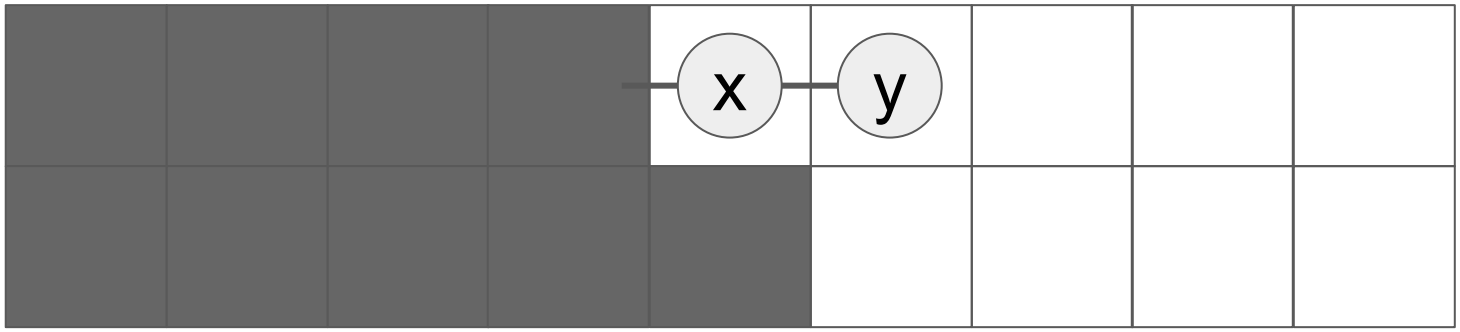
There are $O(N^2)$ states, and it's possible to implement all transitions in O(1) with some precomputation.

To speed it up, let's try to fix delta at 1, and see what breaks.

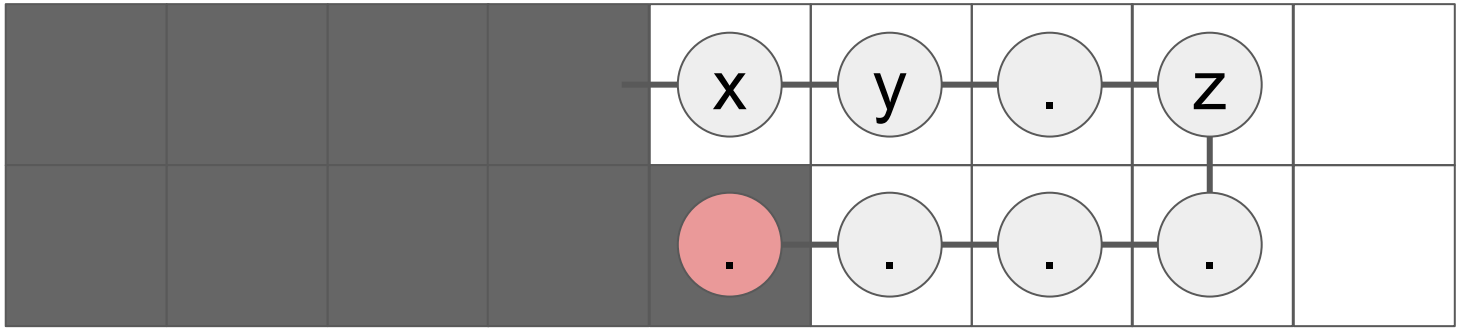
The only case where we actually increase the delta is the one where node x has one child assigned to the right cell.



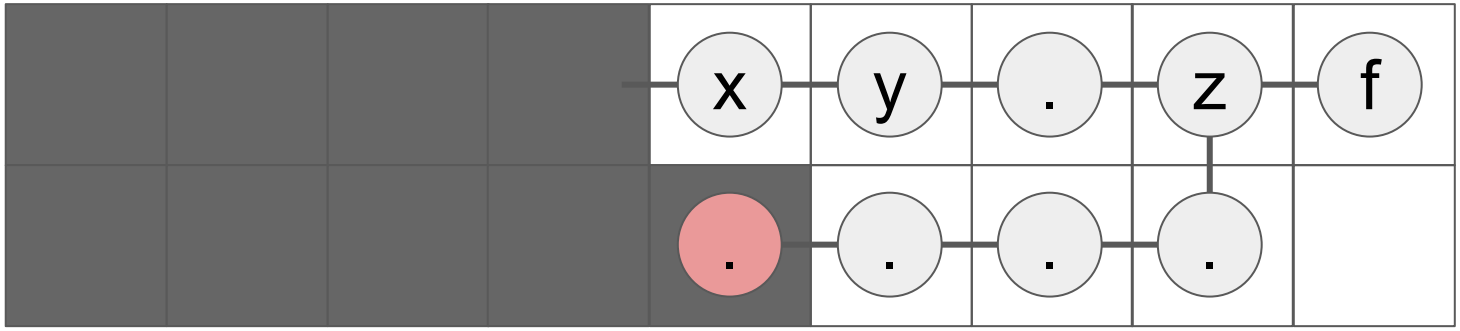
Originally we would transition to state (y, 2), but what kinds of embeddings would we miss if we transitioned to (y, 1) instead?



Originally we would transition to state (y, 2), but what kinds of embeddings would we miss if we transitioned to (y, 1) instead?

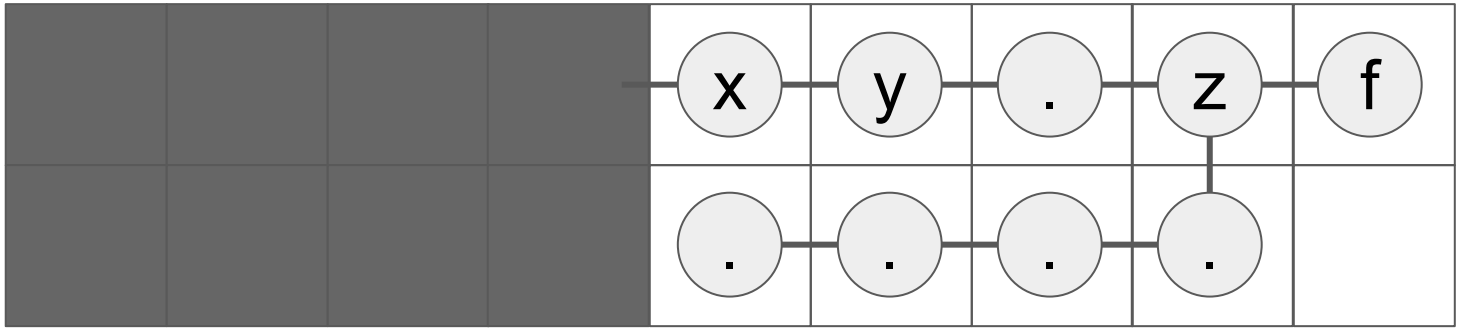


Originally we would transition to state (y, 2), but what kinds of embeddings would we miss if we transitioned to (y, 1) instead?



Originally we would transition to state (y, 2), but what kinds of embeddings would we miss if we transitioned to (y, 1) instead?

We need to identify the node z in the subtree, and assign its neighbour, and verify that there is a chain of the right size going back all the way in the other row.



We've reduced the number of states to O(N) and with some careful programming and precomputation, all the transitions can be done in O(1), so the overall complexity is O(N).