

5.3. Dyck paths to binary trees. Last time we constructed Dyck paths out of binary trees. This can be described in set theoretic language as follows. We constructed a mapping

$$f : \{\text{rooted binary trees with } n \text{ nodes}\} \rightarrow \{\text{Dyck paths of length } 2n\}$$

Today we constructed the backwards mapping

$$g : \{\text{Dyck paths of length } 2n\} \rightarrow \{\text{rooted binary trees with } n \text{ nodes}\}$$

The inverse function g (if it is correct) satisfies the following two properties:

- (1) $f \circ g = id$ on Dyck paths
- (2) $g \circ f = id$ on binary trees.

The description of the function g was *recursive*. We took each Dyck path and made it into one or two shorter paths. We used induction on the number n to get the binary tree for these shorter paths, then put them together to get the answer. The recursive algorithm “calls itself”, i.e., has a loop. But, since the length gets shorter, it does not loop forever.

We took two examples.

- (1) $LRLRLLRR$
- (2) $LLRLRLLRR$

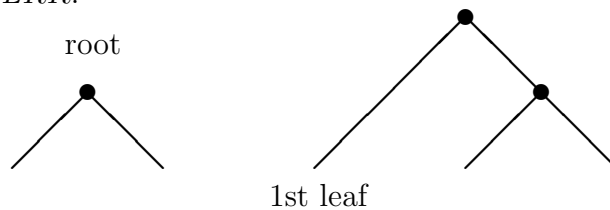
In the first example, we can break the Dyck path into two Dyck paths:

$$LR|LRLRLLRR \quad \text{or} \quad LRLR|LRLRLLRR$$

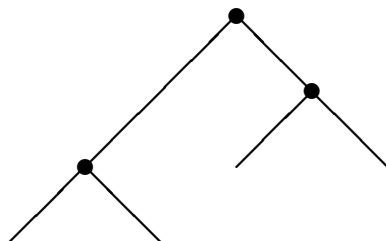
In the second example, there is no breaking point.

5.3.1. Case 1. The Dyck path can be broken into two shorter Dyck paths.

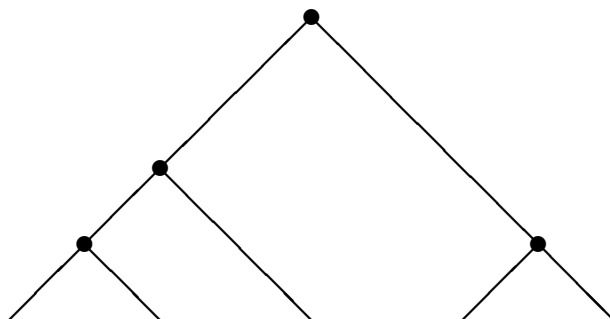
In this case, we make two shorter Dyck paths. Each Dyck path gives a binary tree. To put them together we attach the root of the first tree to the first leaf of the second tree. To make the algorithm deterministic we take the first breaking point. In the example, first we do: $LR|LRLRLLRR$:



to get:



So, $LR|LRLRR$ is given by putting the LR tree on the first leaf of this one. Drawing all leaves at the bottom, we get:



5.3.2. *Case 2.* There is no breaking point.

In this case we get a smaller Dyck path by removing the first and last letter.

Lemma 5.17. *If we are in case 2, i.e., we have a Dyck path so that, whenever we break the path into two parts, there will be more L 's than R 's in the first part, then we get a shorter Dyck path by removing the first and last letter.*

Proof. The first letter in any Dyck path must be L and the last letter must be R . We want to prove that what is in the middle M is a Dyck path. Since LMR has n L 's and n R 's, the middle part M has $n - 1$ L 's and $n - 1$ R 's. We just need to prove that, whenever M is broken into two parts, say $M = AB$ then the first part A has at least as many L 's as R 's.

But the original Dyck path is $LABR$. So, LA is a beginning of the original Dyck path. By assumption, this word has more L 's than R 's. So, if we remove one L we have at least as many L 's as R 's. So, the first part A of the middle word has at least as many L 's as R 's. Therefore the middle word is a Dyck path. \square

In case 2 the Dyck path is LMR where M is the middle part which is a shorter Dyck path. The algorithm is: Construct the binary tree for this middle word, then put it on the right underneath the root:

