# Generate Parentheses

## Problem Description:

Given n pairs of parentheses, write a function to generate all combinations of well-formed parentheses.

Example 1:

* Input: n = 3
* Output: ["((()))","(()())","(())()","()(())","()()()"]

Example 2:

* Input: n = 1
* Output: ["()"]

Constraints:

1 <= n <= 8

## Approach:

A drawing of a triangle

AI-generated content may be incorrect.

The parentheses pattern can be generated by branching and adding one open parenthesis and one close parenthesis. This forms a binary tree as shown in the picture above.

We initially started with one open parenthesis [(] then spread out recursively.

## Tasks

### Create Node data structure

This is to form a binary tree later.

A computer screen shot of text

AI-generated content may be incorrect.

### Create a function to expand parentheses

This function receives a node and an integer which is the maximum number of parentheses allowed.

We check the open/close count of parentheses, if they’ve reached the max value for both open & close then just return, no need to expand.

Otherwise:

* Expand a new node with open ( added to the left
* Expand a new node with close ) added to the right

Note that we cannot have more close parenthesis than open parenthesis as this is invalid, e.g., ()) this is an invalid case so for the close parenthesis we need to ensure the close count < open count

A computer screen shot of a code

AI-generated content may be incorrect.

### Construct the Binary tree of Parentheses

The binary tree is constructed by:

* Creating a first root node with one open (
* A queue used for expanding the tree

The idea is as long as the queue is not empty, we’ll keep expanding the tree.

A computer screen shot of a program code

AI-generated content may be incorrect.

### Traverse to collect all Leaf nodes

After constructing the tree, all the leaf nodes are the generated parentheses.

Use recursive with a storage to collect all the leaf nodes.

A computer screen with text

AI-generated content may be incorrect.

## Performance Evaluation

Runtime Complexity: O(2 \* n)

Space Complexity: O(n)