Documentation

The problem at hand requires removing the minimum number of invalid parentheses from a given string to make it valid. A valid string is defined as one where every opening parenthesis '(' has a corresponding closing parenthesis ')', and the parentheses are balanced. The goal is to return all possible valid strings that can be formed by removing the minimum number of parentheses, ensuring that no duplicate results are included.

The solution approach leverages **Breadth-First Search (BFS)**, which is an ideal technique for this problem because it explores all possible strings at each "depth" (or number of removals) before moving to the next level. This guarantees that the first time a valid string is encountered, it will have the fewest removals. The BFS process starts by enqueueing the original string and then exploring all possible transformations by removing one parenthesis at a time. Once valid strings are found, they are collected, and the search stops at that point to avoid unnecessary work.

To determine whether a string is valid, we need to check if the parentheses are balanced. This involves iterating through the string and maintaining a counter that tracks the balance between opening and closing parentheses. A string is valid if this balance reaches zero at the end of the iteration, and at no point does the balance go negative (which would indicate an unmatched closing parenthesis). This check ensures that the string is well-formed and meets the required conditions for validity.

The BFS algorithm utilizes a queue to explore each string level by level. At each level, the algorithm generates new strings by removing one parenthesis at a time from the current string. These newly formed strings are then added to the queue for further exploration. A set is used to track the strings that have already been visited to avoid redundant checks. This ensures that we do not revisit the same string and reduces the overall time complexity.

Finally, once all valid strings are identified, they are returned as the output. The solution guarantees that the result contains only those strings that are valid and formed by removing the minimum number of parentheses. The use of BFS ensures efficiency in exploring the string space and finding all possible valid configurations, while the set ensures that the result is free from duplicates. This approach is optimal given the problem's constraints and ensures correctness by exploring all potential solutions systematically.