Palindrome Partitioning II Documentation

Problem Statement

Given a string s, partition s such that every substring of the partition is a palindrome. Return the minimum cuts needed for a palindrome partitioning of s.

Example 1

- Input: s = "aab"
- Output: 1
- Explanation: The palindrome partitioning ["aa","b"] can be achieved with 1 cut.

Example 2

- Input: s = "a"
- Output: 0
- Explanation: The string s is already a palindrome, so no cuts are needed.

Example 3

- Input: s = "ab"
- Output: 1
- Explanation: The palindrome partitioning ["a","b"] can be achieved with 1 cut.

Constraints

- 1 <= s.length <= 2000
- s consists of lowercase English letters only.

Solution Explanation

The goal is to find the minimum number of cuts needed to partition the string such that each substring is a palindrome. The provided solution employs dynamic programming to achieve this efficiently.

Step-by-Step Solution

1. Initialization:

- Define n as the length of the string s.
- Create a 2D array is_palindrome of size n x n to store whether a substring s[i:j] is a palindrome.
- Create a 1D array dp of size n to store the minimum number of cuts needed for a palindrome partitioning of the substring s[0:i].

2. Palindrome Detection:

- Initialize all single-character substrings as palindromes in the is_palindrome array.
- Use dynamic programming to fill in the is_palindrome array for all substrings of length 2 to n.
  - For substrings of length 2, check if the two characters are the same.
  - Check if the first and last characters are the same for longer substrings and if the inner substring is a palindrome.

3. Minimum Cuts Calculation:

- Initialize the dp array such that each position i initially assumes the maximum possible cuts (i.e., i cuts for s[0:i]).
- Iterate over each substring ending at i. If the substring s[0:i] is a palindrome, then no cuts are needed.
- For non-palindromic substrings, iterate over possible cuts to find the minimum cuts needed, leveraging previously computed results.

4. **Result:**
    - The result is the value in the last position of the dp array, which represents the minimum cuts needed for the entire string s.

Usage

This method can be used to determine the minimum cuts required to partition any given string s such that every partitioned substring is a palindrome. This is useful in applications where palindromic substrings have specific significance or constraints, such as in certain data compression techniques or error correction algorithms.