Simplify Path Documentation

Problem Description

Given an absolute path for a Unix-style file system, which begins with a slash '/', the goal is to transform this path into its simplified canonical form.

Rules for Simplification

1. A single period '.' signifies the current directory.
2. A double period '..' signifies moving up one directory level.
3. Multiple slashes '//' are interpreted as a single slash '/.
4. Sequences of periods not covered by the above rules (e.g., '...') are treated as valid names for files or directories.

Requirements for the Simplified Path

- It must start with a single slash '/'.
- Directories within the path should be separated by a single slash '/'.
- It should not end with a slash '/', unless it is the root directory.
- It should exclude any single or double periods used to denote current or parent directories.

Examples

1. **Example 1:**

- Input: "/home/"
- Output: "/home"
- Explanation: The trailing slash should be removed.

2. **Example 2:**

- Input: "/home//foo/"
- Output: "/home/foo"
- Explanation: Multiple consecutive slashes are replaced by a single one.

3. **Example 3:**

- Input: "/home/user/Documents/../Pictures"
- Output: "/home/user/Pictures"
- Explanation: A double period ".." refers to the directory up a level.

4. **Example 4:**

- Input: "/../"
- Output: "/"
- Explanation: Going one level up from the root directory is not possible.

5. **Example 5:**

- Input: "/.../a/../b/c/../d/./"
- Output: "/.../b/d"
- Explanation: "..." is a valid name for a directory in this problem.

Constraints

- The length of the path is between 1 and 3000 characters.
- The path consists of English letters, digits, periods '.', slashes '/', or underscores '_'.
- The path is a valid absolute Unix path.

Solution Explanation

The provided solution involves processing the input path to generate its simplified canonical form. The approach uses a stack data structure to handle the directory navigation.

Steps of the Solution

1. **Splitting the Path:**
    - The path is split by '/' to handle each component separately.

2. **Processing Each Component:**
    - Iterate over each component of the split path.
    - Ignore empty components or single periods '.' as they represent the current directory.
    - For double periods '..', pop an element from the stack if the stack is not empty, which moves up one directory level.
    - Push valid directory names onto the stack.

3. **Forming the Simplified Path:**
    - Join the components in the stack with a single slash '/' to form the simplified path.
    - Prepend a single slash '/' to the result as it is an absolute path.

Explanation of the Code

- Initialization: The input path is split into components using split('/').
- Stack Operations: Iterate through each component, and based on its value ('', '.', '..', or a valid directory name), decide whether to skip, pop from the stack, or push onto the stack.
- Final Path Construction: Join the stack with '/' and prepend '/' to form the final simplified path.