The task requires implementing a function, which checks if a given string `S` is properly nested.

A string is considered properly nested if it meets one of the following conditions: (1) it is empty, (2) it has the form "(U)", "[U]", or "{U}" where U is a properly nested string, or (3) it has the form "VW" where V and W are properly nested strings.

The function should return True if `S` is properly nested and False otherwise. The input string `S` consists of N characters and is guaranteed to meet the following constraints: N is an integer within the range [0..200,000], and the characters in `S` are limited to '(', '{', '[', ']', '}', or ')'.

The task requires an efficient algorithm to handle the given constraints and accurately determine if the string is properly nested.

| def test(func):  legals = ['(()[[]][])',  '[[]{}{{}}]',  '{[]{[][]}}',  '()((){[]})',  '{([][]{})}',  '[[][()]]{}',  '(([]{[]}))',  '[]{[[]{}]}',  '{[[]]{}}()',  '{{}}[{()}]']  illegal = ['}}))[{)({]',  '{)({([)){}',  '{{}[((]}}]',  '[){{{{{)}(',  '{}[[}]}(]{',  '}[]]{[})[{',  '][[([}[)()',  '[)(]){]}(]',  '(]}}[)})]]',  '[)((])]{(}']  for s in legals:  assert func(s), s  for s in illegal:  assert not func(s),s |
| --- |