Valid Parentheses

Question:

https://leetcode.com/problems/valid-parentheses/

Given a string `s` containing just the characters `'('`, `')'`, `'{'`, `'}'`, `'['` and `']'`, determine if the input string is valid.

An input string is valid if:

1. *Open brackets must be closed by the same type of brackets.*
2. *Open brackets must be closed in the correct order.*

Example 1:

```
Input: s = "()"
Output: true
```

Example 2:

```
Input: s = "()[]{}"
Output: true
```

Example 3:

```
Input: s = "(]"
Output: false
```

Approach:

We always check for correct parentheses when we find a closing bracket and check it with the bracket just before it.

Using a stack made sense since we would only input opening brackets and when we encounter a closing bracket, we check if it is matching with the last opening bracket in the stack.

If not we return False as the parentheses are not balanced.

After the entire process, we should have an empty stack for balanced parentheses.

Solution:

https://gist.github.com/vermaayush680/0a8c14298163e543640dab4027c95b1d

```
"""
Time Complexity: O(n)
Space Complexity: O(n)
"""

stack=[]

d={'}':'{',')':'(',']':'['}
for i in s:
        if i in d:
                if not stack:
                        return False
                a=stack.pop()
                if d[i]!=a:
                        return False
        else:
                stack.append(i)
if stack:
        return False
return True
```