2018-10-01, 4:31 PM Remove Comments - LeetCode

## 722. Remove Comments



■ Description (/problems/remove-comments/description/)

♀ Hints (/problems/remove-comments/hints/)

Quick Navigation -

View in Article ☑ (/articles/remove-comment ∰

# Approach #1: Parsing [Accepted]

### Intuition and Algorithm

We need to parse the source line by line. Our state is that we either are in a block comment or not.

- If we start a block comment and we aren't in a block, then we will skip over the next two characters and change our state to be in a block.
- If we end a block comment and we are in a block, then we will skip over the next two characters and change our state to be not in a block.
- If we start a line comment and we aren't in a block, then we will ignore the rest of the line.
- If we aren't in a block comment (and it wasn't the start of a comment), we will record the character we are
- At the end of each line, if we aren't in a block, we will record the line.

### **Python**

```
class Solution(object):
    def removeComments(self, source):
        in block = False
        ans = []
        for line in source:
            i = 0
            if not in_block:
                newline = []
            while i < len(line):</pre>
                if line[i:i+2] == '/*' and not in_block:
                    in_block = True
                elif line[i:i+2] == '*/' and in_block:
                    in_block = False
                    i += 1
                elif not in_block and line[i:i+2] == '//':
                    break
                elif not in_block:
                    newline.append(line[i])
            if newline and not in_block:
                ans.append("".join(newline))
        return ans
```

Java

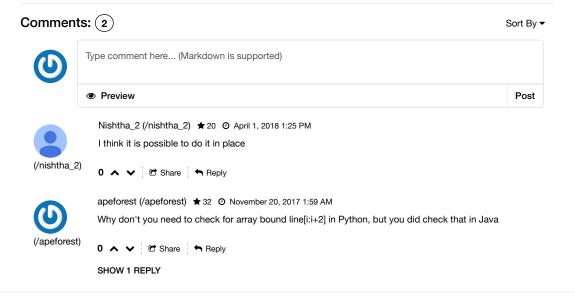
Remove Comments - LeetCode 2018-10-01, 4:31 PM

```
class Solution {
    public List<String> removeComments(String[] source) {
        boolean inBlock = false;
        StringBuilder newline = new StringBuilder();
        List<String> ans = new ArrayList();
        for (String line: source) {
            int i = 0;
            char[] chars = line.toCharArray();
            if (!inBlock) newline = new StringBuilder();
            while (i < line.length()) {</pre>
                if (!inBlock && i+1 < line.length() && chars[i] == '/' && chars[i+1] == '*') {
                    inBlock = true;
                    i++:
                } else if (inBlock && i+1 < line.length() && chars[i] == '*' && chars[i+1] == '/') {
                    inBlock = false;
                } else if (!inBlock && i+1 < line.length() && chars[i] == '/' && chars[i+1] == '/') {</pre>
                    break:
                } else if (!inBlock) {
                    newline.append(chars[i]);
                i++;
            if (!inBlock && newline.length() > 0) {
                ans.add(new String(newline));
        }
        return ans;
    }
}
```

### **Complexity Analysis**

- ullet Time Complexity: O(S), where S is the total length of the source code.
- Space Complexity: O(S), the space used by recording the source code into ans .

Analysis written by: @awice (https://leetcode.com/awice).



Copyright © 2018 LeetCode

Contact Us (/support/) | Jobs (/jobs/) | Students (/students) | Frequently Asked Questions (/faq/) | Terms of Service (/terms/) | Privacy Policy (/privacy/) 

— United States (/region/