3163. String Compression III

Given a string `word`, compress it using the following algorithm:

- Begin with an empty string `comp`. While `word` is **not** empty, use the following operation:
  - Remove a maximum length prefix of `word` made of a *single character* `c` repeating **at most** 9 times.
  - Append the length of the prefix followed by `c` to `comp`.

Return the string `comp`.

Example 1:

Input: word = "abcde"

Output: "1a1b1c1d1e"

Explanation:

Initially, `comp = ""`. Apply the operation 5 times, choosing `"a"`, `"b"`, `"c"`, `"d"`, and `"e"` as the prefix in each operation.

For each prefix, append `"1"` followed by the character to `comp`.

Example 2:

Input: word = "aaaaaaaaaaaaabb"

Output: "9a5a2b"

Explanation:

Initially, `comp = ""`. Apply the operation 3 times, choosing `"aaaaaaaaa"`, `"aaaaa"`, and `"bb"` as the prefix in each operation.

- For prefix `"aaaaaaaaa"`, append `"9"` followed by `"a"` to `comp`.
- For prefix `"aaaaa"`, append `"5"` followed by `"a"` to `comp`.
- For prefix `"bb"`, append `"2"` followed by `"b"` to `comp`.

Constraints:

- $1 <= word.length <= 2 * 10^5$
- `word` consists only of lowercase English letters.

3163. String Compression III

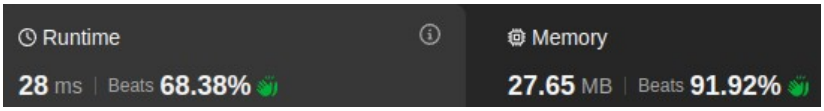
/*

Two pointers

Time Complexity: O(n)

Space complexity: O(1)

*/




```
class Solution {
    public:
        std::string compressedString(std::string word) {
            int n=word.size();

            std::string ans;
            int i=0,j=0;
            while(i<n){
                if(word[i]==word[j]) j++;
                else{
                    int len=j-i;
                    if(len<=9){
                        ans.push_back(len+'0');
                        ans.push_back(word[i]);
                    }
                    else{
                        int count=len/9;
                        for(int k=1;k<=count;++k){
                            ans.push_back('9');
                            ans.push_back(word[i]);
                        }
                        if(len%9!=0){
                            ans.push_back((len%9)+'0');
                            ans.push_back(word[i]);
                        }
                    }
                    i=j;
                }
            }
            return ans;
        }
};
```

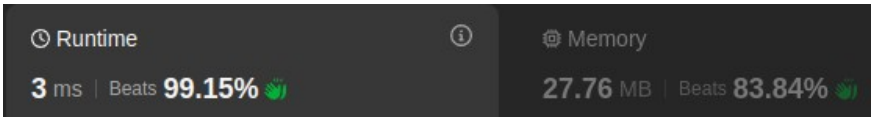
3163. String Compression III

/*

one pointer

Time Complexity: O(n)

Space complexity: O(1)



*/

```
class Solution {
    public:
        std::string compressedString(std::string word) {
            int n=word.size();

            std::string ans;
            int i=0;
            while(i<n){
                int count=0;
                char cur=word[i];
                while(i<n && count<9 && word[i]==cur){
                    count++;
                    i++;
                }
                ans.push_back(count+'0');
                ans.push_back(cur);
            }
            return ans;
        }
};
```