1957. Delete Characters to Make Fancy String

A **fancy string** is a string where no **three consecutive** characters are equal.

Given a string `s`, delete the **minimum** possible number of characters from `s` to make it **fancy**.

Return *the final string after the deletion*. It can be shown that the answer will always be **unique**.

Example 1:

```
Input: s = "leeetcode"
Output: "leetcode"
Explanation:
Remove an 'e' from the first group of 'e's to create "leetcode".
No three consecutive characters are equal, so return "leetcode".
```

Example 2:

```
Input: s = "aaabaaaa"
Output: "aabaa"
Explanation:
Remove an 'a' from the first group of 'a's to create "aabaaaa".
Remove two 'a's from the second group of 'a's to create "aabaa".
No three consecutive characters are equal, so return "aabaa".
```

Example 3:

```
Input: s = "aab"
Output: "aab"
Explanation: No three consecutive characters are equal, so return "aab".
```

Constraints:

- `1 <= s.length <= 10^5`
- `s` consists only of lowercase English letters.

1957. Delete Characters to Make Fancy String

```
/*
   Queue
   Time complexity: O(n)
   Space complexity: O(n)
*/
class Solution {
   public:
      std::string makeFancyString(std::string s){
         int n=s.size();

         // Use a queue to store the indices of the characters to remove
         std::queue<int> to_remove;
         for(int i=1;i<n-1;++i){
            if(s[i]==s[i-1] && s[i]==s[i+1]) to_remove.push(i);
         }


         std::string ans;
         for(int i=0;i<n;++i){
            // Get indices of the characters to remove
            int j=to_remove.front();

            // If the current character index is different from
            // the index extracted from the queue
            // and the current character to the answer
            if(i!=j) ans.push_back(s[i]);

            // Otherwise, skip that cuurent character and
            // pass to the next index of the character to remove
            else to_remove.pop();
         }

         return ans;
      }
};
```

1957. Delete Characters to Make Fancy String

```
/*
   Two pointers
   Time complexity: O(n)
   Space complexity: O(n)
*/
class Solution {
   public:
      std::string makeFancyString(std::string s){
         int n=s.size();
         int left=0,right=0;
         std::string ans;
         while(left<n){
            // if same character extend the window from the right
            if(right<n && s[left]==s[right]) right++;

            // Otherwise
            else{
               int len=right-left; // Window's length

               // If window's length >= 2, keep only two characters
               if(len>=2) ans.push_back(s[left]),ans.push_back(s[left]);

               // Otherwise, don't delete that character
               else ans.push_back(s[left]);

               // Start a new window
               left=right;
            }
         }
         return ans;
      }
};
```

1957. Delete Characters to Make Fancy String

```
/*
   Two pointers space optimization
   Time complexity: O(n)
   Space complexity: O(1)
*/
class Solution {
public:
   string makeFancyString(string s) {
      // If size of string is less than 3, return it.
      if (s.length() < 3) {
         return s;
      }
      int w = 2;
      for (int r = 2; r < s.size(); ++r) {
         // If the current character is not equal to the previously inserted
         // two characters, then we can add it to the string.
         if (s[r] != s[w - 1] || s[r] != s[w - 2]) {
            s[w++] = s[r];
         }
      }

      // Resize the string to the number of characters added in the string,
      // given by w.
      s.resize(w);
      return s;
   }
};
```