420. Strong Password Checker

Add to List

QuestionEditorial Solution

My Submissions

- Total Accepted: **1612**
- Total Submissions: **7696**
- Difficulty: **Hard**
- Contributors: **yduan7**

A password is considered strong if below conditions are all met:

1. It has at least 6 characters and at most 20 characters.
2. It must contain at least one lowercase letter, at least one uppercase letter, and at least one digit.
3. It must NOT contain three repeating characters in a row ("...aaa..." is weak, but "...aa...a..." is strong, assuming other conditions are met).

Write a function strongPasswordChecker(s), that takes a string s as input, and return the **MINIMUM** change required to make s a strong password. If s is already strong, return 0.

Insertion, deletion or replace of any one character are all considered as one change.

```
class Solution {
public:
    int strongPasswordChecker(string s) {
       if(s.size()<2) return 6-s.size();
      char end = s[0];
      bool upper = isupper(end);
      bool digital = isdigit(end);
      bool lower = islower(end);

      int end_rep = 1; int change = 0;
      int deleteArr[3];
      for(int i=0;i<3;++i) deleteArr[i]=0;
      for(int i=1;i<s.size();++i)
      {
             if(end==s[i]) end_rep++;
             else{
                    change+=end_rep/3;
                    if(end_rep/3>0) ++deleteArr[end_rep%3];
                    upper |= isupper(s[i]);
                    digital |= isdigit(s[i]);
                    lower |= islower(s[i]);
                    end = s[i];
                    end_rep=1;
             }
      }
      change+=end_rep/3;
```

```
        if(end_rep/3>0) ++deleteArr[end_rep%3];

        int check_req = upper==true?0:1;
        check_req+=lower==true?0:1;
        check_req+=digital==true?0:1;


        if(s.size()>20)
        {
                int del = s.size()-20;
                if(del<=deleteArr[0]) change-=del;
                else if(del-deleteArr[0]<=2*deleteArr[1]) change-=deleteArr[0]+(del-
deleteArr[0])/2;
                else change-=deleteArr[0]+deleteArr[1]+(del-
deleteArr[0]-2*deleteArr[1])/3;
                return del+max(check_req,change);
        }

        int ta = max(check_req,change);
        int tb = 6-s.size();
        return max(ta,tb);
     }
};
```