664. Strange Printer

There is a strange printer with the following two special properties:

- The printer can only print a sequence of **the same character** each time.
- At each turn, the printer can print new characters starting from and ending at any place and will cover the original existing characters.

Given a string `s`, return *the minimum number of turns the printer needed to print it*.

Example 1:

```
Input: s = "aaabbb"
Output: 2
Explanation: Print "aaa" first and then print "bbb".
```

Example 2:

```
Input: s = "aba"
Output: 2
Explanation: Print "aaa" first and then print "b" from the second place of the
string, which will cover the existing character 'a'.
```

Constraints:

- `1 <= s.length <= 100`
- `s` consists of lowercase English letters.

664. Strange Printer

```
/*
   Memoization
   Time complexity: O(n^3)
   Space complexity: O(n^2)
*/
typedef std::vector<int> vi;
typedef std::vector<vi> vvi;

class Solution {
   public:
      int strangePrinter(string s) {
         int n=s.size();

         vvi memo(n,vi(n,-1));

         auto solve=[&](int i,int j, auto& self)->int{
         if (i==j) return 1;

         if(memo[i][j]!=-1) return memo[i][j];

         int ans=INT_MAX;

         for (int k=i;k<j;++k) {
               ans=std::min(ans, self(i,k,self)+self(k+1,j,self));
         }

         return memo[i][j]=s[i]==s[j]?ans-1:ans;

         };


         return solve(0,n-1,solve);
      }
};
```

664. Strange Printer

```
/*
   Memoization
   Time complexity: O(n^3)
   Space complexity: O(n^2)
*/
typedef std::vector<int> vi;
typedef std::vector<vi> vvi;

class Solution {
   public:
      // Remove duplicates: aaabbbaaaacccaaa=>abaca
      std::string compression(std::string& s){
         std::string res;
         int n=s.size(),i=0;
         while(i<n){
            char c=s[i];
            res.push_back(c);
            while(i<n&&c==s[i]) i++;
         }
         return res;
      }

      int strangePrinter(std::string s) {

         s=compression(s);

         int n=s.size();

         vvi memo(n,vi(n,-1));

         auto solve=[&](int i,int j, auto& self)->int{

         if (i==j) return 1;

         if(memo[i][j]!=-1) return memo[i][j];

         int ans=INT_MAX;

         for (int k=i;k<j;++k) {

               ans=std::min(ans, self(i,k,self)+self(k+1,j,self));

         }

         return memo[i][j]=s[i]==s[j]?ans-1:ans;

         };

         return solve(0,n-1,solve);

      }
};
```

664. Strange Printer

```
/*
   Tabulation
   Time complexity: O(n^3)
   Space complexity: O(n^2)
*/
typedef std::vector<int> vi;
typedef std::vector<vi> vvi;

class Solution {
   public:
      // Remove duplicates: aaabbbaaaacccaaa=>abaca
      std::string compression(std::string& s){
         std::string res;
         int n=s.size(),i=0;
         while(i<n){
            char c=s[i];
            res.push_back(c);
            while(i<n&&c==s[i]) i++;
         }
         return res;
      }

      int strangePrinter(std::string s) {

         s=compression(s);

         int n=s.size();

         vvi dp(n,vi(n,INT_MAX/2));

         for(int i=0;i<n;++i) dp[i][i]=1;

         for(int length=2;length<=n;++length){
            for(int i=0;i<=n-length;++i){
               int j=i+length-1;
               dp[i][j]=length;
               for(int k=0;k<length-1;++k){
                  dp[i][j]=std::min(dp[i][j],dp[i][i+k]+dp[i+k+1][j]-(int(s[i]==s[j])));
               }
            }
         }
        return dp[0][n-1];
      }
};
```