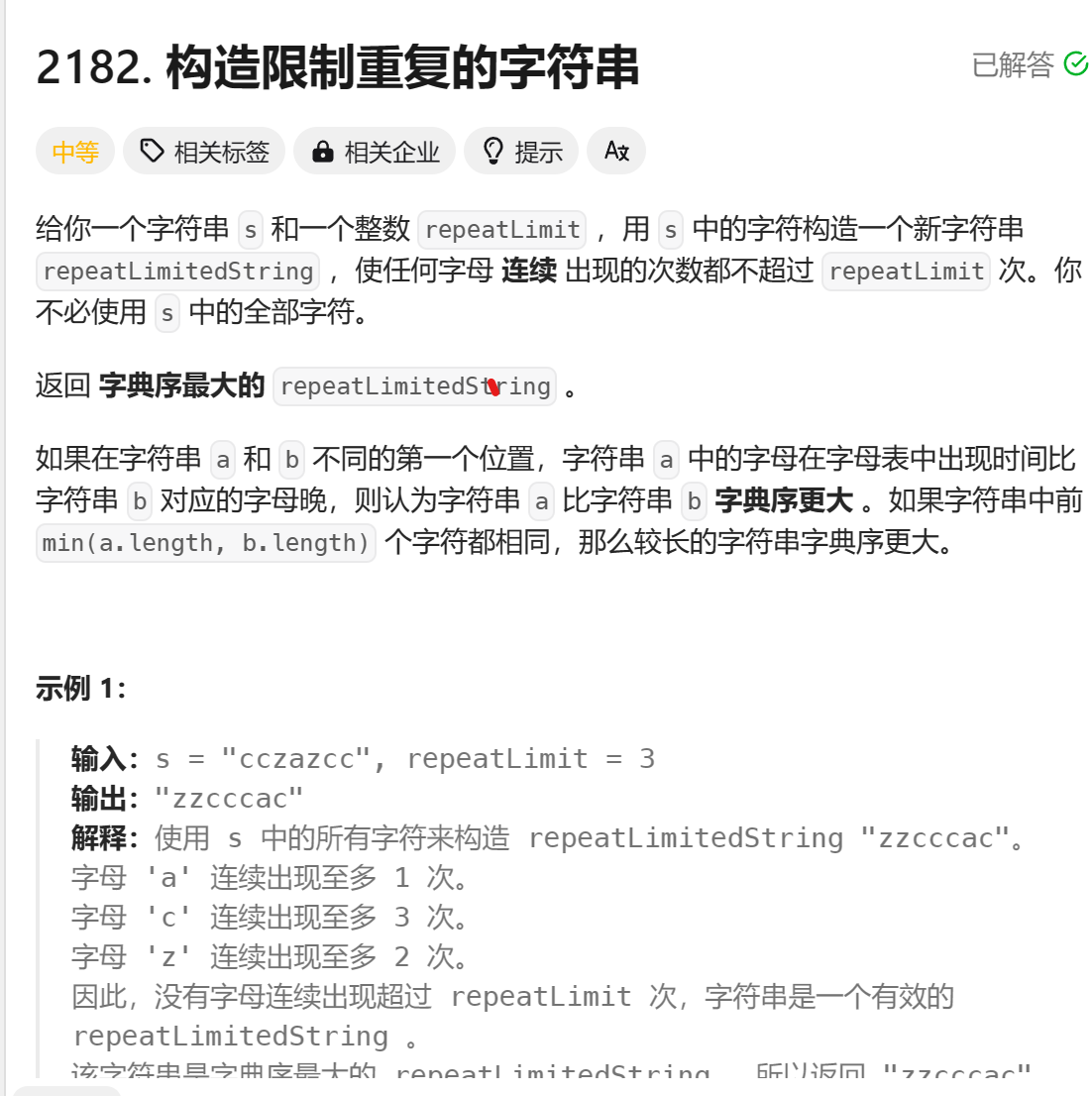
双指针模板总结



这题暴露了我在代码能力构思的薄弱 ，明明已经想到什么样的情况会得到最答案却不会运用代码技巧。

 int N=26;

    string repeatLimitedString(string s, int repeatLimit) {

        // 当需要不断两个对象进行操作-> 例如不断需要取最大的目标和次大的目标 这类的题目需要双指针去指向最大和次大的目标

           vector<int> count(N);

           string ans;

           for(auto key:s){

               count[key-'a']++;

           }

           int m=0;

           //i指向字典序最大指针，j指向字典次大的

           for(int i=N-1,j=N-2;i>=0&&j>=0;){

               if(count[i]==0){

                m=0;

                i--;

               }

               else if(m<repeatLimit&&count[i]>0){

                       ans.push\_back('a'+i);

                       count[i]--;

                       m++;

               }

               else if(i<=j||count[j]==0){//要一直保证j指向次大的字符串必须要先保证 j一直是指向次大的字符

                   j--;

               }

                else if(m>=repeatLimit&&count[j]>0){

                   count[j]--;

                   ans.push\_back('a'+j);

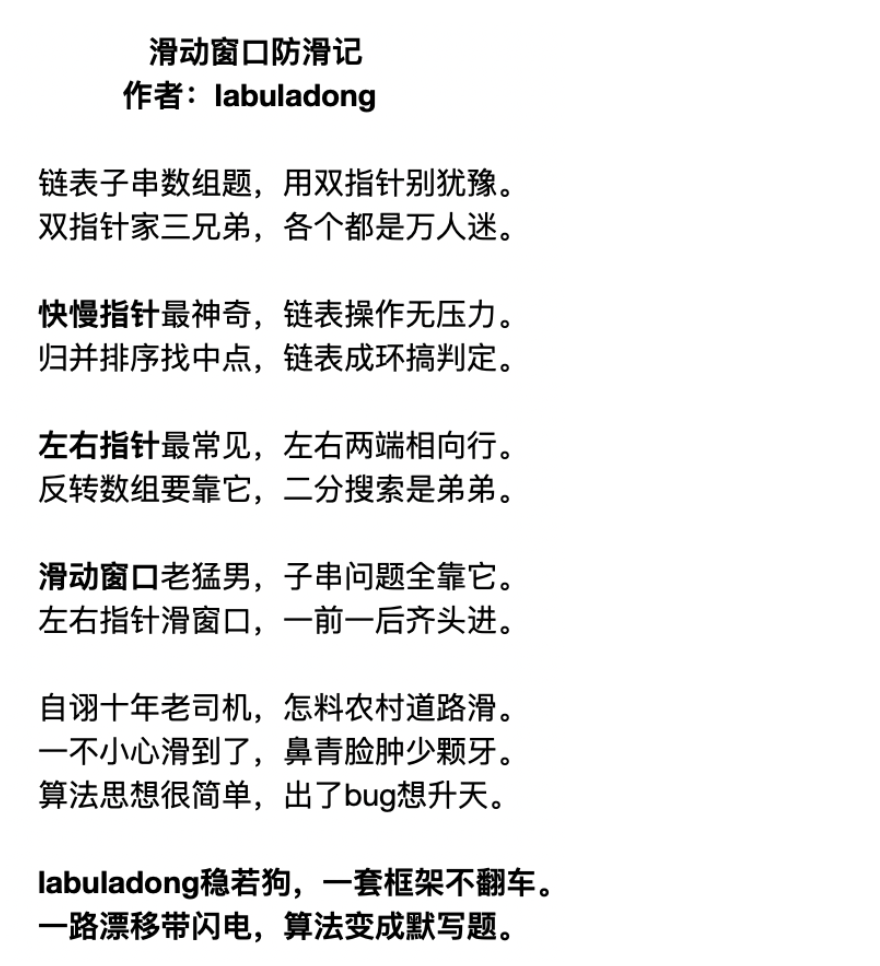
                   m=0;

               }

           }

           return ans;

滑动窗口（模板）





  vector<int> res ;

       int n = s.size();

       int len = p.size();

       if(len > n){

           return {};

       }

       map<char,int> hashs,hashp;//hashs用来记录一个窗口内的的各个字符个数

       //hashp是用来记录p对应的各个字符个数用来与窗口内的字符个数比较

       //滑动窗口需要先扩大窗口大小再根据需要缩小窗口大小

       for(auto &k :p){

           hashp[k]++;

       }

       int left =0 ;

       int right = 0;

       int cnt = 0 ;//用来记录窗口大小

       while(right < n){//窗口变大

               char c =s[right];

            hashs[s[right]]++;

            right++;

            cnt++;

            while(hashs[c] > hashp[c]){//窗口收缩

                hashs[s[left]]--;

                left++;

                cnt-- ;

            }

            if(cnt == len){

                res.push\_back(left);

            }

       }

    return res;

    }