给定一个字符串s，你可以从中删除一些字符，使得剩下的串是一个回文串。如何删除才能使得回文串最长呢？

输出需要删除的字符个数。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | import java.util.\*;  public class Main{  public static void main(String[] args){  Scanner sc = new Scanner(System.in);  while(sc.hasNext()){  String str =sc.nextLine();  char[] strchar = str.toCharArray();  int length= strchar.length;  int[][] dp = new int[length][length];  for(int j=1;j<length;j++){  dp[j-1][j]=strchar[j-1]==strchar[j]?0:1;  for(int i=j-2;i>-1;i--){  if(strchar[i]==strchar[j]){  dp[i][j]=dp[i+1][j-1];  }else{  dp[i][j]=Math.min(dp[i+1][j],dp[i][j-1])+1;  }  }  }  }  }  } |

小 Q 最近遇到了一个难题：把一个字符串的大写字母放到字符串的后面，各个字符的相对位置不变，且不能申请额外的空间。

你能帮帮小Q 吗？

不能构建额外空间，那么交换移动元素使用位操作的那个版本 swap()

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | #include<iostream>  #include<string> using namespace std;    bool isCap(char c)  {  if (c >= 'A' && c <= 'Z')  return true;  else  return false;  } |

|  |  |
| --- | --- |
| 12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44 | void mSwap(char &a, char &b)  {  if (a != b)  {  a ^= b;  b ^= a;  a ^= b;  }  }    int main()  {  string s;  while (cin >> s)  {  int len = s.size();  int end = len;  for (int i = 0; i<end; ++i)  {  if (isCap(s[i]))  {  int j = i;  for (; j<len- 1; ++j)  mSwap(s[j], s[j + 1]);  --end;  --i;  }  }  cout << s <<endl;  }  return 0;  } |

小 Q 今天在上厕所时想到了这个问题：有 n 个数，两两组成二元组，差的绝对值最小的有多少对呢？差的绝对值最大的呢？

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | #include <iostream>  #include <map> #include <utility> using namespace std;  // 用一个 map 来存储输入的数，当存在相同的数时不插入新的数，而是将计数值+1 int main()  { |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 8 |  |  |  |  | int num; |
| 9 |  |  |  |  | while(cin>>num) |
| 10 |  |  |  |  | { |
| 11 |  |  |  |  | map<int,int> myMap; |
| 12 |  |  |  |  | bool flag = false; |
| 13 |  |  |  |  | for(int i = 0; i < num ; i++) |
| 14 |  |  |  |  | { |
| 15 |  |  |  |  | int k ; |
| 16 |  |  |  |  | cin>>k; |
| 17 |  |  |  |  | map<int,int>::iterator ite; |
| 18 |  |  |  |  | ite = myMap.find(k); |
| 19 |  |  |  |  | if(ite != myMap.end()) |
| 20 |  |  |  |  | { (\*ite).second++;flag = true;} |
| 21 |  |  |  |  | else |
| 22 |  |  |  |  | { |
| 23 |  |  |  |  | myMap.insert(make\_pair(k,1)); |
| 24 |  |  |  |  | } |
| 25 |  |  |  |  | } // end of for 读取输入的数据 |
| 26 |  |  |  |  | map<int,int>::iterator ite = myMap.begin(); |
| 27 |  |  |  |  | int min =0; |
| 28 |  |  |  |  | int minv = -1; |
| 29 |  |  |  |  | if(flag) //如果存在相同的数 |
| 30 |  |  |  |  | { |
| 31 |  |  |  |  | for( ; ite!= myMap.end(); ite++) |
| 32 |  |  |  |  | { |
| 33 |  |  |  |  | if((\*ite).second > 1) |

34 min += ((\*ite).second \* ((\*ite).second -1 ))/2;

35

36

37

38

} //最小差元组对数等于所有相等的数构成的元组对

}

else

{

39 for( map<int,int>::iterator ite2 = (++myMap.begin()); (ite2)!=

40

41

42

43

ite2++,ite++ )

{

int k = (\*(ite2)).first - (\*(ite)).first;

if( minv ==-1 || k < minv )

44 { min = (\*ite).second \* (\*ite2).secon

45

46

47

48

49

50

51

minv = k; }

else if(minv == k)

{

min+= (\*ite).second \* (\*ite2).second;

}

} // end of for 求不存在相等的值时的最小差的元组对数



|  |  |
| --- | --- |
| 52  53  54  55  56 | }// 最小对的个数  int max = (\*myMap.rbegin()).second \* (\*myMap.begin()).second;  //最大差的对数  cout<< min<<" "<<max<<endl;    }  } |