730. Count Different Palindromic Subsequences

DescriptionHintsSubmissionsDiscussSolution

- Difficulty:Hard
- Total Accepted:255
- Total Submissions:1.1K
- Contributor: imsure

Given a string S, find the number of different non-empty palindromic subsequences in S, and **return that number modulo `10^9 + 7`.**

A subsequence of a string S is obtained by deleting 0 or more characters from S.

A sequence is palindromic if it is equal to the sequence reversed.

Two sequences `A_1, A_2, ...` and `B_1, B_2, ...` are different if there is some `i` for which `A_i != B_i`.

Example 1:

```
Input:
S = 'bccb'
Output: 6
Explanation:
The 6 different non-empty palindromic subsequences are 'b', 'c', 'bb', 'cc',
'bcb', 'bccb'.
Note that 'bcb' is counted only once, even though it occurs twice.
```

Example 2:

```
Input:
S = 'abcdabcdabcdabcdabcdabcdabcddcbadcbadcbadcbadcbadcbadcbadcba'
Output: 104860361
Explanation:
There are 3104860382 different non-empty palindromic subsequences, which is
104860361 modulo 10^9 + 7.
```

Note:

- The length of S will be in the range `[1, 1000]`.
- Each character `S[i]` will be in the set `{'a', 'b', 'c', 'd'}`.

```
#include<stdio.h>
#include<iostream>
#include<vector>
#include<algorithm>
#include<string>
#include<stdlib.h>
#include<string.h>
#include<unordered_map>
```

```
#include<map>
using namespace std;
static int div2 = 1000000007;
// >=
int ceiling(vector<int> &num, int val)
{
     for(auto elem:num)
     {
          if(elem>=val) return elem;
     }
     return -1;
}
//<
int lower(vector<int> &num, int val)
{
     for(int i=num.size()-1;i>=0;i--)
     {
          int elem = num[i];
          if(elem<val) return elem;
     }
     return -1;
}

int helper(string s, vector<vector<int>> &chars,
vector<vector<int>> &dp,
             int start, int end)
{
     if(start>=end) return 0;
     if(dp[start][end]) return dp[start][end];
     long ans = 0;
     for(int i=0;i<26;i++)
     {
          if((int)chars[i].size()==0) continue;
          int newstart = ceiling(chars[i],start);
          int newend = lower(chars[i],end);
          if(newstart>=end || newend==-1 || newstart==-1) continue;
          printf("%d,%d--%d\n",newstart,newend,i);
          ans++;
          if(newstart!=newend) ans++;
          ans+=helper(s,chars,dp,newstart+1,newend);
     }
     dp[start][end] = (int)(ans%div2);
     return dp[start][end];
}

static bool cmp(int a, int b)
{
     return a<b;
}
int countPalindromicSubsequences(string s) {
     int n = (int)s.size();
     vector<vector<int>> chars(26,vector<int>());
     for(int i=0;i<n;i++)
```

```
     {
          char c = s[i];
          chars[c-'a'].push_back(i);
     }
     for(int i=0;i<26;i++)
     {
          if((int)chars[i].size())
          {
               sort(chars[i].begin(),chars[i].end(),cmp);
          }
     }
     vector<vector<int>> dp(n+1,vector<int>(n+1,0));
     int res = helper(s,chars,dp,0,n);
     return res;
}

int main(int argc, char *argv[])
{
     int ans = countPalindromicSubsequences("abbb");
     cout << ans << endl;
     return 0;
}
```