Q1

(a)

```
int countPalindromeCut (string s) {
    int n = s.size(), maxlen = 0;
    vector<vector<boolean>> dp (n, vector<boolean>(n, false));
    for (int i=1; i<n; i++) dp[i][i] = true;
    for (int i=n-2; i>=0; i--){
      for (int j=i+1; j<n; i++ ) {
          len = j-i+1;
          if s[i]==s[j] {
              if length <3 {
                  dp[i][j] = true;
              } else {
                  dp[i][j] = dp[i+1][j-1];
              }
          }
      }
    }
```

$O(n^2)$

Ps. 有 O(n) 解法，估计不要求

REF: leetcode.cn 132题

```
    vector<int> cut(n, 0)
    for(int j=1; j<n; j++){
        if (df[0][j]) {
            cut[j]=0;
        } else{
            cut[j] = cut[j-1]+1;
            for (int i=1; i<j; i++) {
                if (df[i][j]) {
                    cut[j] = min(cut[j], cut[i-1]+1)
                }
            }
        }
    }
    return cut[n-1];
}
```

(b)

```
int lenOfLIS (string s){
    int [] dp = new int [s.size()];
    for (i=0 ; i<s.size() ; i++)   dp[i] = 1 ;
    for (j=0 ; j<s.size() ; j++){
        for (i=0; i<j ; i++) {
            if ( s[i] < s[j] )
                dp[i] = max(dp[i]+1 , dp[j]);
        }
    }

    int res=0;
    for (i=0; i < s.size(); i++)
        res = max(res , dp[i];

    return res
}
```

$O(n^2)$ PS. 有 $O(n\log n)$ 的二分搜索法 但不太容易想到

REF: leetcode.cn 第300题