**Palindromic Partitioning: -**

**Hard Accuracy: 27.82% Submissions: 110K+ Points: 8**

Given a string **str**, a partitioning of the string is a palindrome partitioning if every sub-string of the partition is a palindrome. Determine the fewest cuts needed for palindrome partitioning of the given string.

**Example 1:**

**Input:** str = "ababbbabbababa"

**Output:** 3

**Explaination:** After 3 partitioning substrings

are "a", "babbbab", "b", "ababa".

**Example 2:**

**Input:** str = "aaabba"

**Output:** 1

**Explaination:** The substrings after 1

partitioning are "aa" and "abba".

**Your Task:**  
You do not need to read input or print anything, Your task is to complete the function **palindromicPartition()** which takes the string str as the input parameter and returns the minimum number of partitions required.

**Expected Time Complexity:** O(n\*n) [n is the length of the string str]  
**Expected Auxiliary Space:** O(n\*n)

**Constraints:**  
1 ≤ length of str ≤ 500

**Code: -**

//{ Driver Code Starts

// Initial Template for c++

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

// User function Template for C++

class Solution{

public:

int n;

bool check(string &str, int start, int end){

while(start <= end){

if(str[start++] != str[end--])

return false;

}

return true;

}

int helper(string &str, int start, vector<int> &dp){

// base case

if(start >= n) return 0;

// dp found case

if(dp[start] != -1)

return dp[start];

// recursive case

int cutcount = INT\_MAX;

for(int i = start; i < n; ++i){

if(check(str, start, i) == true){

cutcount = min(cutcount, 1 + helper(str, i+1, dp));

}

}

// return from current state

return dp[start] = cutcount;

}

int palindromicPartition(string str){

n = str.size();

vector<int> dp(n+1, -1);

return helper(str, 0, dp) - 1;

}

};

//{ Driver Code Starts.

int main(){

int t;

cin>>t;

while(t--){

string str;

cin>>str;

Solution ob;

cout<<ob.palindromicPartition(str)<<"\n";

}

return 0;

}

// } Driver Code Ends

**T.C: - O(N2)**

**S.C: - O(N)**