**Question 5**

Divyansh Mathur, 210357

1. **To design a strategy to find number of palindromic substrings in the hidden string so your crew can safely escape from this region in less than or equal to O(n\*log2n)**

**# in whole question 0 based indexing is used**

*Brute force:*

The brute force approach is easily visible, viewing every substring and then counting ++ if this substring is palindrome or not. But this will take O(n2) time.

*Observation:*

1. Odd sized palindrome: Consider an index i, if we assume the length of the largest palindrome such that i is its middle element to be k, then there will be

(k-1)/2+(k-3)/2+…. 1 such palindrome centered at i.

1. Even sized palindrome: Similar approach works here also. If i, i+1 makes a palindrome.

Then check for the longest length (front from i+1 and rear for i) such that i,j makes a palindrome.

*Designing Algorithm:*

First, we will count palindrome for odd length and then for even length sum their answer. Now, to do this in optimal time we would use **binary search on answer**. Let we are standing on an index i, the minimum odd sized palindrome will be of length 0 (we are counting the extent of palindrome to right and left) and the maximum we can go right or left will be M=min{ i, n-i-1} (0 based indexing) now we need to find maximum length k such that S(i-k,i+k) [0<=k<=M] is palindrome also we will add k to the ans.

Then we will check if i , i+1 can be middle element of a palindrome {By ispalindrome(i,i+1)}. If they are then again do binary search minimum again being 0 and maximum being M = min{i,n-i-2} (0 based indexing) now we need to find maximum length k such that S(i-k,i+k) [0<=k<=M] is palindrome also we will add k to the ans.

*Time Complexity:*

In the worst case we might end up applying both the binary search (even and odd) on a index, and there are n index which may make up to 2\*n\*logn operations. Which is less than n\*log2n operations.

*Pseudocode:*

ispalindrome(i,j){

    blackbox function which return true or false whether S(i,j) is palindrome in O(1) time

}

sum=0

for i from 0 to n-1

{

    low=0, high= min{i,n-i-1}, ans

    while(low <= high){

        mid=(low+high)/2

        if(ispalindrome(i-mid,i+mid)== true){

            ans=mid

            low=mid+1

        }else{

            high=mid-1

        }

    }

    sum= sum+ ans+1 // +1 because length 0 also counts as palindrome

    if( i<n-1 and ispalindrome(i,i+1)==true){

        low=0, high= min{i,n-i-2}, ans=0

        while(low <= high){

            mid=(low+high)/2

            if(ispalindrome(i-mid,i+mid+1)== true){

                ans=mid

                low=mid+1

            }else{

                high=mid-1

            }

        }

        sum= sum+ans

    }

}

print(sum)