**Problem :** [**Longest Palindrome**](https://cses.fi/problemset/task/1111/)

**DP Approach :**

**i=starting index,j=ending index**

Pal[i][j] is true iff 1) string[i]==string[j] //First and Last Character are same.

2) pal[i+1][j-1]=1 //The middle one is substring

First,calculate separately for 1 and 2 length strings, and then separately for the rest.

string longestPalindrome(string s) {

int len = s.length();

bool pal[len][len];

int left\_index=0,right\_index=0;

**// Initialize dp table to 0.**

for(int i=0;i<len;i++)

{

for(int j=0;j<len;j++)

{

pal[i][j]=0;

}

}

**// 1 length string always a substring.**

for(int i=0;i<len;i++) pal[i][i]=1;

**// For 2 length substrings.**

for(int i=0;i<(len-1);i++)

{

if(s[i]==s[i+1])

{

pal[i][i+1]=1;

left\_index=i;

right\_index=i+1;

}

}

**// For others.**

for(int i=3;i<=len;i++ )

{

for(int j=0;j<len-i+1;j++)

{

if(s[j]==s[j+i-1] && pal[j+1][j+i-2])

{

pal[j][j+i-1]=1;

left\_index=j;

right\_index=j+i-1;

}

}

}

return s.substr(left\_index,right\_index-left\_index+1);

}

O(n^2) time and space.

**APPROACH2 : MANACHER’S algo in O(n)**

=> **Palindromes are Symmeteric At the Centres.**

=> So we try to expand around every alphabet and update a range l...r(in implementation we don’t need ‘l’).If we are inside the range, we copy from mirror.

We define An array ‘p’,

p[i] stores how many characters after index i are same as before i,basically the length of right half of the Palindrome.

Index 0123456

E.g: **babab**ad

-> Expand around index 1, that is character ‘a’ and we get p[1]=1 as the 2 b’s around it match,so we expand l=0,r=2.

-> Same way,p[2]=2 as ‘ba’ and ‘ab’ around ‘b’ form palindrome.

- > Now at index 3,we are inside the range,so we copy from mirror,, we get p[3]=p[1]=2, (bcoz of symmetry and we are also inside the range so we copy from p[mirror],and it’s true,’b’ and ‘b’ around ‘a’ match)

-> But as we said palindromes are symmetric only till the range,so try to expand further beyond the range and YES! , s[5]=’a’ and s[1]=’a’ match

So p[3]++ so it becomes 3.

So we update the range, C(centre)=index 3,l=1,r=5

Index : 0123456

-> New Range : b**ababa**d

-> Now at index 4, we are inside range, so copy from mirror,p[4]=p[2]=2 which is wrong,bcoz palindrome are symmetric only till the range,and the 2nd character to left of ‘b’(at index 2) which is at index 0 is outside the range and if you see carefully ‘b’ at index 0 and ‘d’ at index 6 don’t match so can’t expand outside the range,So if by copying p[mirror],if we are going outside the range on left , we just update p[i] **r-i**=5-4=1 , and it’s true only 1 character around ‘b’ match that is ‘a’.

So finally we come to condition :

if(i<r)

p[i] = min(p[mirror],r-i);

-> In actual code ,we put ‘#’ around all characters so that we can even expand around even length palindromes.

E.g : #a#b#b#a , here centre is ‘#’ only then we can expand.

-> Also we put a ‘$’ at start of string and a ‘@’ at the end(in short 2 different characters which are unique),so that in worst case we don’t expand beyond the string length in worst case.

**Code :** [**https://ideone.com/P5eTPb**](https://ideone.com/P5eTPb)