Question: https://leetcode.com/problems/longest-palindromic-substring/

We use DP to solve this problem.

But HOW?

Think carefully when can a string be palindromic?

->If its first and last elements are same and the substring after excluding the first and last character is a palindrome.(1)

So here we got our sub-problem.

Now we will use a matrix where dp[i][j] represents substring from ith pos to jth pos.

Now all single characters are palindromes, thus substring with a length 1 are substring by default.

And substrings of length 2 are substring if both characters are same.

And for rest of lengths we can use the sub problem which we discussed earlier (1).

Now we need longest so the palindromic substring with longest length is the substring.

We have used gap strategy to travel through dp matrix.

Code:  
class Solution {

public String longestPalindrome(String s) {

int n=s.length(), dp[][] = new int[n][n], max=0, start=0, end=0;

for(int z=0; z<n; z++){

int i=0, j=z;

while(i<n && j<n){

if(i==j){

dp[i][j] = 1;

}

else if(j-i==1){

if(s.charAt(j)==s.charAt(i)){

dp[i][j]=1;

}

else{

dp[i][j]=0;

}

}

else{

if(s.charAt(j)==s.charAt(i) && dp[i+1][j-1]==1){

dp[i][j]=1;

}

else{

dp[i][j]=0;

}

}

if(dp[i][j]==1){

if(max<j-i){

max=j-i;

start=i;

end=j;

}

}

i++;

j++;

}

}

return s.substring(start, end+1);

}

}

Github Link :<https://lnkd.in/ecwtJeaz>