Yash Pradhan

leet code 5: Longest Palindromic Substring.

Notes:

> Base Cases:

length 1 : True; is a palindrome

length 2: [i.j] True if str[i] == str[j];
False otherwise.

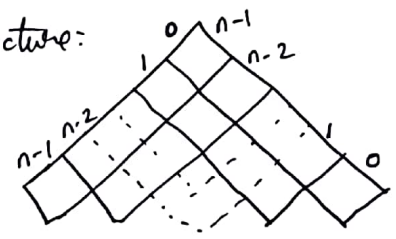
Palindrome expands around center

str[i....j] is a palindrome
iff str[i+1....j-1] is a palindrome
and str[i] == str[j].

Structure to solve this problem.

starting from len 1 i.e. str[i....i]
to length 2; str[i..i+1]
to length n; str[0....n-1]

proposed structure:



Fill direction.

Recurrence Relation:

$$isPal(i,j) = \begin{cases} True ; \text{ if } i == j \\ True ; \text{ if } j == i+1 \text{ and } str[i] == str[j] \\ False ; \text{ otherwise.} \\ isPal(i+1, j-1) \quad // len > 2 \\ \text{and } str[i] == str[j] \end{cases}$$

Pseudo Code

```
for i ← 0 to n-1
    tab[i][i] ← T
for i ← 0 to n-2
    tab[i][i+1] ← T
        iff s[i] == s[i+1]
for K ← 3 to n
    for i ← 0 to n-K
        j ← i+K-1
        tab[i][j] = T
        iff tab[i+1][j-1]
        && s[i] == s[j]
```

Core Idea:

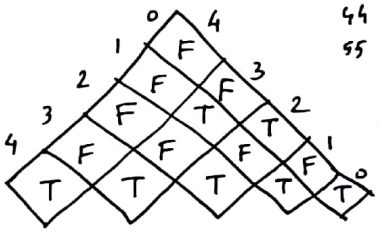
x a b a y
i i+1 j-1 j

pal(i+1, j-1) ??
&& x == y ??

str: babad
0 1 2 3 4

K →

len	1	2	3	4	5
	00	01	02	03	04
	11	12	13	14	
	22	23	24		
	33	34			
	44				
	55				



acaabcacd

table[0..2] = table[1..1] && str[0] == str[2]
table[1..3] = table[2..2] && str[1] == str[3]
table[2..4] = table[3..3] && str[2] == str[4]

table[0.3] = table[1..2] && str[0] == str[3]
...

then a scan of matrix to determine longest palindrome substring