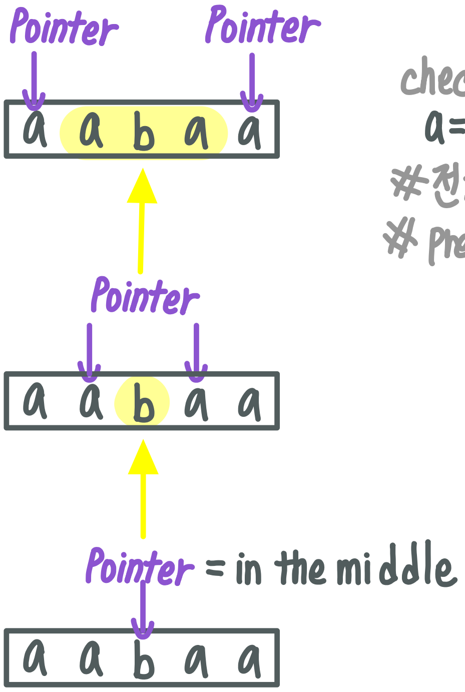
first of all. What is the Palindromic

Palindrome means if you read from the starting or from the end
It's going to read the same. means from right to left or left to right

1. check whether the entire string is a Palindrome or not.
 - if the entire string is a pailndrome,

2. move two Pointers



check if the last characters are equal or not
a==a

전체 : 가운데는 '이미' Pailndrom

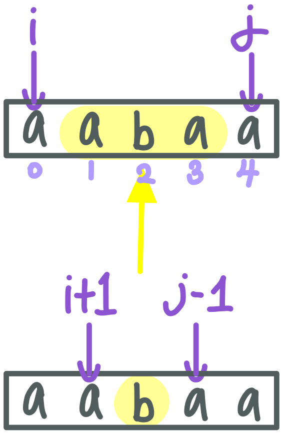
Pre-condition : ~~in the middle range of~~ inner string already computed is a pailndrom already.

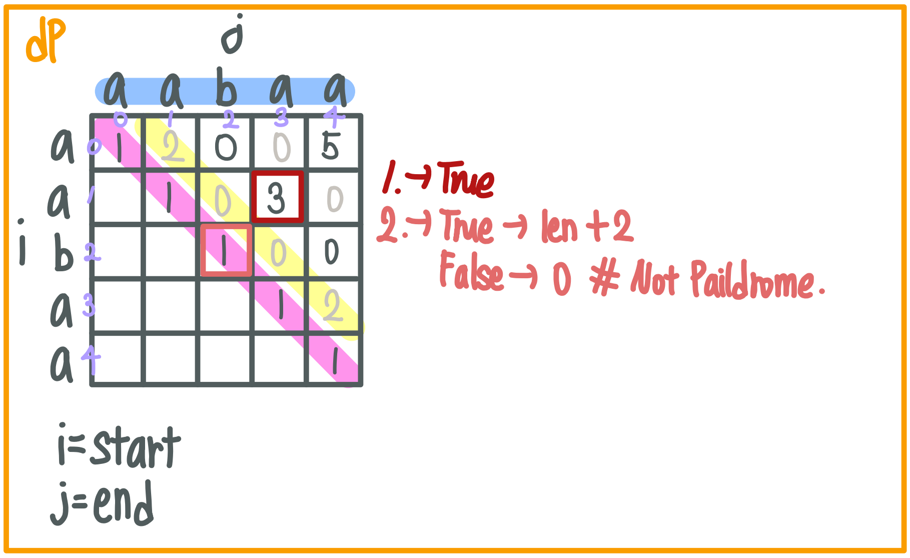
• Represent as DP

1. if $\text{String}[\overset{0}{i}]==\text{String}[\overset{4}{j}]$

2. $DP[\underset{0}{i}][\underset{4}{j}] = \underline{DP[i+1][j-1]}+2$ #length

return must be "True"





dp

i \ j	a 0	a 1	b 2	a 3	a 4
a 0	1	2	0	0	5
a 1		1	0	3	0
b 2			1	0	0
a 3				1	2
a 4					1

1. → True

2. → True → len + 2
   False → 0 # Not Paildrome.

i=start
j=end

- implement code

```
//cpp
int n=s.size();
vector<vector<int>> dp(n,vector<int>(n,0));

string ans;
int maxlength=0;

for (int diff = 0; diff <n; diff ++)
{
  for (int i=0, j=i+diff; j<n; i++, j++)
{
 if(i==j)
  {
    dp[i][j] = 1;
  {
  }
 else if (diff==1)
  {
      dp[i][j]=(s[i]== s[j])? 2:0;
  {
    else
    {
        if(s[i]==s[j] && dp[i+1][j-1]) ★
            dp[i][j]=dp[i+1][j-1]+2;
    }


    if(dp[i][j])
    {     'aba' → 2-0+1 = 3
          끝index - 시작인덱스 + 1 = Pailndrome 길이
      if(j-i+1 >maxlength)
      {
        maxlength=j-i+1;
        ans=s.substr(i,maxlength);
                Cut string   from start_index, to length
      }
    }
  }
}
return ans;
```