Visualization of the solution of

516. Longest Palindromic Subsequence

Medium

Given a string s, find the longest palindromic subsequence's length in s. You may assume that the maximum length of s is 1000.

https://leetcode.com/problems/longest-palindromic-subsequence/

Slides credit to @BryanBo-Cao

Solution author @花花酱

http://zxi.mytechroad.com/blog/dynamic-programming/leetcode-516-longest-palindromic-subsequence/

516. Longest Palindromic Subsequence

```
Solution 1: DP

dp[i][j]:= sol of s[i..j]
Stage: length of the substring

for len = 1 to n:
  for i = 0 to n - len:
    j = i + l - 1
    if s[i] == s[j]:
      dp[i][j] = dp[i + 1][j - 1] + 2
    else:
       dp[i][j] = max(dp[i + 1][j],
                      dp[i][j - 1])

Ans: dp[0][n-1]

Time complexity: O(n^2)
Space complexity: O(n^2)  86 ms
                  -> O(n)    28 ms
```

```
Base case:
  a -> dp[i][i] = 1

case 1: s[i] == s[j]
  a*****a -> dp[i][j]
             = dp[i+1][j-1] + 2

case 2: s[i] != s[j]
  ab****b dp[i][j] = dp[i+1][j]
  a****ab dp[i][j] = dp[i][j-1]
  ^     ^
  i     j
```


http://zxi.mytechroad.com/blog/dynamic-programming/leetcode-516-longest-palindromic-subsequence/

https://www.youtube.com/watch?v=OZX1nqaQ_9M

Example:

c b b s a c
0 1 2 3 4 5

return 4
(len of “cbbc”)

Initialize the table, note that blank means 0

j

i

	0	1	2	3	4	5
0						
1						
2						
3						
4						
5						

c	b	b	s	a	c
0	1	2	3	4	5
i,j					

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n)
i: from 0 to 5 (n - len)
j: i + len - 1

crrnt_v: 1
crrnt_v: 0
crrnt_v: 0

Note: crrnt_v refers to current value

	0	1	2	3	4	5
0						
1						
2						
3						
4						
5						

c	b	b	s	a	c
0	1	2	3	4	5
i,j					

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n)
i: from 0 to 5 (n - len)
j: i + len - 1
since i == j
dp[0][0] = 1

crrnt_v: 1
crrnt_v: 0
crrnt_v: 0

	0	1	2	3	4	5
0	1					
1						
2						
3						
4						
5						

c	b	b	s	a	c
0	1	2	3	4	5
	i,j				

```
1  class Solution {
2  public:
3      int longestPalindromeSubseq(string s) {
4          const int n = s.length();
5          vector<vector<int>> dp(n, vector<int>(n, 0));
6          for (int len = 1; len <= n; ++len) {
7              for (int i = 0; i <= n - len; ++i) {
8                  int j = i + len - 1;
9                  if (i == j) {
10                     dp[i][j] = 1;
11                     continue;
12                 }
13                 dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
14                 if (s[i] == s[j])
15                 dp[i][j] = dp[i + 1][j - 1] + 2;
16             }
17         }
18         return dp[0][n - 1];
19     }
20 };
```

len: from 1 to 6 (n)
i: from 0 to 5 (n - len)
j: i + len - 1

crrnt_v: 1
crrnt_v: 1
crrnt_v: 1

	0	1	2	3	4	5
0	1					
1						
2						
3						
4						
5						

c	b	b	s	a	c
0	1	2	3	4	5
	i,j				

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n)
i: from 0 to 5 (n - len)
j: i + len - 1
since i == j
dp[1][1] = 1

crrnt_v: 1
crrnt_v: 1
crrnt_v: 1

	0	1	2	3	4	5
0	1					
1		1				
2						
3						
4						
5						

c	b	b	s	a	c
0	1	2	3	4	5
		i,j			

len: from 1 to 6 (n)
i: from 0 to 5 (n - len)
j: i + len - 1

crrnt_v: 1
crrnt_v: 2
crrnt_v: 2

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

	0	1	2	3	4	5
0	1					
1		1				
2						
3						
4						
5						

c	b	b	s	a	c
0	1	2	3	4	5
		i,j			

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n)
i: from 0 to 5 (n - len)
j: i + len - 1
since i == j
dp[2][2] = 1

crrnt_v: 1
crrnt_v: 2
crrnt_v: 2

	0	1	2	3	4	5
0	1					
1		1				
2			1			
3						
4						
5						

c	b	b	s	a	c
0	1	2	3	4	5
			i,j		

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n)
i: from 0 to 5 (n - len)
j: i + len - 1

crrnt_v: 1
crrnt_v: 3
crrnt_v: 3

	0	1	2	3	4	5
0	1					
1		1				
2			1			
3						
4						
5						

c	b	b	s	a	c
0	1	2	3	4	5
			i,j		

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n)
i: from 0 to 5 (n - len)
j: i + len - 1
since i == j
dp[3][3] = 1

crrnt_v: 1
crrnt_v: 3
crrnt_v: 3

	0	1	2	3	4	5
0	1					
1		1				
2			1			
3				1		
4						
5						

c	b	b	s	a	c
0	1	2	3	4	5
				i,j	

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n)
i: from 0 to 5 (n - len)
j: i + len - 1

crrnt_v: 1
crrnt_v: 4
crrnt_v: 4

	0	1	2	3	4	5
0	1					
1		1				
2			1			
3				1		
4						
5						

c	b	b	s	a	c
0	1	2	3	4	5
				i,j	

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n)
i: from 0 to 5 (n - len)
j: i + len - 1
since i == j
dp[4][4] = 1

crrnt_v: 1
crrnt_v: 4
crrnt_v: 4

	0	1	2	3	4	5
0	1					
1		1				
2			1			
3				1		
4					1	
5						

c	b	b	s	a	c
0	1	2	3	4	5
					i,j

len: from 1 to 6 (n)
i: from 0 to 5 (n - len)
j: i + len - 1

crrnt_v: 1
crrnt_v: 5
crrnt_v: 5

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

	0	1	2	3	4	5
0	1					
1		1				
2			1			
3				1		
4					1	
5						

c	b	b	s	a	c
0	1	2	3	4	5
					i,j

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n)
i: from 0 to 5 (n - len)
j: i + len - 1
since i == j
dp[5][5] = 1

crrnt_v: 1
crrnt_v: 5
crrnt_v: 5

	0	1	2	3	4	5
0	1					
1		1				
2			1			
3				1		
4					1	
5						1

c	b	b	s	a	c
0	1	2	3	4	5
i	j				

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n)
i: from 0 to 4 (n - len)
j: i + len - 1

crrnt_v: 2
crrnt_v: 0
crrnt_v: 1

	0	1	2	3	4	5
0	1					
1		1				
2			1			
3				1		
4					1	
5						1

c	b	b	s	a	c
0	1	2	3	4	5
i	j				

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n)
i: from 0 to 4 (n - len)
j: i + len - 1
since i != j

crrnt_v: 2
crrnt_v: 0
crrnt_v: 1

dp[0][1] = max(dp[1][1], dp[0][0])
= max(1, 1) = 1

	0	1	2	3	4	5
0	1	1				
1		1				
2			1			
3				1		
4					1	
5						1

c	b	b	s	a	c
0	1	2	3	4	5
	i	j			

len: from 1 to 6 (n)
i: from 0 to 4 (n - len)
j: i + len - 1

crrnt_v: 2
crrnt_v: 1
crrnt_v: 2

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

	0	1	2	3	4	5
0	1	1				
1		1				
2			1			
3				1		
4					1	
5						1

c	b	b	s	a	c
0	1	2	3	4	5
	i	j			

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n) crrnt_v: 2
i: from 0 to 4 (n - len) crrnt_v: 1
j: i + len - 1 crrnt_v: 2
since s[i] == s[j] (== 'b')
dp[1][2] = dp[2][1] + 2 = 2

	0	1	2	3	4	5
0	1	1				
1		1	2			
2		0	1			
3				1		
4					1	
5						1

c	b	b	s	a	c
0	1	2	3	4	5
		i	j		

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n)
i: from 0 to 4 (n - len)
j: i + len - 1

crrnt_v: 2
crrnt_v: 2
crrnt_v: 3

	0	1	2	3	4	5
0	1	1				
1		1	2			
2			1			
3				1		
4					1	
5						1

c	b	b	s	a	c
0	1	2	3	4	5
		i	j		

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n)
i: from 0 to 4 (n - len)
j: i + len - 1
since i != j

crrnt_v: 2
crrnt_v: 2
crrnt_v: 3

dp[2][3] = max(dp[3][3], dp[2][2])
= max(1, 1) = 1

	0	1	2	3	4	5
0	1	1				
1		1	2			
2			1	1		
3				1		
4					1	
5						1

c	b	b	s	a	c
0	1	2	3	4	5
			i	j	

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n)
i: from 0 to 4 (n - len)
j: i + len - 1

crrnt_v: 2
crrnt_v: 3
crrnt_v: 4

	0	1	2	3	4	5
0	1	1				
1		1	2			
2			1	1		
3				1		
4					1	
5						1

c	b	b	s	a	c
0	1	2	3	4	5
			i	j	

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n)
i: from 0 to 4 (n - len)
j: i + len - 1
since i != j

crrnt_v: 2
crrnt_v: 3
crrnt_v: 4

dp[3][4] = max(dp[4][4], dp[3][3])
= max(1, 1) = 1

	0	1	2	3	4	5
0	1	1				
1		1	2			
2			1	1		
3				1	1	
4					1	
5						1

c	b	b	s	a	c
0	1	2	3	4	5
				i	j

len: from 1 to 6 (n)
i: from 0 to 4 (n - len)
j: i + len - 1

crrnt_v: 2
crrnt_v: 4
crrnt_v: 5

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

	0	1	2	3	4	5
0	1	1				
1		1	2			
2			1	1		
3				1	1	
4					1	
5						1

c	b	b	s	a	c
0	1	2	3	4	5
				i	j

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n)
i: from 0 to 4 (n - len)
j: i + len - 1
since i != j

crrnt_v: 2
crrnt_v: 4
crrnt_v: 5

dp[4][5] = max(dp[5][5], dp[4][4])
= max(1, 1) = 1

	0	1	2	3	4	5
0	1	1				
1		1	2			
2			1	1		
3				1	1	
4					1	1
5						1

c	b	b	s	a	c
0	1	2	3	4	5
i		j			

len: from 1 to 6 (n)
i: from 0 to 3 (n - len)
j: i + len - 1

crrnt_v: 3
crrnt_v: 0
crrnt_v: 2

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

	0	1	2	3	4	5
0	1	1				
1		1	2			
2			1	1		
3				1	1	
4					1	1
5						1

c	b	b	s	a	c
0	1	2	3	4	5
i		j			

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n) crrnt_v: 3
i: from 0 to 3 (n - len) crrnt_v: 0
j: i + len - 1 crrnt_v: 2
since i != j

dp[0][2] = max(dp[1][2], dp[0][1])
= max(2, 1) = 2

	0	1	2	3	4	5
0	1	1	2			
1		1	2			
2			1	1		
3				1	1	
4					1	1
5						1

c	b	b	s	a	c
0	1	2	3	4	5
	i		j		

len: from 1 to 6 (n)
i: from 0 to 3 (n - len)
j: i + len - 1

crrnt_v: 3
crrnt_v: 1
crrnt_v: 3

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

	0	1	2	3	4	5
0	1	1	2			
1		1	2			
2			1	1		
3				1	1	
4					1	1
5						1

c	b	b	s	a	c
0	1	2	3	4	5
	i		j		

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n)
i: from 0 to 3 (n - len)
j: i + len - 1
since i != j

crrnt_v: 3
crrnt_v: 1
crrnt_v: 3

dp[1][3] = max(dp[2][3], dp[1][2])
= max(1, 2) = 2

	0	1	2	3	4	5
0	1	1	2			
1		1	2	2		
2			1	1		
3				1	1	
4					1	1
5						1

c	b	b	s	a	c
0	1	2	3	4	5
		i		j	

len: from 1 to 6 (n)
i: from 0 to 3 (n - len)
j: i + len - 1

crrnt_v: 3
crrnt_v: 2
crrnt_v: 4

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

	0	1	2	3	4	5
0	1	1	2			
1		1	2	2		
2			1	1		
3				1	1	
4					1	1
5						1

c	b	b	s	a	c
0	1	2	3	4	5
		i		j	

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n)
i: from 0 to 3 (n - len)
j: i + len - 1
since i != j

crrnt_v: 3
crrnt_v: 2
crrnt_v: 4

dp[2][4] = max(dp[3][4], dp[2][3])
= max(1, 1) = 1

	0	1	2	3	4	5
0	1	1	2			
1		1	2	2		
2			1	1	1	
3				1	1	
4					1	1
5						1

c	b	b	s	a	c
0	1	2	3	4	5
			i		j

len: from 1 to 6 (n)
i: from 0 to 3 (n - len)
j: i + len - 1

crrnt_v: 3
crrnt_v: 3
crrnt_v: 5

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

	0	1	2	3	4	5
0	1	1	2			
1		1	2	2		
2			1	1	1	
3				1	1	
4					1	1
5						1

c	b	b	s	a	c
0	1	2	3	4	5
			i		j

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n) crrnt_v: 3
i: from 0 to 3 (n - len) crrnt_v: 3
j: i + len - 1 crrnt_v: 5
since i != j

dp[3][5] = max(dp[4][5], dp[3][4])
= max(1, 1) = 1

	0	1	2	3	4	5
0	1	1	2			
1		1	2	2		
2			1	1	1	
3				1	1	1
4					1	1
5						1

c	b	b	s	a	c
0	1	2	3	4	5
i			j		

len: from 1 to 6 (n)
i: from 0 to 2 (n - len)
j: i + len - 1

crrnt_v: 4
crrnt_v: 0
crrnt_v: 3

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

	0	1	2	3	4	5
0	1	1	2			
1		1	2	2		
2			1	1	1	
3				1	1	1
4					1	1
5						1

c	b	b	s	a	c
0	1	2	3	4	5
i			j		

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n)
i: from 0 to 2 (n - len)
j: i + len - 1
since i != j

crrnt_v: 4
crrnt_v: 0
crrnt_v: 3

dp[0][3] = max(dp[1][3], dp[0][2])
= max(2, 2) = 2

	0	1	2	3	4	5
0	1	1	2	2		
1		1	2	2		
2			1	1	1	
3				1	1	1
4					1	1
5						1

c	b	b	s	a	c
0	1	2	3	4	5
	i			j	

len: from 1 to 6 (n)
i: from 0 to 2 (n - len)
j: i + len - 1

crrnt_v: 4
crrnt_v: 1
crrnt_v: 4

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

	0	1	2	3	4	5
0	1	1	2	2		
1		1	2	2		
2			1	1	1	
3				1	1	1
4					1	1
5						1

c	b	b	s	a	c
0	1	2	3	4	5
	i			j	

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n)
i: from 0 to 2 (n - len)
j: i + len - 1
since i != j

crrnt_v: 4
crrnt_v: 1
crrnt_v: 4

dp[1][4] = max(dp[2][4], dp[1][3])
= max(1, 2) = 2

	0	1	2	3	4	5
0	1	1	2	2		
1		1	2	2	2	
2			1	1	1	
3				1	1	1
4					1	1
5						1

c	b	b	s	a	c
0	1	2	3	4	5
		i			j

len: from 1 to 6 (n)
i: from 0 to 2 (n - len)
j: i + len - 1

crrnt_v: 4
crrnt_v: 2
crrnt_v: 5

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

	0	1	2	3	4	5
0	1	1	2	2		
1		1	2	2	2	
2			1	1	1	
3				1	1	1
4					1	1
5						1

c	b	b	s	a	c
0	1	2	3	4	5
		i			j

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n)
i: from 0 to 2 (n - len)
j: i + len - 1
since i != j

crrnt_v: 4
crrnt_v: 2
crrnt_v: 5

dp[2][5] = max(dp[3][5], dp[2][4])
= max(1, 2) = 2

	0	1	2	3	4	5
0	1	1	2	2		
1		1	2	2	2	
2			1	1	1	1
3				1	1	1
4					1	1
5						1

c	b	b	s	a	c
0	1	2	3	4	5
i				j	

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n)
i: from 0 to 1 (n - len)
j: i + len - 1

crrnt_v: 5
crrnt_v: 0
crrnt_v: 4

	0	1	2	3	4	5
0	1	1	2	2		
1		1	2	2	2	
2			1	1	1	1
3				1	1	1
4					1	1
5						1

c	b	b	s	a	c
0	1	2	3	4	5
i				j	

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n) crrnt_v: 5
i: from 0 to 1 (n - len) crrnt_v: 0
j: i + len - 1 crrnt_v: 4
since i != j

dp[0][4] = max(dp[1][4], dp[0][3])
= max(2, 2) = 2

	0	1	2	3	4	5
0	1	1	2	2	2	
1		1	2	2	2	
2			1	1	1	1
3				1	1	1
4					1	1
5						1

c	b	b	s	a	c
0	1	2	3	4	5
	i				j

len: from 1 to 6 (n)
i: from 0 to 1 (n - len)
j: i + len - 1

crrnt_v: 5
crrnt_v: 1
crrnt_v: 5

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

	0	1	2	3	4	5
0	1	1	2	2	2	
1		1	2	2	2	
2			1	1	1	1
3				1	1	1
4					1	1
5						1

c	b	b	s	a	c
0	1	2	3	4	5
	i				j

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n)
i: from 0 to 1 (n - len)
j: i + len - 1
since i != j

crrnt_v: 5
crrnt_v: 1
crrnt_v: 5

dp[1][5] = max(dp[2][5], dp[1][4])
= max(1, 2) = 2

	0	1	2	3	4	5
0	1	1	2	2	2	
1		1	2	2	2	2
2			1	1	1	1
3				1	1	1
4					1	1
5						1

c	b	b	s	a	c
0	1	2	3	4	5
i					j

len: from 1 to 6 (n)
i: from 0 to 0 (n - len)
j: i + len - 1

crrnt_v: 6
crrnt_v: 0
crrnt_v: 5

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

	0	1	2	3	4	5
0	1	1	2	2	2	
1		1	2	2	2	2
2			1	1	1	1
3				1	1	1
4					1	1
5						1

c	b	b	s	a	c
0	1	2	3	4	5
i					j

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n) crrnt_v: 6
i: from 0 to 0 (n - len) crrnt_v: 0
j: i + len - 1 crrnt_v: 5
since s[i] == s[j] (== 'c')
dp[0][5] = dp[1][4] + 2 = 2 + 2 = 4

	0	1	2	3	4	5
0	1	1	2	2	2	4
1		1	2	2	2	2
2			1	1	1	1
3				1	1	1
4					1	1
5						1

c	b	b	s	a	c
0	1	2	3	4	5
i					j

```cpp
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        const int n = s.length();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for (int len = 1; len <= n; ++len) {
            for (int i = 0; i <= n - len; ++i) {
                int j = i + len - 1;
                if (i == j) {
                    dp[i][j] = 1;
                    continue;
                }
                dp[i][j] = max(dp[i + 1][j], dp[i][j - 1]);
                if (s[i] == s[j])
                dp[i][j] = dp[i + 1][j - 1] + 2;
            }
        }
        return dp[0][n - 1];
    }
};
```

len: from 1 to 6 (n)
i: from 0 to 0 (n - len)
j: i + len - 1

crrnt_v: 6
crrnt_v: 0
crrnt_v: 5

return 4 (dp[0][5])

	0	1	2	3	4	5
0	1	1	2	2	2	4
1		1	2	2	2	2
2			1	1	1	1
3				1	1	1
4					1	1
5						1

Reference

https://docs.google.com/presentation/d/1KhxVVgI8jzc-g7unDNKFiHY6XDNVSK6LNsadxB14K3U/edit?usp=sharing

https://leetcode.com/problems/longest-palindromic-subsequence/

http://zxi.mytechroad.com/blog/dynamic-programming/leetcode-516-longest-palindromic-subsequence/

https://www.youtube.com/watch?v=jOkE4X-PWOI&feature=youtu.be