Palindromic Subsequence Pattern

This pattern helps solve problems involving **palindromic (reads same forwards and backwards) subsequences or substrings** in a string, often using **dynamic programming**.

Common Questions:

- What is the length of the **Longest Palindromic Subsequence** in a string?
- What is the minimum number of deletions to make a string a palindrome?
- What is the length of the **Longest Palindromic Substring**? (very similar!)

PALINDROMIC SUBSEQUENCE PATTERN: LONGEST PALINDROMIC SUBSEQUENCE

Input String: Find ab dennth of he longess subsequence that is a pandemone. / alinrod.

1. **Start:** Find the longest subsequence (not necessarily continuous) that reads the same forwards obewads 1

2. **Goal:** Starts DP table (seqrussing indices [i...j).

b	b	a	b	c	b	c

1 b a b c c b

Every single character is a palindrom of length 1

3. **Step 2:** Use a DP table :::| magt annsers for bigger substrings using smaller ones!

When s[i] = S[j] → 2 + dp[i+1][j-1]

Δ Vhax c[i pi[|j], i[j], dp (i|j-1))

Bulca answers for bigger substrings usslore✓

	0	1	2	3	4	5	6
0	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1
6	1	1	1	1	1	1	7

Length of Longest Palindromic Subsequence = 7

BONUS TIP This pattern is great for making strings palindromic with fewest edits, or finding

palindromic subsequences in DNA/texts!

Classic Problem: Longest Palindromic Subsequence

Problem Statement:
Given a string, find the length of the longest subsequence that is a palindrome.
(*A subsequence can skip some characters, but order must be kept*)

Example:
Input: `"bbabcbcab"`
Output: `7` (the longest palindromic subsequence is `"babcbab"`)

Explanation

1. **Subproblems:**
   For every substring from index `i` to `j`, what is the longest palindromic subsequence?
2. **Base Case:**
   - If `i == j`, a single character is always a palindrome of length 1.
3. **Build Up:**
   - If characters at `i` and `j` are **same**:
     Length = 2 + length from `i+1` to `j-1`.
   - Else:
     Take max from `i+1` to `j` and `i` to `j-1`.
4. **Use a DP table** to avoid recalculating for overlapping subproblems.

C Code Example: Longest Palindromic Subsequence

```
#include <stdio.h>
#include <string.h>

// Returns length of LPS in s[0...n-1]
int longestPalindromeSubseq(char *s) {
    int n = strlen(s);
    int dp[n][n];

    // All single letters are palindromes of length 1
    for (int i = 0; i < n; i++)
        dp[i][i] = 1;

    // Build the table for substrings of length 2 to n
    for (int len = 2; len <= n; len++) {
        for (int i = 0; i <= n - len; i++) {
            int j = i + len - 1;
            if (s[i] == s[j]) {
                if (len == 2)
                    dp[i][j] = 2;
                else
                    dp[i][j] = 2 + dp[i + 1][j - 1];
            } else {
                dp[i][j] = (dp[i + 1][j] > dp[i][j - 1]) ? dp[i + 1][j] : dp[i][j - 1];
            }
        }
    }
```

```
    return dp[0][n - 1];
}

int main() {
    char s[] = "bbabcbcab";
    printf("Length of Longest Palindromic Subsequence: %d\n", longestPalindromeSubseq(s));
    return 0;
}
```

How does this work?

- **dp[i][j]**: length of LPS in substring `s[i...j]`
- If ends match, expand inward and add 2.
- Else, take the longer between skipping left or right character.
- The table is built up from short substrings to long.

Where is this pattern useful?

- Making a string palindrome with minimum deletions
- Finding palindromic subsequences in DNA, texts, etc.
- Palindrome-based DP problems (substrings, subsequences)

Practice Challenge

- Try finding the **minimum number of deletions** to make the string a palindrome
  (Hint: `n - LPS length`)
- Try the similar problem: **Longest Palindromic Substring** (slight tweak)