Recurrence (the code you used)

- If `i == j` → `dp[i][j] = 1`
- Else if `s[i] == s[j]` → `dp[i][j] = dp[i+1][j] + dp[i][j-1] + 1`
- Else → `dp[i][j] = dp[i+1][j] + dp[i][j-1] - dp[i+1][j-1]`

We fill `dp` by increasing `gap = j - i` from `0` to `n-1`.

Dry run for `s = "abca"` (a b c a)

We will show the `dp` matrix after filling each `gap`. Cells where `j < i` are irrelevant (I'll mark `–`).

Initial (all zeros):

```ini
    j=0 1 2 3
i=0 [ 0 0 0 0 ]
i=1 [ 0 0 0 0 ]
i=2 [ 0 0 0 0 ]
i=3 [ 0 0 0 0 ]
```

gap = 0 (substrings of length 1: single chars)

For every `i`, `dp[i][i] = 1`.

```ini
    j=0 1 2 3
i=0 [ 1 – – – ]   // "a"
i=1 [ – 1 – – ]   // "b"
i=2 [ – – 1 – ]   // "c"
i=3 [ – – – 1 ]   // "a"
```

(We use `–` where j<i.)

gap = 1 (substrings length 2)

Cell by cell:

- `dp[0][1]` for substring `"ab"`: `s[0] != s[1]`
  `dp[0][1] = dp[1][1] + dp[0][0] - dp[1][0] = 1 + 1 - 0 = 2`
  (palindromic subseqs: `"a"`, `"b"`)
- `dp[1][2]` for `"bc"`: `= 1 + 1 - 0 = 2` (`"b"`, `"c"`)
- `dp[2][3]` for `"ca"`: `= 1 + 1 - 0 = 2` (`"c"`, `"a"`)

Matrix after gap=1:

```ini
    j=0 1 2 3
i=0 [ 1 2 – – ]
i=1 [ – 1 2 – ]
i=2 [ – – 1 2 ]
i=3 [ – – – 1 ]
```

gap = 2 (substrings length 3)

- `dp[0][2]` for `"abc"`: `s[0] != s[2]`
  `dp[0][2] = dp[1][2] + dp[0][1] - dp[1][1] = 2 + 2 - 1 = 3`
  (palindromic subseqs: `"a"`, `"b"`, `"c"`)
- `dp[1][3]` for `"bca"`: `s[1] != s[3]`
  `dp[1][3] = dp[2][3] + dp[1][2] - dp[2][2] = 2 + 2 - 1 = 3`
  (palindromic subseqs: `"b"`, `"c"`, `"a"` [the last a])

Matrix after gap=2:

```ini
    j=0 1 2 3
i=0 [ 1 2 3 – ]
i=1 [ – 1 2 3 ]
i=2 [ – – 1 2 ]
i=3 [ – – – 1 ]
```

gap = 3 (whole string length 4: `i=0, j=3`, substring `"abca"`)

- `dp[0][3]`: here `s[0] == s[3]` (`'a' == 'a'`), so use the matching formula:

  ```markdown
  dp[0][3] = dp[1][3] + dp[0][2] + 1
           = 3 + 3 + 1
           = 7
  ```

  Intuition: everything counted in `s[1..3]` plus everything in `s[0..2]` plus **1** for the new palindromes created by pairing the matching ends (and counts of subsequences that include both ends are implicitly covered by the +1 rule here).

Final matrix:

```ini
    j=0 1 2 3
i=0 [ 1 2 3 7 ]
i=1 [ – 1 2 3 ]
i=2 [ – – 1 2 ]
i=3 [ – – – 1 ]
```

Validate final result `dp[0][3] = 7`

List palindromic subsequences (counting by positions, not unique strings):

- Length 1: positions 0:'a', 1:'b', 2:'c', 3:'a' → **4**
- Length 2: `"aa"` from positions (0,3) → **1**
- Length 3: `"aba"` (0,1,3) and `"aca"` (0,2,3) → **2**
- No length 4 palindrome.

Total = 4 + 1 + 2 = **7** → matches `dp[0][3]`.

Quick tips for reading/filling the table in future

1. **Fill by gap** (shorter substrings first). Each `dp[i][j]` only depends on `dp[i+1][j]`, `dp[i][j-1]`, and `dp[i+1][j-1]` — all of which are computed for smaller gaps.
2. When `s[i] == s[j]` you **add 1** (this accounts for new palindromes that use both ends).
3. When `s[i] != s[j]` you **subtract** `dp[i+1][j-1]` to remove double-counted subsequences.
4. For any `dp[x][y]` with `x>y` treat it as `0` (this happens implicitly in the formula for small substrings).