# Longest Repeating Subsequence :

$\underrightarrow{a_0\ b_1\ c_2}\ \ \underrightarrow{a_3\ b_4\ c_5}$

Repeating subseq.   ③ → Ans.

⊛ If two different subseq. same character is not allowed.

✓~~What?~~ → LCS ( s, s)

↳ longest common subseq.   complete string.

✓~~How?~~

✓~~Why?~~

↦ At the time of equality check. make sure that index is different.

$a_i \neq a_j$  if  $i == j$

Longest c. sub. $(s_1, s_2)$

| $ch1 == ch2$ && $\boxed{i \neq j}$ | $ch1 \neq ch2$ |
|---|---|
| $\text{longest}(r_1, r_2) + 1$ | max. $[\text{longest}(s_1, r_2)\ ,\ \text{longest}(r_1, s_2)]$ |

$S_1 \to ch_1\ (r_1)$

$S_2 \to ch_2\ (r_2)$

$\boxed{c_2\ a_3\ b_4\ c_5}$

$\boxed{\begin{matrix} a_3 \\ b_4 \\ c_5 \end{matrix}}$

$\longrightarrow S_1$

$\downarrow$ S2

| | $a_0$ | $b_1$ | $c_2$ | $a_3$ | $b_4$ | $c_5$ | $-_6$ |
|---|---|---|---|---|---|---|---|
| $a_0$ | 3 | 3 | 3 | 3 | 2 | 1 | 0 |
| $b_1$ | 3 | 2 | 2 | 2 | 2 | 1 | 0 |
| $c_2$ | 3 | 2 | 1 | 1 | 1 | 1 | 0 |
| $a_3$ | 3 | 2 | * 1 | 1 | 1 | 1 | 0 |
| $b_4$ | 2 | 2 | 1 | 1 | 1 | 1 | 0 |
| $c_5$ | 1 | 1 | 1 | 0 | 0 | 0 | 0 $s_1, s_2$ |
| $-_6$ | 0 | 0 | 0 | 0 | 0 | 0 $s_1, r_2$ | 0 |

## Longest common substring :→

Brute force → $l_1^2 * l_2^2$     optimise → $l_1 * l_2$

String →① p q a b c x y → l1 ] → $l_1^2$     String② x y z a b c p → $l_2$ ] → $l_2^2$

Why? [Compare all prefix of str1 to all prefix of str2 and find longest common suffix b/w all comparison.]

p q a b c x y     x y z a b c p

all prefix
- p
- p q
- p q a
- p q a b
- p q a b c
- p q a b c x
- p q a b c x y

- x
- x y
- x y z
- x y z a
- x y z a b
- x y z a b c
- x y z a b c p.

p q a b c ↔ x y z a b c → [longest common suffix → abc]

suffix → O(1)

↑ longest common substring

$s_1 \to$ p q a b c x y

$s_2 \to$ x y z a b c p

longest common suffix

null

| | x | xy | xyz | xyza | xyzab | xyzabc | xyzabcp |
|---|---|---|---|---|---|---|---|
| p | ϕ | ϕ | ϕ | ϕ | ϕ | ϕ | p |
| pq | ϕ | ϕ | ϕ | ϕ | ϕ | ϕ | ϕ |
| pqa | ϕ | ϕ | ϕ | a | ϕ | ϕ | ϕ |
| pqab | ϕ | ϕ | ϕ | ϕ | ab | ϕ | ϕ |
| pqabc | ϕ | ϕ | ϕ | ϕ | ϕ | abc | ϕ |
| pqabcx | x | ϕ | ϕ | ϕ | ϕ | ϕ | ϕ |
| pqabcxy | ϕ | xy | ϕ | ϕ | ϕ | ϕ | ϕ |

abc $\to$ longest common substring.

String $s_1$ → p q a b c x y

String $s_2$ → x y z a b c p

$s_1$ ↓ $r_1$ ch1

$s_2$ ↓ $r_2$ ch2

| | – | x | y | z | a | b | c | p |
|---|---|---|---|---|---|---|---|---|
| – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| p | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| q | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| a | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| b | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 |
| c | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 |
| x | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| y | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 |

| ch1 == ch2 | ch1 ≠ ch2 |
|---|---|
| $(r_1, r_2) + 1$ | 0 |

$s_1$ → $r$ ch1

$s_2$ → $r_2$ ch2

[illegible]

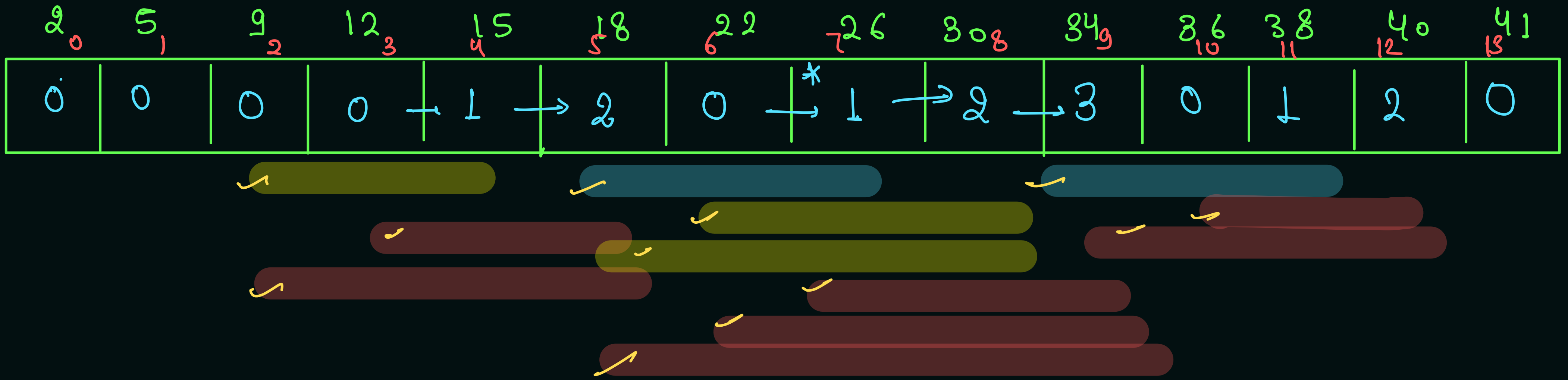
$r_1$ $r_2$

# Arithmatic Slice -1

No. of Subarray which are AP, no. of Elements in subarray greater than or equal to 3.

Total No. of APs → add all Element from DP

↓ $\boxed{12}$



AP ≥ 3

* No. of AP (considering subarrays) ending at current element.

Arithmetic slices 2: No. of APs (from all subseq) with more than 3 Elements. (length ≥ 2)

array →

dp → HashMap[int, int][] dp.

Time - O(n²)

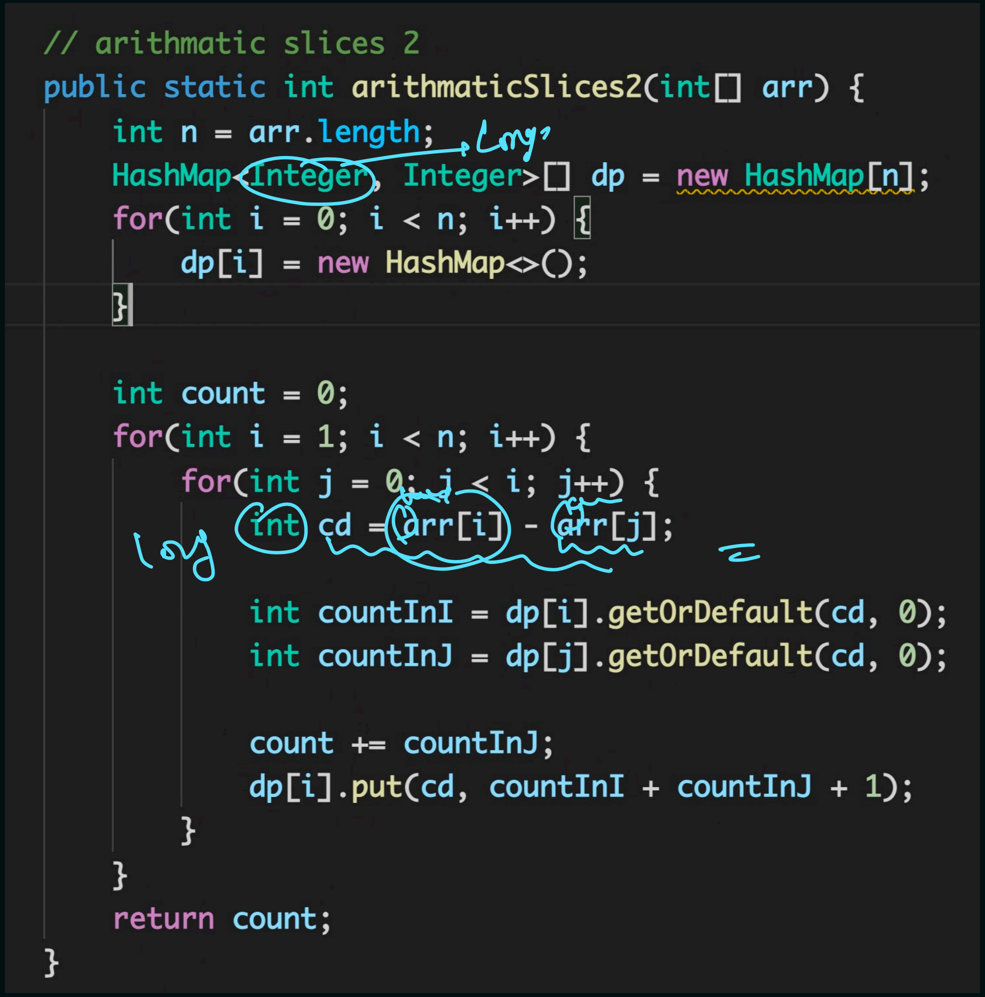
Common diff vs. count of AP ending at current index

j i

| 4 | 2 | 3 | 5 | 2 | 6 | 7 | 9 |
|---|---|---|---|---|---|---|---|
| 0 | -2 1 [4→2] | -1 1 [4→3] | 1 1 [4→5] | -2 1 [4→2] | 2 1 [4→6] | 3 1 [4→7] | 5 1 [4→9] |
| | | 1 1 [2→3] | 3 1 [2→5] | 0 1 [2→2] | 4 ~~1~~ 2 [2→6] [2→6] | 5 ~~1~~ 2 [2→7] [2→7] | 7 ~~1~~ 2 [2→9] [2→9] |
| | | | 2 1 [3→5] | -1 2 [4→3→2, 3→2] | 3 1 [3→6] | 4 1 [3→7] | 6 1 [3→9] |
| | | | | -3 1 [5→2] | 1 2 [4→5→6, 5→6] | 2 2 [3→5→7, 5→7] | 4 1 [5→9] |
| | | | | | | 1 3 [4→5→6→7, 5→6→7, 6→7] | 3 2 [3→6→9, 6→9] |
| | | | | | | | 2 3 [3-5-7-9, 5-7-9, 7-9] |

Result →

1. 4→3→2
2. 4→5–6
3. 3→5→7
4. 4→5→6→7
5. 5-6-7
6. 3→6→9
7. 3-5-7-9
8. 5-7-9

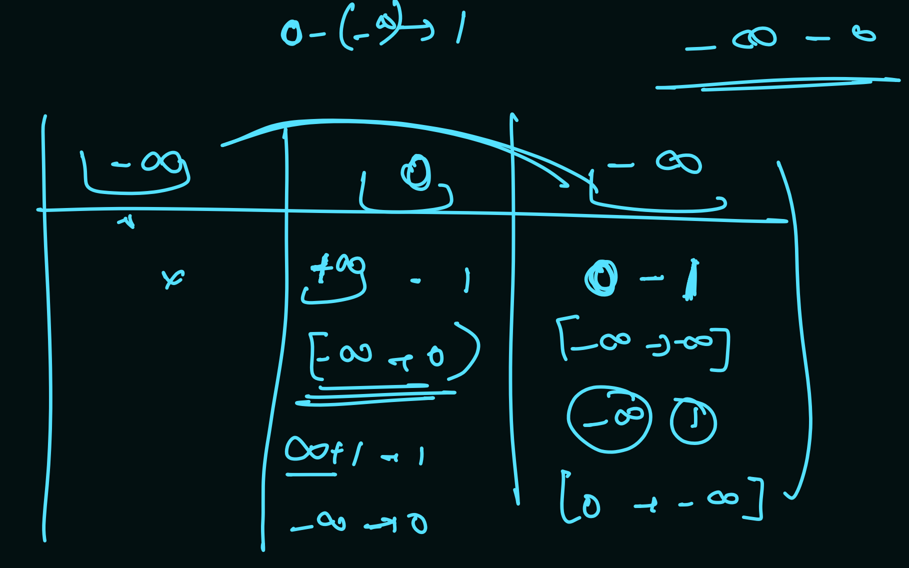


```java
// arithmatic slices 2
public static int arithmaticSlices2(int[] arr) {
    int n = arr.length;
    HashMap<Integer, Integer>[] dp = new HashMap[n];
    for(int i = 0; i < n; i++) {
        dp[i] = new HashMap<>();
    }

    int count = 0;
    for(int i = 1; i < n; i++) {
        for(int j = 0; j < i; j++) {
            int cd = arr[i] - arr[j];

            int countInI = dp[i].getOrDefault(cd, 0);
            int countInJ = dp[j].getOrDefault(cd, 0);

            count += countInJ;
            dp[i].put(cd, countInI + countInJ + 1);
        }
    }
    return count;
}
```



$$\text{long} := 0 - (-\infty)$$

$$+\boxed{\infty + 1}$$