# Count Distinct Subsequences

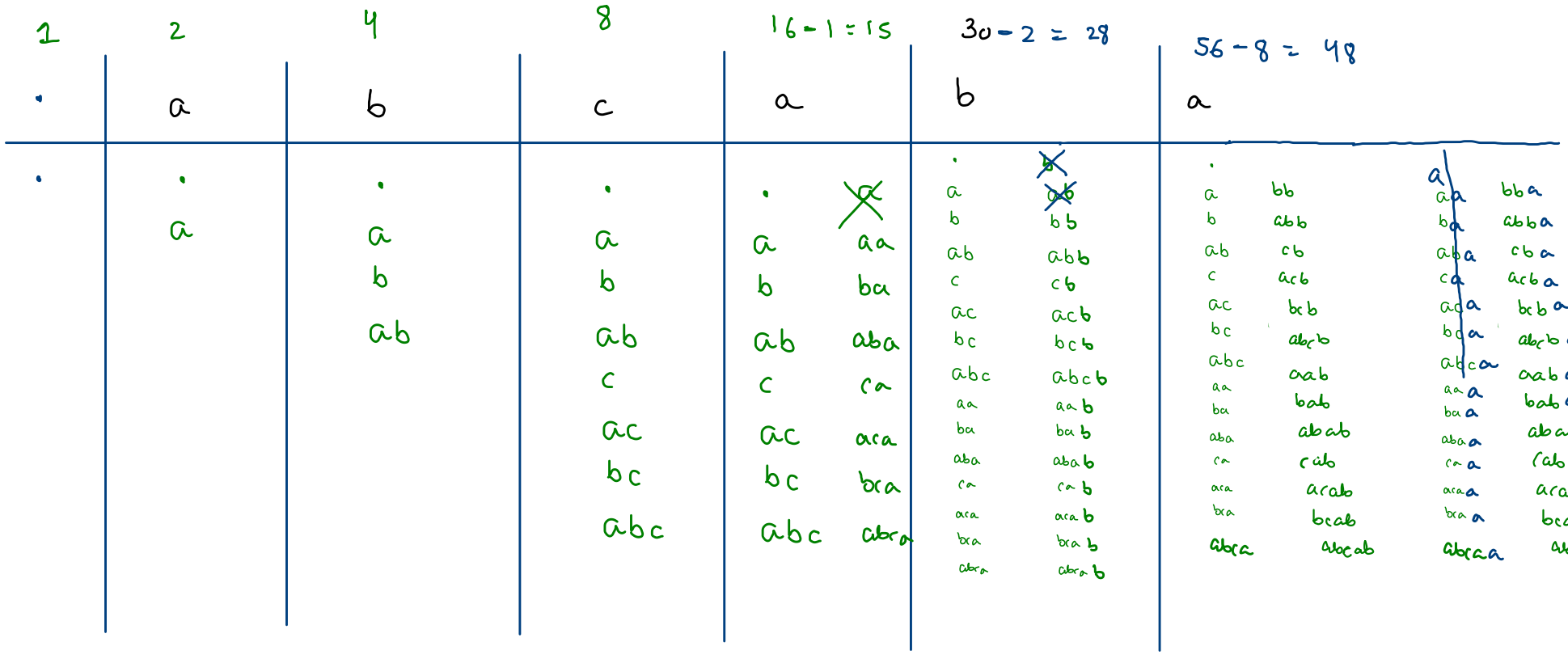
aba

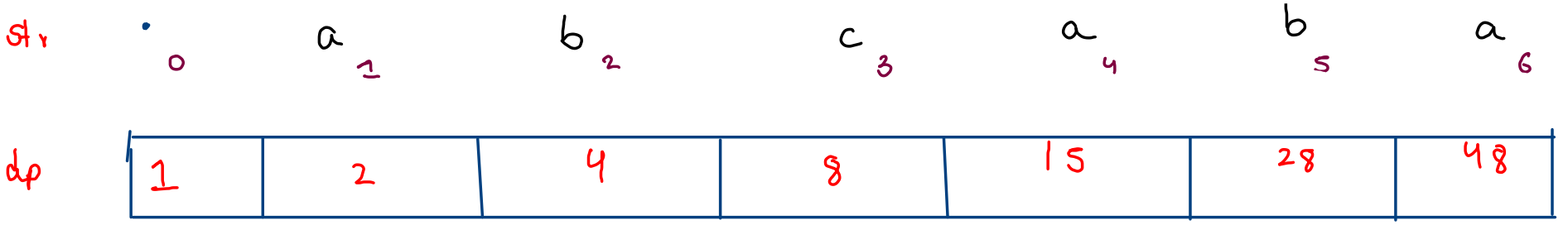
all subseq

- - -
- - a
- b -
- b a
a - -
a - a
a b -
a b a

distinct subseq

- - -
- - a
- b -
- b a
a - a
ab -
a b a


1
2
4
8
16 - 1 = 15
30 - 2 = 28
56 - 8 = 48
a
b
c
a
b
a
a
b
ab
c
ac
bc
abc
aa
ba
aba
ca
aca
bca
abca
bb
abb
cb
acb
bcb
abcb
aab
bab
abab
cab
acab
bcab
abcab
aa
ba
aba
ca
aca
bca
abca
bba
abba
cba
acba
bcba
abcba
aaba
baba
ababa
caba
acaba
bcaba
abcaba




non-empty = 47

a - ~~1~~ 4 6

b - ~~3~~ 5

c - 3

$1_0$

# Count Of Distinct Palindromic Subsequences

Str : a b a b a

c1 m c2

all pal. subseq.

count pal. ss

- s(m) - ①
- s(m) c2 ②
- c1 s(m) - ③
- c1 s(m) c2 ④

| c1 == c2 | c1 != c2 |
| --- | --- |
| ① + ② + ③ + ④ | ① + ② + ③ + ① - ① |
| -> ① + ② + ③ + ① + 1 | cps(mc2) + cps(c1m) - cps(m) |
| cps(mc2) + cps(c1m) + 1 | |

updates these formula for distinct pal. subseq.

④ = ① + 1

c1 != c2

a b c a b d

c1 m c2

ans = cdps(c1 m) + cdps(m c2) - cdps(m)

abcab

a
b
c
aa
bb
aba
aca
bab
bcb

bcabd

~~a~~
~~b~~
~~c~~
~~bb~~
~~bcb~~
~~bab~~
d

bcab

~~b~~
~~c~~
~~a~~
~~bcb~~
~~bab~~
~~bb~~

c1 == c2 ⟶ 3 cases

(i) First case

i j

a m c c m a

c1 m c2

m a m a

c a c a

c c a c c a

m m a m m a

m c m a m c m a

aa

a

(iii) Second case.

i j

a b a b a

c1 m c2

b a b a

a a a a a

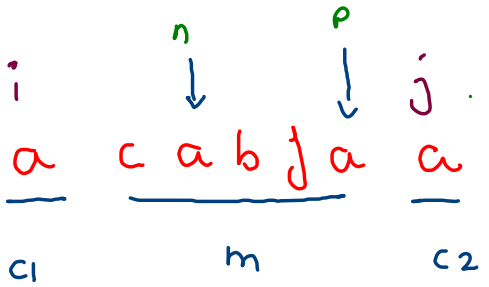
b b a b b a

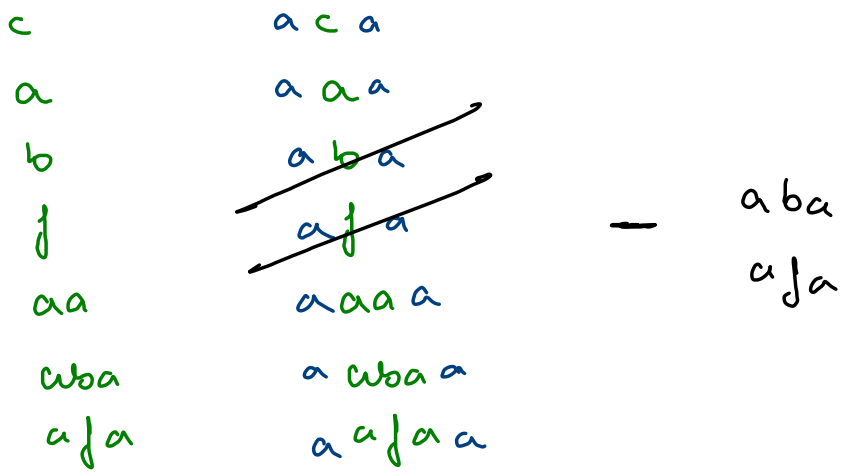
b a b a b a

aa

ans = 2 * dp[i+1][j-1] + 1

(iii) third case




$$ans = 2 * dp[i+1][j-1] - dp[n+1][p-1]$$

next [i] > prev [j] → case I

next [i] == prev [j] → case II

next [i] < prev [j] → case III

a - 6 0
m - 1
p - 4
x - 3
g - 2

next

| 6 | 5 | -1 | -1 | -1 | -1 | -1 |
|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |

just next occ of ith char on right

a m g x p m a
0 1 2 3 4 5 6
i j

prev

| -1 | -1 | -1 | -1 | -1 | 1 | 0 |
|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |

just next occ. of ith char on left

a - 0 6
m - 5
g - 2
x - 3
p - 4

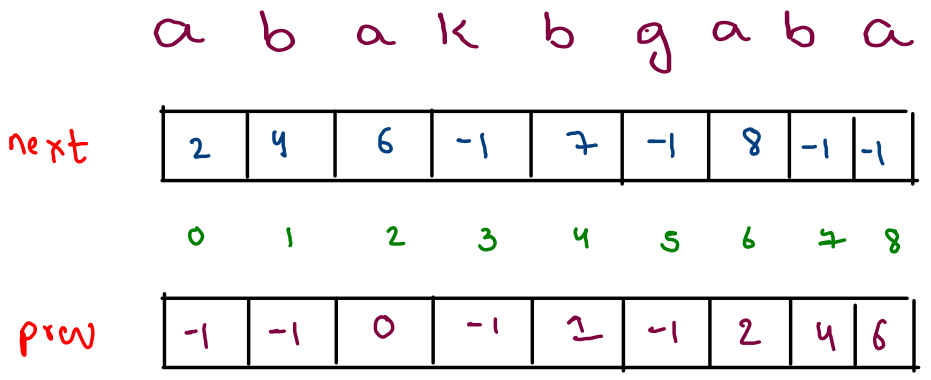
# Preprocessing

```
int[]next = new int[len];
int[]prev = new int[len];
HashMap<Character,Integer>map = new HashMap<>();

//travel from right to left
for(int i=len-1; i >= 0;i--) {
    char ch = str.charAt(i);
    next[i] = map.getOrDefault(ch,-1);
    map.put(ch,i);
}

map.clear();


//travel from left to right
for(int i=0; i < len;i++) {
    char ch = str.charAt(i);
    prev[i] = map.getOrDefault(ch,-1);
    map.put(ch,i);
}
```



a b a k b g a b a

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| next | 2 | 4 | 6 | -1 | 7 | -1 | 8 | -1 | -1 |
| prev | -1 | -1 | 0 | -1 | 1 | -1 | 2 | 4 | 6 |

```java
for(int d = 0; d < len;d++) {
    for(int i = 0, j = d; j < len;i++,j++) {
        if(d == 0) {
           dp[i][j] = 1;
        }
        else if(d == 1) {
            dp[i][j] = 2;
        }
        else {
            if(str.charAt(i) != str.charAt(j)) {
                dp[i][j] = dp[i][j-1] + dp[i+1][j] - dp[i+1][j-1];
            }
            else {
                if(next[i] > prev[j]) {
                    //case 1
                    dp[i][j] = 2*dp[i+1][j-1] + 2;
                }
                else if(next[i] == prev[j]) {
                    //case 2
                    dp[i][j] = 2*dp[i+1][j-1] + 1;
                }
                else {
                    //case 3
                    int n = next[i];
                    int p = prev[j];

                    dp[i][j] = 2*dp[i+1][j-1] - dp[n+1][p-1];
                }
            }
        }
    }

}
```

| | $a_0$ | $b_1$ | $a_2$ | $k_3$ | $b_4$ | $g_5$ | $a_6$ | $b_7$ | $a_8$ |
|---|---|---|---|---|---|---|---|---|---|
| $a_0$ | 1 | 2 | 4 | 5 | 8 | 9 | 15 | 21 | 31 |
| $b_1$ | X | 1 | 2 | 3 | 6 | 7 | 11 | 17 | 21 |
| $a_2$ | X | X | 1 | 2 | 3 | 4 | 8 | 11 | 15 |
| $k_3$ | X | X | X | 1 | 2 | 3 | 4 | 7 | 9 |
| $b_4$ | X | X | X | X | 1 | 2 | 3 | 6 | 8 |
| $g_5$ | X | X | X | X | X | 1 | 2 | 3 | 5 |
| $a_6$ | X | X | X | X | X | X | 1 | 2 | 4 |
| $b_7$ | X | X | X | X | X | X | X | 1 | 2 |
| $a_8$ | X | X | X | X | X | X | X | X | 1 |

modulus

$a + b \rightarrow [(a \% m) + (b \% m)] \% m$

$a * b \rightarrow [(a \% m) * (b \% m)] \% m$

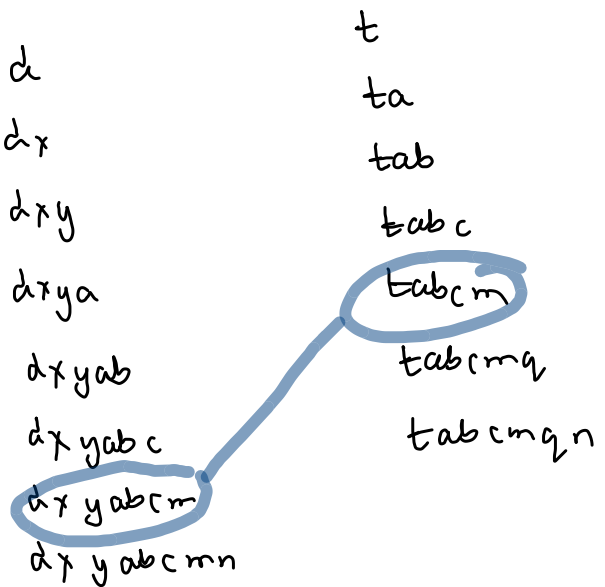
$a - b \rightarrow [(a \% m) - (b \% m) + m] \% m$

# Longest Common Substring

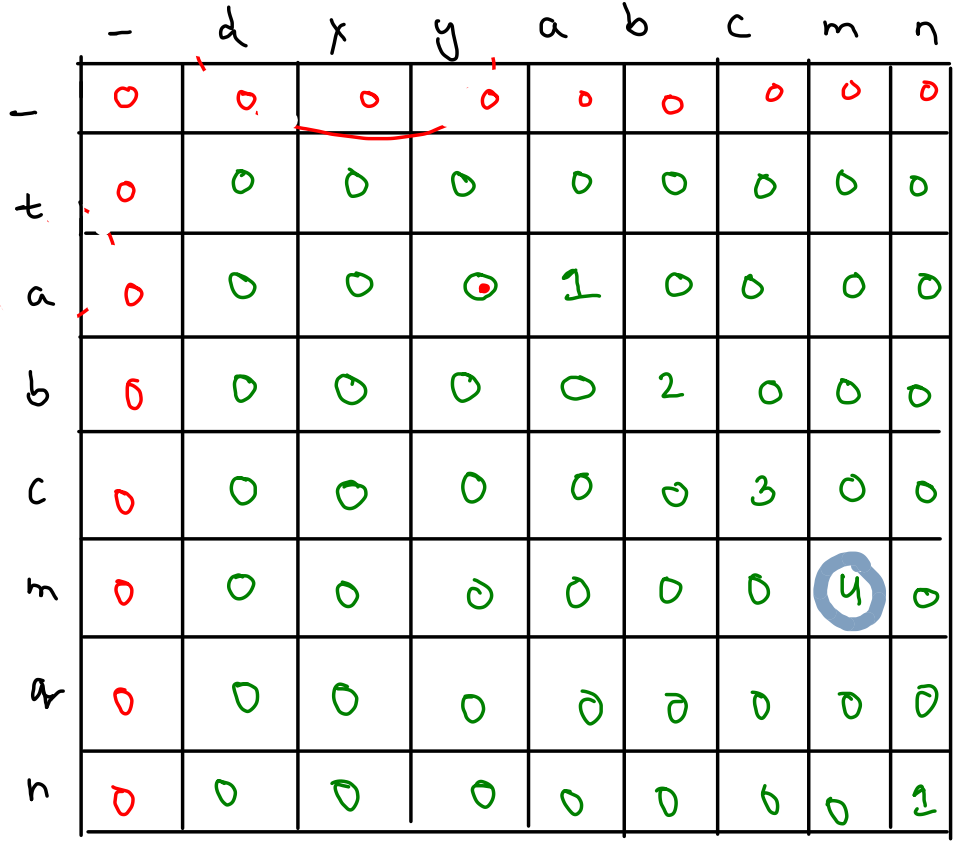
dp[i][j] → longest common suffix of two strings s1.ss (0 to i) and s2.ss (0 to j)

s1 = d x y a b c m n

s2 = t a b c m q n




| | | d | x | y | a | b | c | m | n |
|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| t | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| a | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| b | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 |
| c | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 |
| m | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 |
| q | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| n | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |




s1 s2

r1 c1 r2 c2

LCS

```
if (c1 != c2) {
    dp[i][j] = 0;
}
else {
    dp[i][j] = 1 + dp[i-1][j-1];
}
```