# Dynamic Programming

Video - 117

Note :- This playlist is only for explanation of dns & solutions.

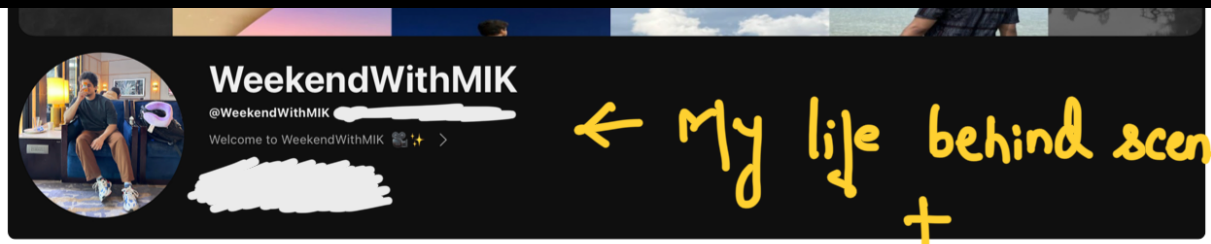
See my "DP Concepts & dns" playlist for understanding DP from scratch...


Leetcode - 2327 Medium


codestorywithmiK

CSwithMIK

codestorywithMIK




← My life behind scenes
+
Tech News/updates

# Motivation :-

Tough times only make you stronger. Never run away from it.

MIK...

## 2327. Number of People Aware of a Secret

Medium | Topics | Companies | Hint

day + delay

On day `1`, one person discovers a secret.

You are given an integer `delay`, which means that each person will **share** the secret with a new person **every day**, starting from `delay` days after discovering the secret. You are also given an integer `forget`, which means that each person will **forget** the secret `forget` days after discovering it. A person **cannot** share the secret on the same day they forgot it, or on any day afterwards.

Given an integer `n`, return *the number of people who know the secret at the end of day* `n`. Since the answer may be very large, return it **modulo** `10^9 + 7`.

n = 6 delay = 2 forget = 4

Example :- n = 6 delay = 2 forget = 4

Output :- 5

**Example 1:**

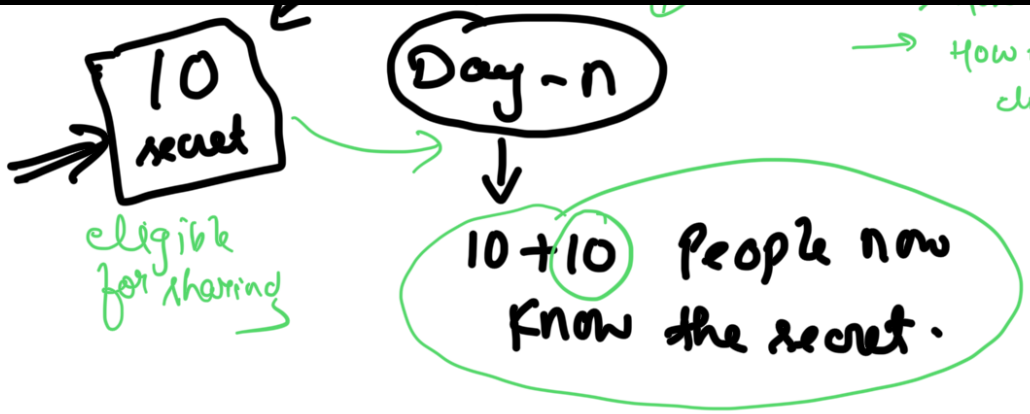
**Input:** n = 6, delay = 2, forget = 4
**Output:** 5
**Explanation:**
Day 1: Suppose the first person is named A. (1 person)
Day 2: A is the only person who knows the secret. (1 person)
Day 3: A shares the secret with a new person, B. (2 people)
Day 4: A shares the secret with a new person, C. (3 people)
Day 5: A forgets the secret, and B shares the secret with a new person, D. (3 people)
Day 6: B shares the secret with E, and C shares the secret with F. (5 people)

A → A
B → A, B
A, B, C
BCD
BCDEF

# Thought Process

n = 6, delay = 2, forget = 4

→ How many ?




n = 6, delay = 2, forget = 4

D1 D2 D3 D4 D5 D6

n - delay

n - forget + 1 → 6 - 4 + 1 = 2 + 1 = 3

People who learned the secret on

Days = [n - forget + 1 , n - delay]

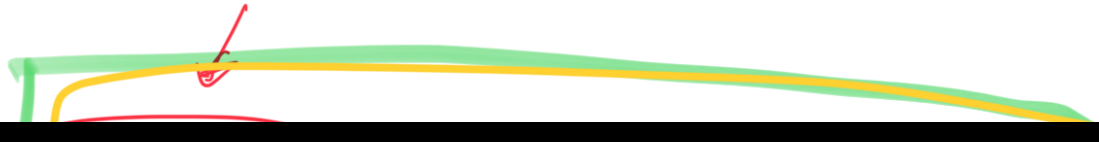
will contribute to $n^{th}$ day.

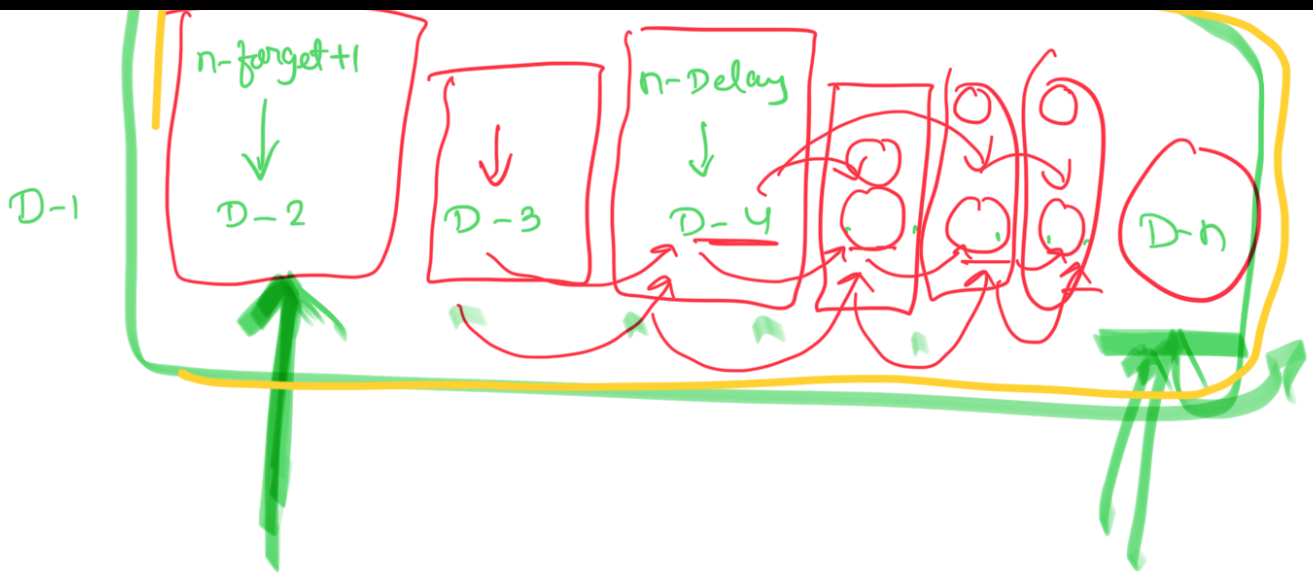
Similar smaller Sub-problem.

(Recursion + Memo + Bottom up) } DP.

**Days ≡ [n - forget + 1, n - delay]**

```
// returns no of people know the secret on "day"
int Solve (day) {
    if (day == 1)
        return 1; only 1 person knows.

    result = 0;
    for( d = day - forget + 1; d <= day - delay ; d++) {

        result = (result + Solve(d)) % M;

    }

    Return result;
}
```






```
total = 0
for (day = n-forget+1; day<=n ; day++){
        total = (total + Solve(day))%M;
}
return total;
```

n-forget

n-delay = (forget - delay)

Time Complexity : O(n * (forget - delay))

S.C = O(n)

# Bottom up :-

Recur + Memo → Memoization t[day] 1-D

Solve (day) = no of people who know the secret on "day"

// Stack definition.

t[day] = no of people who will know the secret on "day".

t[n+1, -1]

t[1] = 1 ;

```
for(int day = 2; day <= n ; day++) {
    int count = 0
    for(int prev = day-forget+1; prev <= day-delay ; prev++) {
        count = (count + t[prev]) % M;
    }
    t[day] = count;
}
```

T.C = O(n * (forget - delay))

```
int result = 0

for( int day = n-forget+1; day<=n; day++){
    if(day >0) {
    result = (rest + t[day]) % M;
    }

{

re rest;
```

T = O(n);

# optimising

## Bottom up

```
t[1] = 1;

for(int day = 2; day <= n; day++) {
    long long count = 0;
    for(int prev = day - forget+1; prev <= day - delay; prev++) {
        if(prev > 0) {
            count = (count + t[prev]) % M;
        }
    }
    t[day] = count;
}

int result = 0;
```

start

end

day

D-2 D-3 start end

count += f[day-delay];

count -= f[day-forget];

```
count = 0;
for( day = 2; day<=n; day++) {
    count = (count + f[day-delay]) % M;
    count = (count - f[day-forget]) % M
    f[day] = count;
}
```

increase day-forget to d-delay