1621. Number of Sets of K Non-Overlapping Line Segments

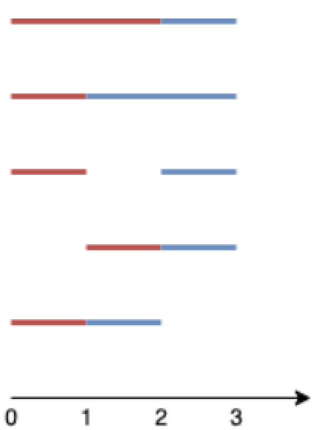
Solved

Medium | Topics | Companies | Hint

Given `n` points on a 1-D plane, where the i^{th} point (from `0` to `n-1`) is at `x = i`, find the number of ways we can draw **exactly** `k` **non-overlapping** line segments such that each segment covers two or more points. The endpoints of each segment must have **integral coordinates**. The `k` line segments **do not** have to cover all `n` points, and they are **allowed** to share endpoints.

Return *the number of ways we can draw* `k` *non-overlapping line segments*. Since this number can be huge, return it **modulo** $10^9 + 7$.

Example 1:




```
Input: n = 4, k = 2
Output: 5
Explanation: The two line segments are shown in red and blue.
The image above shows the 5 different ways {(0,2),(2,3)}, {(0,1),(1,3)}, {(0,1),
(2,3)}, {(1,2),(2,3)}, {(0,1),(1,2)}.
```

Example 2:

```
Input: n = 3, k = 1
Output: 3
Explanation: The 3 ways are {(0,1)}, {(0,2)}, {(1,2)}.
```

Example 3:

> **Input:** n = 30, k = 7
> **Output:** 796297179
> **Explanation:** The total number of possible ways to draw 7 line segments is 3796297200. Taking this number modulo 10^9 + 7 gives us 796297179.

Constraints:

- `2 <= n <= 1000`
- `1 <= k <= n-1`

Let $V_{N,K}$ denote the number of ways to draw K non-overlapping line segments over N identical intervals, which we label $\{1,\dots,N\}$. Consider the rightmost interval N. Suppose a segment occupies this interval. How long is this segment? I argue it cannot cover interval $K-1$, for then there would not be enough space in the remaining $K-2$ intervals to fit the remaining $K-1$ segments.

Suppose now that this segment ends right before interval $I \in \{K-1,\dots,N-1\}$. Then, the problem starts over, but with $K-1$ non-overlapping line segments and I identical intervals. The reader may be tempted to draw from this reasoning the recursion

$$V_{N,K} = V_{N-1,K-1} + \cdots + V_{K-1,K-1}$$

However, this is wrong! We have not considered the possibility that interval N could be empty, in which case we start the problem again but with K non-overlapping line segments over $N-1$ identical intervals (discarding the rightmost one). The correct recursion is

$$\begin{aligned} V_{N,K} &= V_{N-1,K} + V_{N-1,K-1} + \cdots + V_{K-1,K-1} \\ &= V_{N-1,K} + \sum_{I=K-1}^{N-1} V_{I,K-1} \end{aligned}$$

with the boundary conditions

$$V_{N,K} = \begin{cases} 1 & N = K \\ 1 & K = 0 \end{cases}$$

In the case $N = K$, there are exactly as many segments as there are intervals, so there is only one way to fit them all. In the case $K = 0$, we have already placed all of our segments, and so the only way to fill up the remaining leftward space is with nothing. This suggests the code:

```
class Solution:
    def numberOfSets(self, n: int, k: int) -> int:
        @cache
        def V(N,K):
            if N == K or K == 0:
```

```
                return 1

            return V(N-1,K) + sum(V(I,K-1) for I in range(K-1,N))

        return V(n-1,k) % (10**9 + 7)
```

Note that we call on $n-1$ because n points gives us $n-1$ intervals.

While correct, this code would not pass the time constraint because it involves too many function calls. The trick is to introduce another recursion for the sum. Denote

$$S_{N,K} = \sum_{I=K}^{N} V_{I,K}$$

and observe that $S_{N,K}$ obeys the recurrence

$$S_{N,K} = V_{N,K} + S_{N-1,K}$$

and has the boundary condition $S_{K,K} = V_{K,K} = 1$. We obtain the system of recurrences

$$\begin{aligned} V_{N,K} &= V_{N-1,K} + S_{N-1,K-1} \\ S_{N,K} &= V_{N,K} + S_{N-1,K} \end{aligned}$$

subject to the boundary conditions

$$V_{N,K} = \begin{cases} 1 & N = K \\ 1 & K = 0 \end{cases}$$
$$S_{K,K} = 1$$

Coding this up, we obtain

```
class Solution:
    def numberOfSets(self, n: int, k: int) -> int:
        @cache
        def V(N,K):
            if N == K or K == 0:
                return 1

            return V(N-1,K) + S(N-1,K-1)

        @cache
        def S(N,K):
            if N == K:
                return 1

            return V(N,K) + S(N-1,K)

        return V(n-1,k) % (10**9 + 7)
```

The second version of the code runs in $O(nk)$ time and requires $O(nk)$ space.