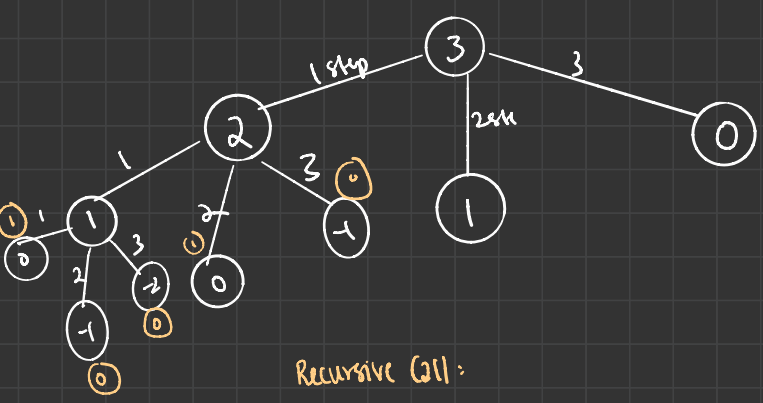
1.2 **Tutorial:** Consider a special version of the `count_stairways` problem, where instead of taking 1 or 2 steps, we are able to take up to and including k steps at a time.

Write a function `count_k` that figures out the number of paths for this scenario. Assume n and k are positive.

```
def count_k(n, k):
    """
    >>> count_k(3, 3) # 3, 2 + 1, 1 + 2, 1 + 1 + 1
    4
    >>> count_k(4, 4)
    8
    >>> count_k(10, 3)
    274
    >>> count_k(300, 1) # Only one step at a time
    1
    """
```

count(3,3) n=3 take 3 steps total
k=3 up to 3 steps per turn



Base Case

n == 0
return 1

n < 0
return 0

Recursive Call:

2 steps: count(n-1) + count(n-2)

3 steps: count(n-1) + count(n-2) + count(n-3)

4 steps: count(n-1) + count(n-2) + count(n-3) + count(n-4)

k steps: count(n-1) + count(n-2) + ... count(n-k)

Base case:

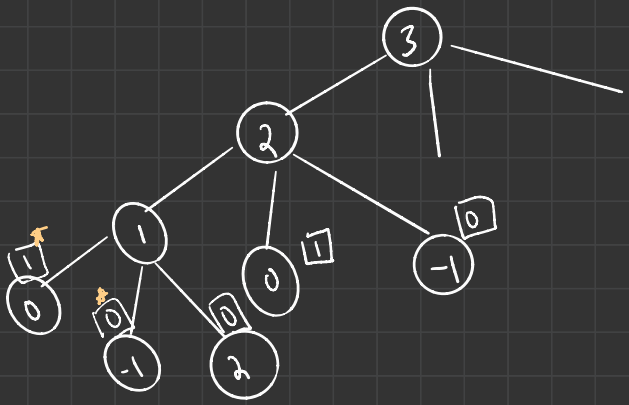
```
n == 0
  return 1
n < 0
  return 0
```

recursive call:

```
i = 1
total = 0
while (i ≤ k)
    total += count(n-i, k)
    i += 1
return total
```

```
factorial(n)
if n == 1:
    return 1
return n * factorial(n-1)
```

```
factorial(n):
  i = 1
  total = 1
  while (i <= n)
    total = total * i
    i += 1
  return total
```




```
if n == 0:
    return 1
if n == 1:
    return 0
else:
    count_k(n-1, k) +
    count_k(n-2, k) +
```

```
total = 0
i = 1
while (i <= k)
    total += count_k(n-1, k) +
             count_k(n-2, k) + ...
             count_k(n-2, k)
return total
```