Tree Recursion

Tree-shaped processes arise whenever executing the body of a recursive function makes more than one recursive call.

\[
\begin{align*}
0, & 1, 2, 3, 4, 5, 6, 7, 8, \ldots, 15, 30 \\
\text{fib}(n): & \quad n, n, 1, 2, 3, 5, 8, 13, 21, \ldots, 8, 13, 21
\end{align*}
\]

```python
def fib(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fib(n-2) + fib(n-1)
```

Repetition in Tree-Recursive Computation

This process is highly repetitive; `fib` is called on the same argument multiple times.

Counting Partitions

```
def knap(n, k):
    if n == 0:
        return k == 0
    with_last = knap(n//10, k-n%10)
    without_last = knap(n//10, k)
    return with_last or without_last
```

Example: Counting Partitions

```
count_partitions(6, 4)
count_partitions(10, 10)
count_partitions(10, 6)
count_partitions(16, 16)
count_partitions(10, 16)
count_partitions(16, 16)
```

Counting Partitions

The number of partitions of a positive integer \( n \), using parts up to size \( n \), is the number of ways in which \( n \) can be expressed as the sum of positive integer parts up to \( n \) in increasing order.

```
count_partitions(16, 16)
```

A Tree-Recursive Process

The computational process of `fib` evolves into a tree structure.
Counting Partitions

The number of partitions of a positive integer \( n \), using parts up to size \( m \), is the number of ways in which \( n \) can be expressed as the sum of positive integer parts up to \( m \) in non-decreasing order.

- Recursive decomposition: finding simpler instances of the problem.
- Explore two possibilities:
  - Use at least one 4
  - Don’t use any 4
- Solve two simpler problems:
  - count_partitions(2, 4)
  - count_partitions(6, 3)

Tree recursion often involves exploring different choices.

def count_partitions(n, m):
  if n == 0:
    return 1
  elif n < 0:
    return 0
  elif m == 0:
    return 0
  else:
    with_m = count_partitions(n-m, m)
    without_m = count_partitions(n, m-1)
    return with_m + without_m

(Demo)