RECURSION

**Step1) What is the simplest possible input? Our base case.**

ex – sum of all non-negative integers – sum(0) -> 0

**Step2) Play around with examples and visualize!**

See if any interesting relationship between cases.

Write tons of example, it will make you comfortable with the problems

ex – n = 1 (1)

n = 2 (1+2)

n = 3 (1+2+3)

n = 4 (1+2+3+4)

**Step3) Relate hard case to simpler cases.**

**Build a relatable case from hard to simpler case to generalize a pattern.**

ex – Can you relate sum(4) and sum(5)

Can you relate sum(3) and sum(4)

**Step4) Generalize the pattern**

Sum(n) = {0 if n = 0}

{sum(n-1) +n}

* ex –
  + sum (5)
  + 5 + sum (5-1)
  + 4 + sum (4-1)
  + 3 + sum (3-1)
  + 2 + sum (2-1)
  + 1 + sum (1-1)
  + 0

**RECURSIVE LEAP OF FAITH**

- Assume simpler cases work out.

**Q) Write a function that takes two inputs n and m and outputs the number of unique paths from the top left corner to bottom right corner of a n x m grid.**

**Constraints: you can only move down or right 1 unit at a time.**

Grid\_paths(2, 4) - > 4

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

Paths



Grid\_paths(3, 3) -> 6

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

Paths



**Step1) What’s the simplest possible input?**

Grid\_paths(1,1) -> 1

Grid\_paths(2,1) -> 1

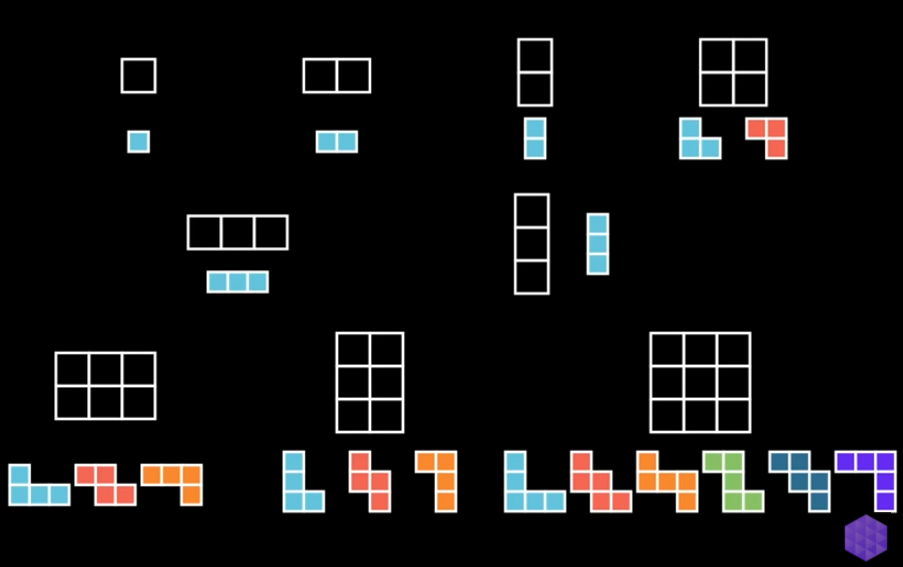
Grid\_paths(1,2) -> 1

Grid\_paths(1,m) -> 1

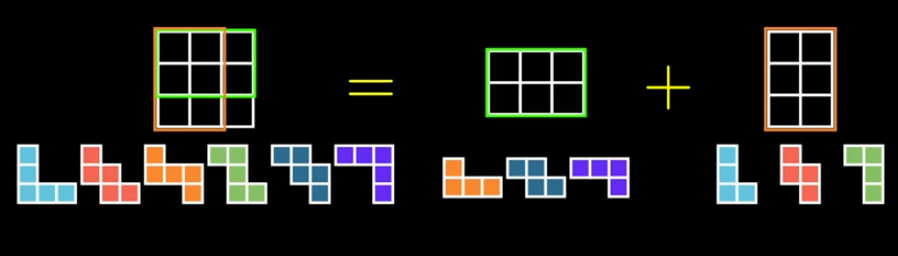
Grid\_paths(n,1) -> 1

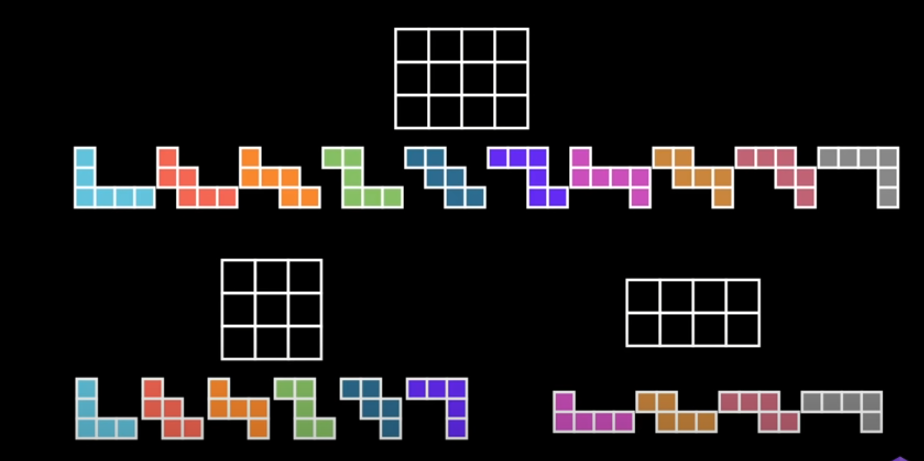
Grid\_paths(n, m) -> 1 if n = 1or m= 1 (base case)

**Step2) Play around with examples and visualize!**

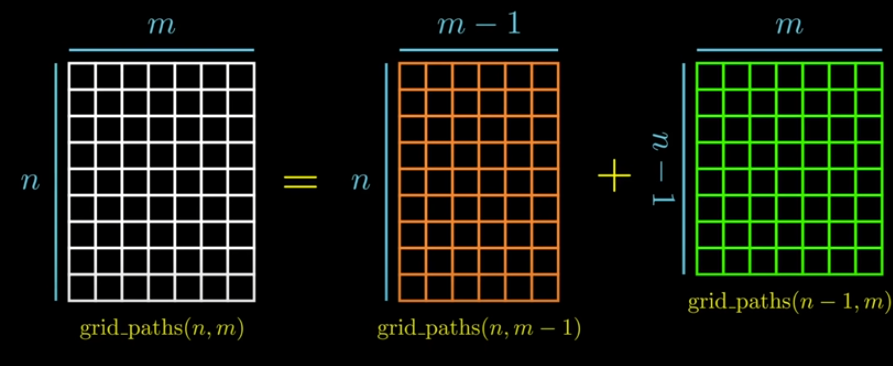


**Step3) Relate hard cases to simpler cases.**





**Step4) Generalize the pattern**

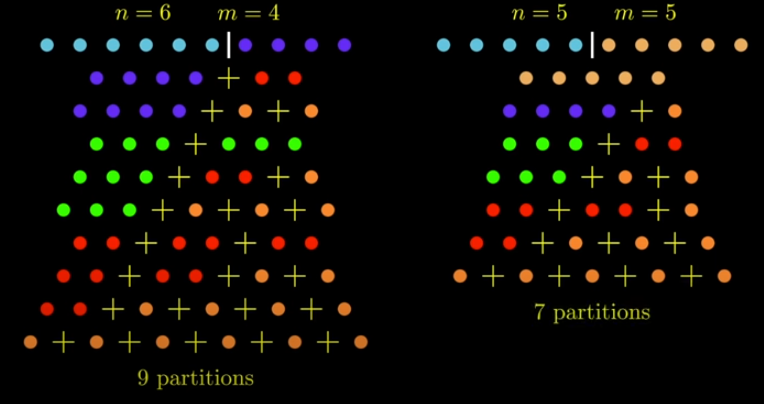


**Step5)Write code by combining recursive pattern with base case.**

Grid\_paths(n, m) = { 1 if n = 1 or m = 1 }

{ grid\_paths(n, m-1) + grid\_paths(n-1, m) }

**Q)Write a function that counts the number of ways you can partition n objects using parts up to m (assuming m >= 0)**



**Step1) What is the simplest possible input?**

Count\_partitions(0, 0) -> 1

Count\_partitions(0, 1) -> 1

Count\_partitions(0, 2) -> 1

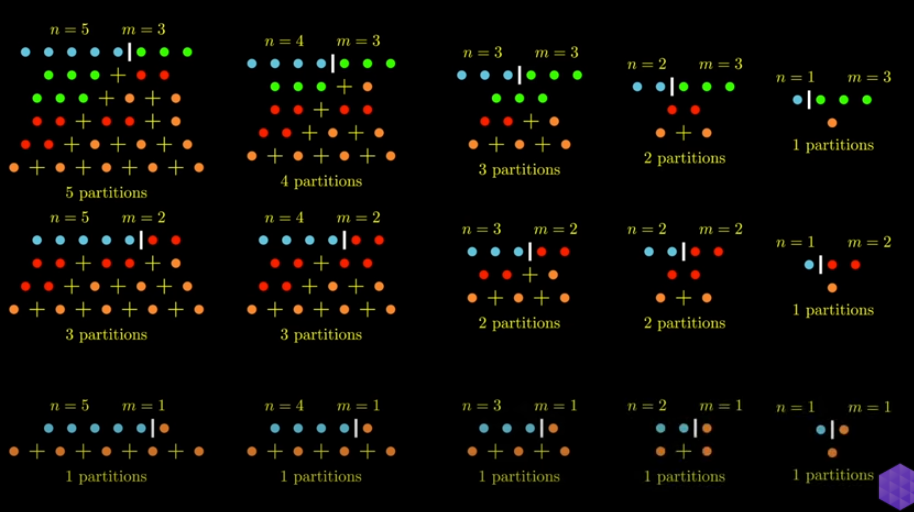
**Count\_partitions(n, m) - > 1 if n = 0**

Count\_partitions(1, 0) -> 0

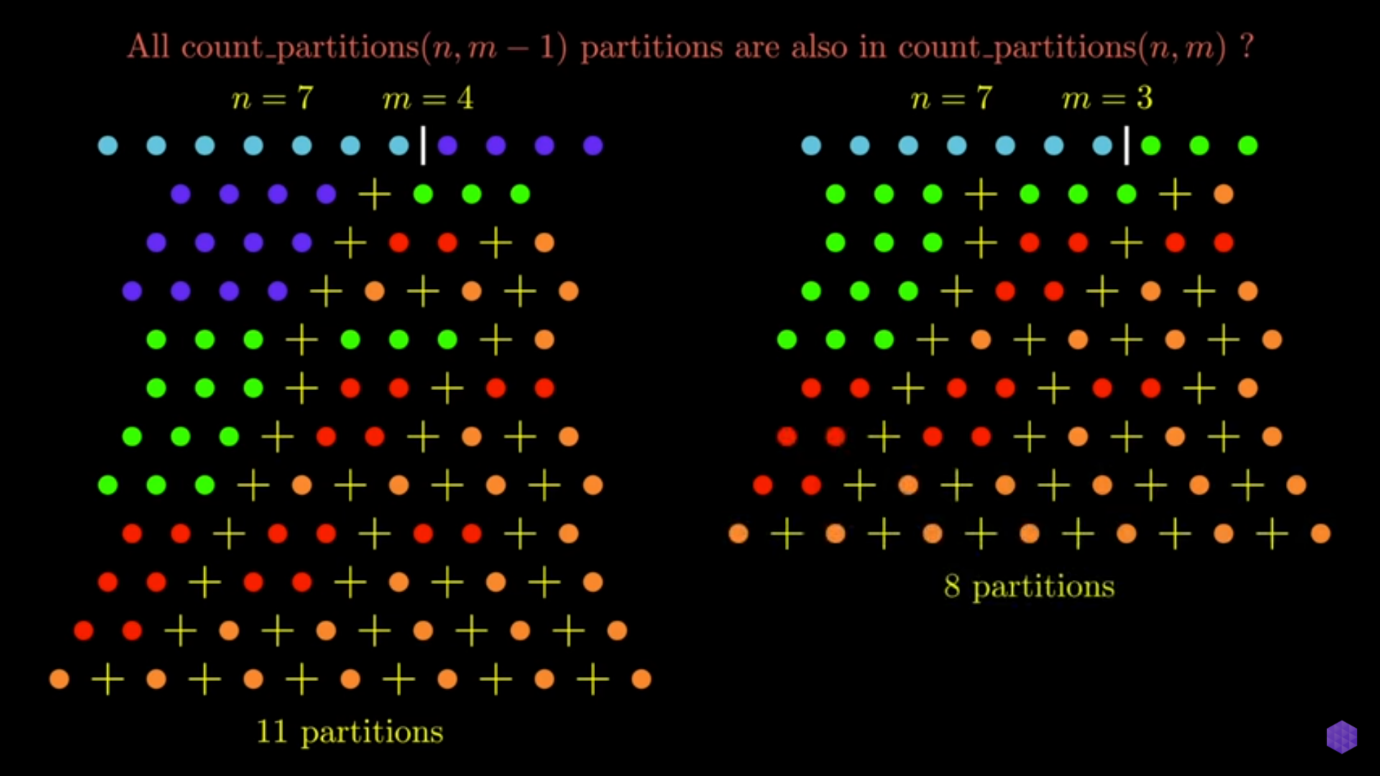
Count\_partitions(2, 0) -> 0

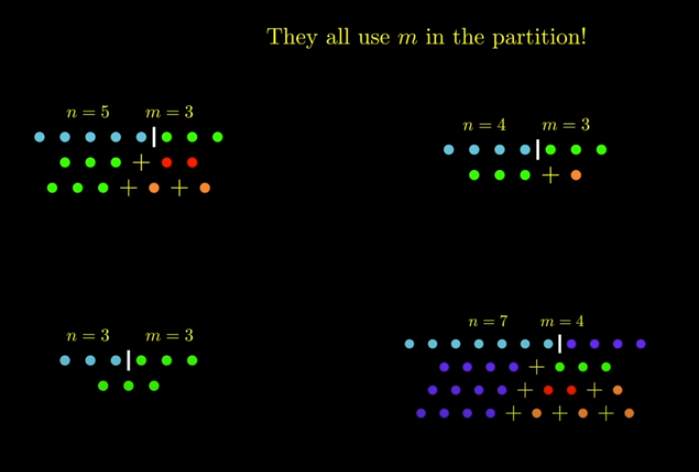
**Count\_partitions(n, m) -> 0 if m = 0**

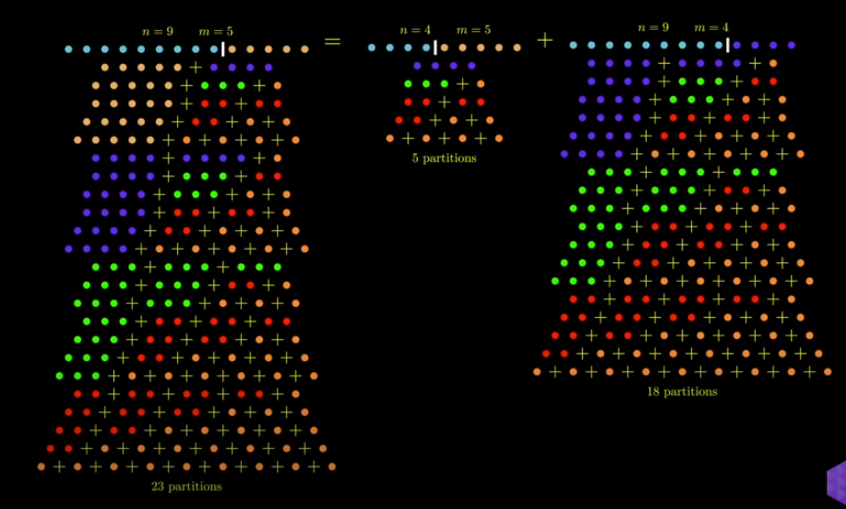
**Step2) Play around with examples and visualize!**



**Step3) Relate hard cases with simple cases**

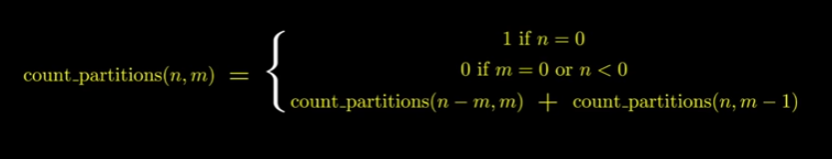
****

****

****

**Step4) Generalize the pattern with base case**

**Count\_partitions(a, b) = Count\_partitions(n-m, m) + Count\_partitions(n, m-1)**

****