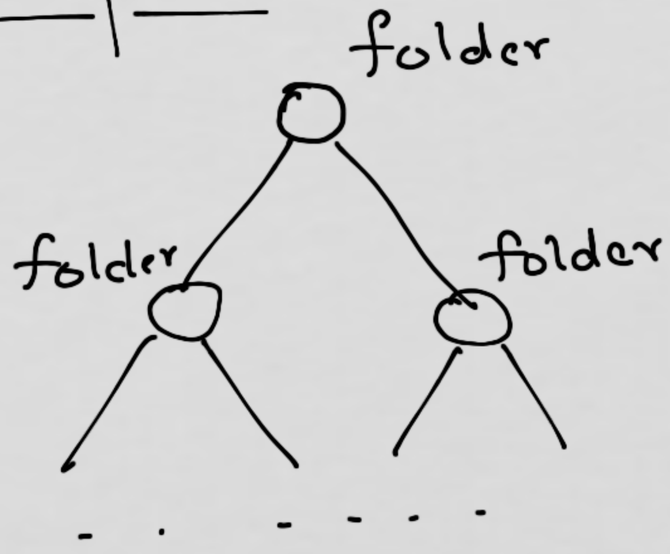
COL 100 - Lec 10

1. Uses of examples worked out

- Counting change → In vending machines
  - more generally, it is a special case of integer knapsack problem [resource allocation]

- Tower of Hanoi → disk backups, neuropsychological tests etc.

More Examples



Searching for a folder in the file system.

- getLeftChild
- getRigetChild

Search (root, nameOfFolder)

$$= \begin{cases} \text{root} & \text{if root's name = nameOfFolder} \\ \text{NotFound} & \text{if root's name = NULL} \\ \text{Search}(\text{getLeftChild}(\text{root}), \text{nameOfFolder}) & \text{Otherwise} \\ \text{Search}(\text{getRightChild}(\text{root}), \text{nameOfFolder}) & \end{cases}$$

$$\text{getLeftChild}(\text{node}) = \begin{cases} \text{NULL} & \text{if node is a leaf} \\ 1^{st}\ \text{Child} & \text{Otherwise} \end{cases}$$

$$\text{getRightChild}(\text{node}) = \begin{cases} \text{NULL} & \text{if node is a leaf} \\ 2^{nd}\ \text{child} & \text{if \# of children} = 2 \end{cases}$$

Q: How does the above function unfold?
[Also called Depth first Search]

Q: Consider a binary tree, perform search over a binary tree? Can you do better?

Ex2:

Consider a 4 bit binary numbers

— find all 4-bit (in general N-bit) numbers without consecutive 1's.

i.e.

0 0 0 0

0 0 0 1

0 0 1 0

0 1 0 0

0 1 0 1

1 0 0 0

1 0 0 1

1 0 1 0

Base Case ?

i) n = 0 then 0

[Base case not finished yet!]

Observations

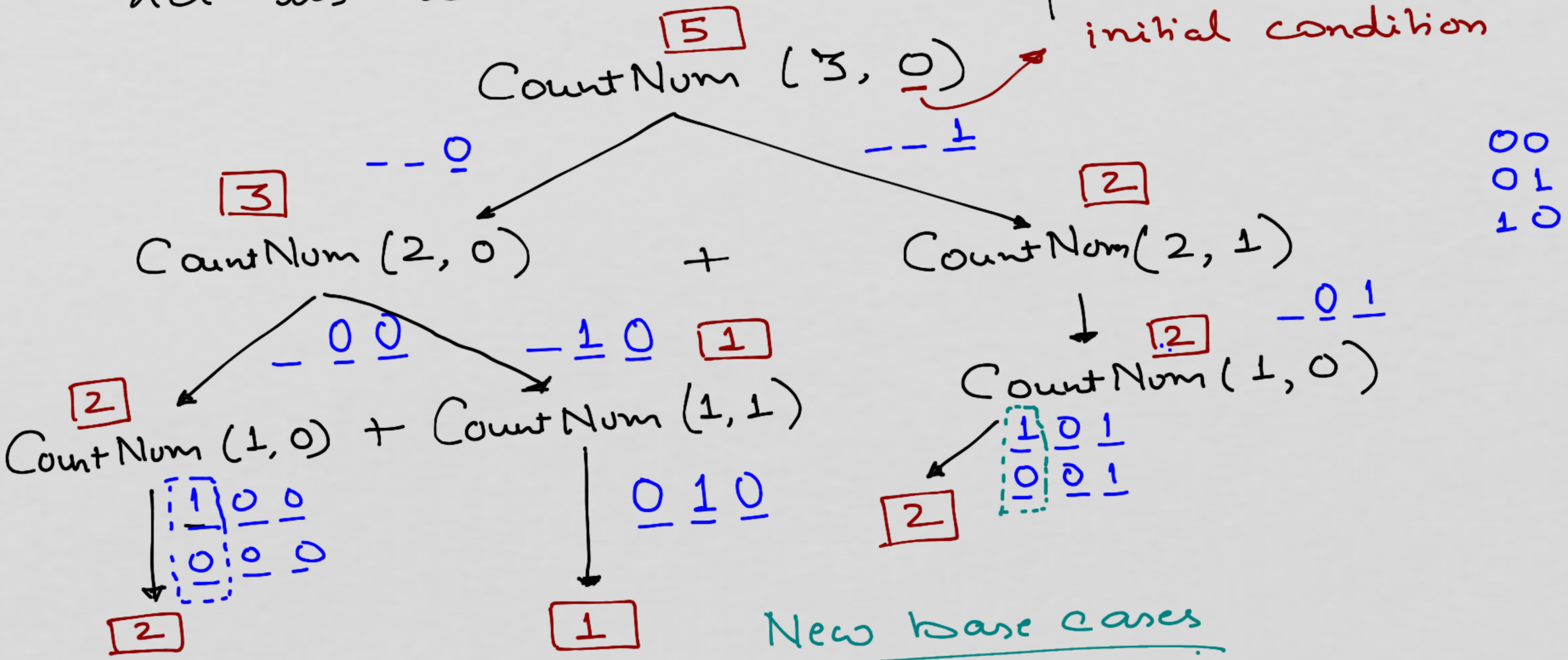
- if last digit is 0
  - we can have both 0 & 1 in the current digit.

countNum(n-1, 0) + count(n-1, 1)

- if last digit is 1
  - we can only have 0 at current position

countNum(n-1, 0)

Let us work out an example



New base cases

- when n = 1, last digit = 0
  then return 2
- when n = 1, last digit = 1
  then return 1

Therefore

$$CountNum(n, k) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \wedge k = 1 \\ 2 & \text{if } n = 1 \wedge k = 0 \\ CountNum(n-1, 0) + CountNum(n-1, 1) & \text{if } n > 0 \wedge k = 0 \\ CountNum(n-1, 0) & \text{if } n > 0 \wedge k = 1 \end{cases}$$

Alternate Solution

n = 1

0

1

2

n = 2

00

01

10

1 1

3

n = 3

000

001

010

011

100

101

110

111

5

n = 4

0000

0001

0010

0011

0100

0101

0110

0111

1000

1001

1010

1011

1100

1101

1110

1111

X

5 + 3

Therefore

$\forall n \geq 3$, Count (N) = Count (N-1) + Count (N-2)

Count (1) = 2 , Count (2) = 3

If I had asked you to generate such binary strings, then?

GenNum(N, numStr, step, last_digit)

→ first generate all strings starting with '0'

- if N = step
  then print numStr
- if last_digit is 1
  then numStr ^ "0"
  GenNum (N, numStr, n+1, 0)
- if last_digit is 0
  then

GenNum (N, numStr^"0", n+1, 0)
GenNum (N, numStr^"1", n+1, 1)

String concatenation Syntax

- May change from one language to other
- In C → string.concat (a, b)
- In SML → "Hello" ^ "_World"
- In python → "Hello" + "World"