Recursion

Agenda of the Microcours → Basics of Recur
→ 1 week

Intermediate Recur

Problem solving

Agenda for 1 week Backtracking → Basics

Intermediate

Problem solve

Q→ What is Recursion ??

→ Mathematical Definition → It's a logical procedure, which is specified by a sub procedure that yields values or instances of a function repeatedly by applying a given routine operation

$$f(x) \Rightarrow f(f(a))$$

2 2

→

Recursion is defined when a function calls itself by applying some subroutine on the parameters by keeping an extra space overhead.

factorial

$\rightarrow f(n) = n \times f(n-1)$

func^n that return $n!$

$5! = 5 \times 4!$

broke down bigger problem $\rightarrow$ smaller subproblem

$n = 5$

Recursion Tree

$f(n)$

$f(5)$ 120

$5 \times f(4)$ 5×24

$4 \times f(3)$ 4×6 6

$3 \times f(2)$ 3×2 2

$2 \times f(1)$ 2×1

if $f(n) \rightarrow$ correct implement of $n!$

$f(4)$

Principal of Mathematical Induction → Recursion

$1 + 2 + 3 + 4 + 5 \ldots\ldots\ldots\ldots n$ (Sum of n natural no.)

$\frac{n}{2}$

$= \left(\frac{n \times n+1}{2}\right)$

Three steps

1 → what me minimal value for which we know the ans

$\boxed{\text{Proof}}$

$\frac{1 \times (1+1)}{2} = 1$

$n = 1$

1

2 → Assume the formula works for $n = k$

$$\frac{k \times (k+1)}{2} \qquad n = (k+1)$$

3 → Prove that formula works for $k+1$

$$\frac{(k+1)(k+2)}{2}$$

1 → find out the smallest subproblem for which we know the ans

2 → Assume that for the given problem recursion will correctly calc a subproblem

3 → Self work

$$\left(f(n) = n \times f(n-1) \right) \longrightarrow \text{Recurrence Relation}$$

factorial

1) $\Rightarrow$ $n = 1$ $\quad f(1) = 1$ $\quad$ (Base Case)

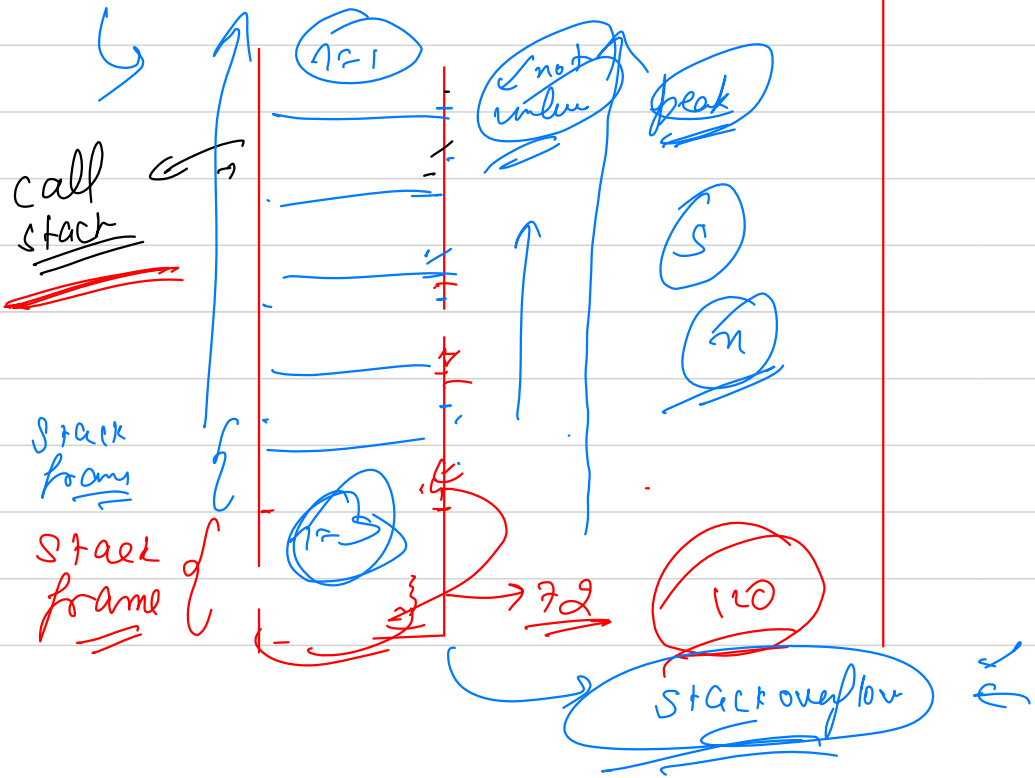
2) $\Rightarrow$ Calc $f(n-1)$ $\longrightarrow$ Recursive assumption

3) Self work return $n \times f(n-1)$

Q. How memory is distributed for a process ??

Stack (Linear stack)

heap (big pool)



return → stack frame is removed

$f(n) \rightarrow$ factorial

Total operation

2

$TC \rightarrow O(n)$

$SC \rightarrow O(n)$

2

$O(1)$

a) avg space
b) total space
c) max space
d) None

Space complexity is defined as the max space allocated at any point of time during execution of the process

any i^{th} term is sum of previous 2 terms

0th 1st 2nd 3rd 4th 5th 6th - - - -

Q ⇒ 0, 1, 1, 2, 3, 5, 8, 13, ... (fibonacci series)

Write a recursive program to calc n^{th} fib

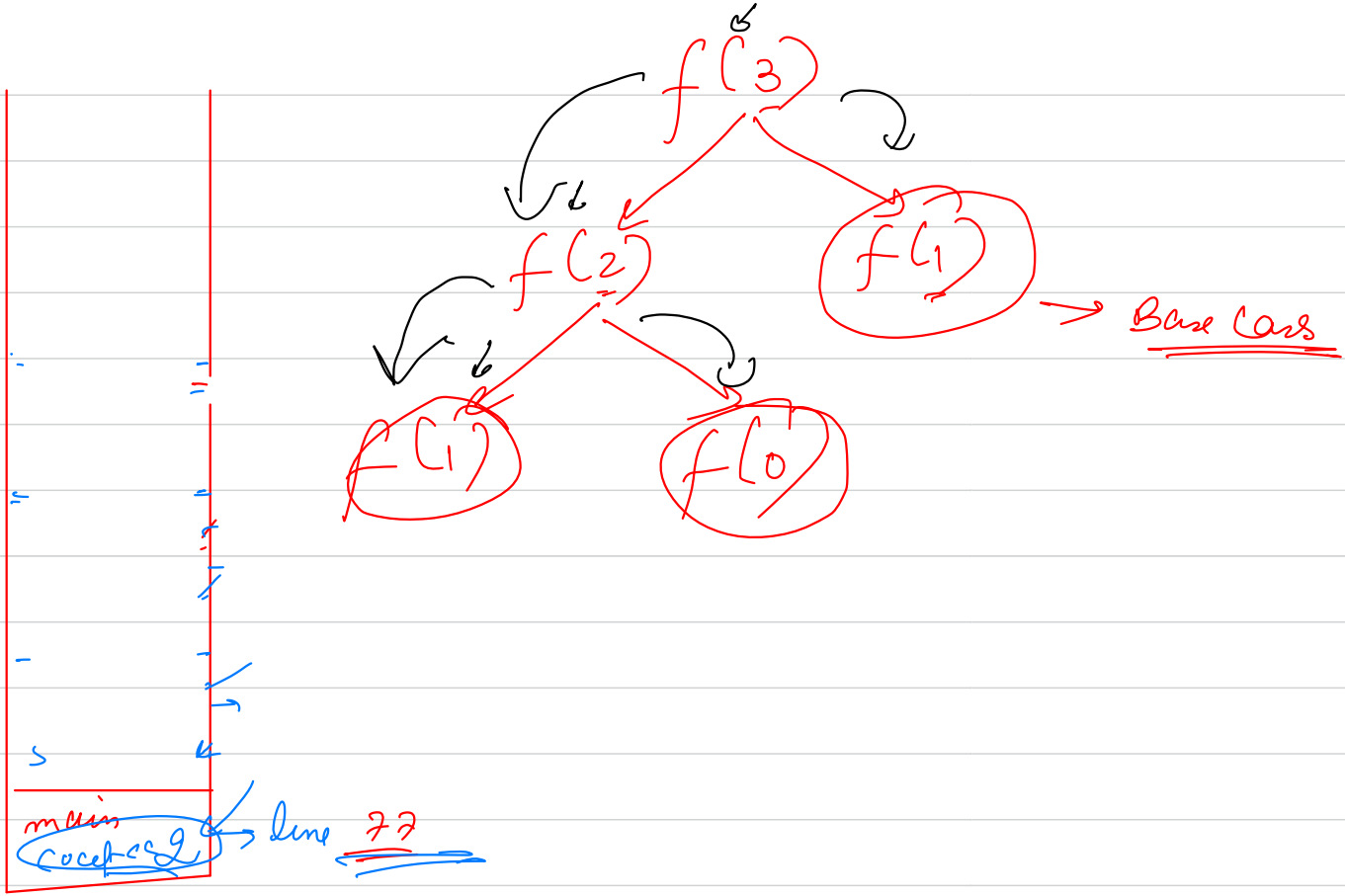
1) Base Case → 0th fib == 0 & 1st fib == 1

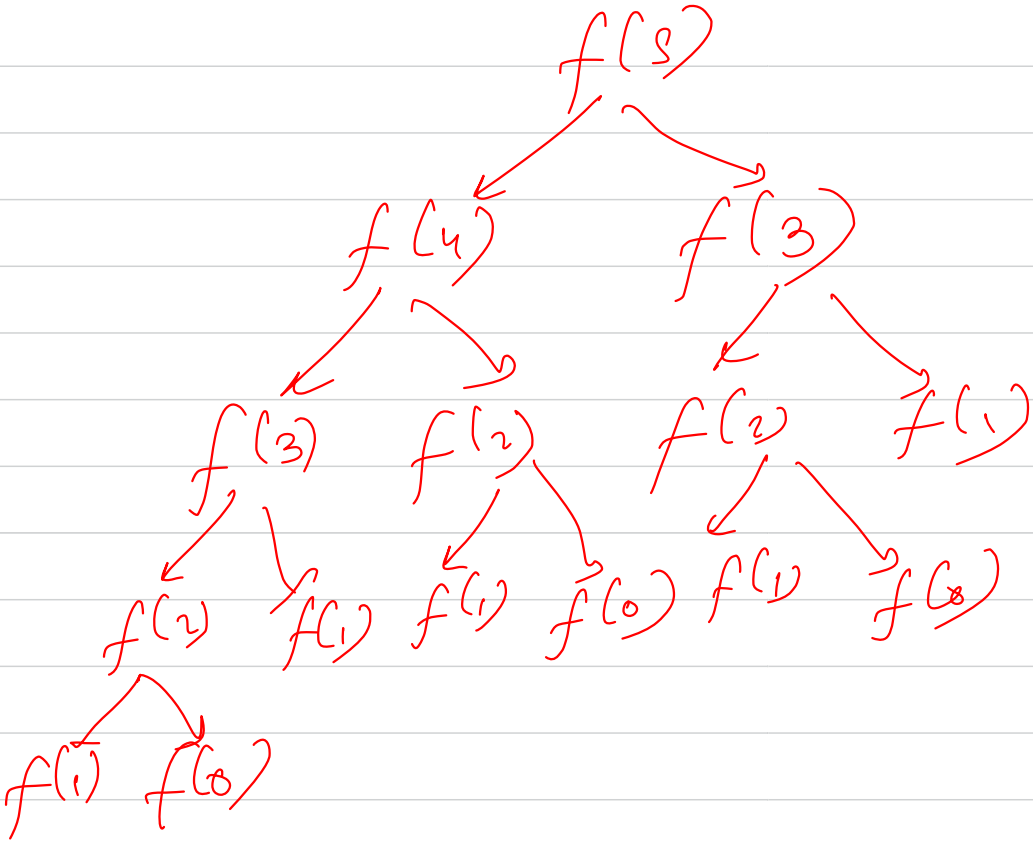
2) Recursive entities ⇒ f(n) depends on calc f(n-1) and f(n-2)

n^{th} fibonacci

3) Self work = return f(n-1) + f(n-2)

$$f(n) = f(n-1) + f(n-2)$$


f(3)
f(2)
f(1)
f(1)
f(0)
Base Cases
main
fact(3)
line 77


f(5)
f(4)
f(3)
f(3)
f(2)
f(2)
f(1)
f(2)
f(1)
f(1)
f(0)
f(1)
f(0)
f(1)
f(0)

Q: Print first N natural numbers, recursively

N = 5

1

2

3

4

5

1) Base Case → n == 1 or n == 0

2) Recursion Assumption → Anyhow go and print any the from 1 → n-1

→ execute f(n-1)

3) Self work → print n

f(n)

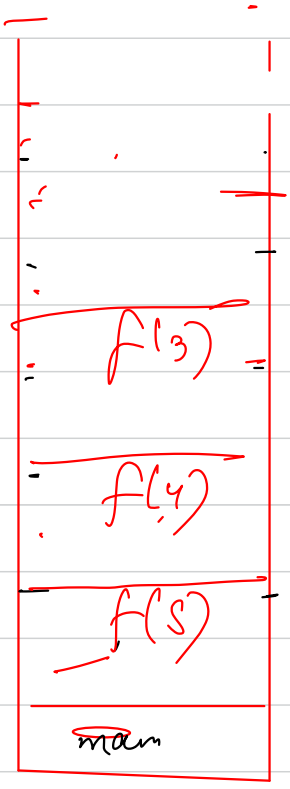
print 1 → n

n == 0

5 → 4 → 3 → 2 → 1 → 0

n == 1

print 1



5
4
3
2
1

→ countdown(n)
f(n-1)

Q: Given n print first n natural no, in decreasing order

n = 5

5
4
3
2
1

Q.n

n=5

single recursion

5
4
3
2
1
2
3
4
5

Reason assumption → Go and print $\overbrace{n-1 \text{ — } 1}^{\text{dec}} \overbrace{\text{ — } n-1}^{\text{inc}}$

Selfwork → print 1 before & after

Q. Given a value n, how many binary strings of length n are there with no consecutive ones

Recurrence

$\boxed{n=3}$

ans → 5 =

000
001
010
100
101

$\boxed{n=4}$

0000
0001
0010
0100
1000
1001
1010
0101

→ 8

SANKETID

Fib

$n=1$ — 0, 1 → 2

$n=2$ — 00, 10, 01 → 3

$n=3$ — 5

$n=4$ — 8

$n=5$ — 13

Fibonacci

00000
00001
00010
00100
01000
10000
10001
10010
10100
10101

01010
01001
00101

$$f(n) = f(n-1) + f(n-2)$$

$n == 0 \rightarrow 2$

$n == 1 \rightarrow 3$

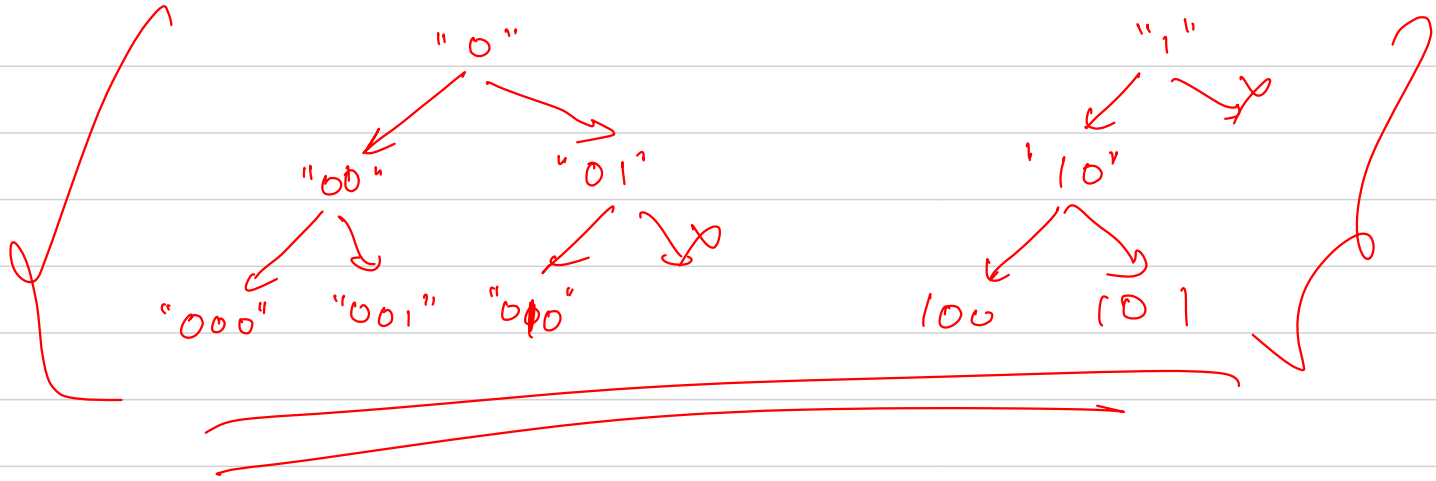
Base Case

$n = 1 \rightarrow \boxed{0} \rightarrow 2$

1

$n = 2 \rightarrow$ 0 0, 1 0, 0 1 $\rightarrow 3$

$n = 3 \Rightarrow f(2) + f(1)$


"0"
"1"
"00"
"01"
"10"
"000"
"001"
"010"
100
101