finding of Power of an element

Interview Question (Amazon)

$\begin{cases} n = 16 \\ a = 2 \end{cases} \Rightarrow 2^{16} = 2^8 * 2^8$

$n \rightarrow$ even $\longrightarrow$ Divide & Conquer

$2^n = 2^{n/2} * 2^{n/2}$

65536 = 256 * 256

2^{16} $\quad$ 2^{17}

mid = 8

b * b

Small problem

↳ n = 0

2^8 → mid $\quad$ 2^8

16 * 16 = 256

2^4 $\quad$ 2^4

b

4 * 4 = 16

2^0, 4^0 $\Big\}$ $\begin{cases} a^0 = 1 \\ a' = a \end{cases}$

Small ← Problem

2^2 $\quad$ 2^2

2 * 2 = 4

2^1 $\quad$ 2^1

4 small Problem

↳ 2

$2^{17} = 2^{16} * 2$

$= 131072$

b = 2

Result = 2 * 2

= 4

$2^{47} = 2^{46} * 2$

$7/2 = 3$

$2^{14} = 2^7 * 2^7$

$2^6 * 2$

2^3 $\quad$ 2^3 $\quad$ $2^2 * 2$

$2^1 * 2^1$

$n = -2$

$a = 2$

$2^{-2} \Rightarrow \frac{1}{2^2}$ ←

c_1

Recursive Tree

findPower(2, 16) ~65536

c_2

256 * 256 = 65536

b = findPower(2, 8)

c_3

16 * 16 = 256

b = findPower(2, 4)

c_4

4 * 4 = 16

b = findPower(2, 2)

c_5

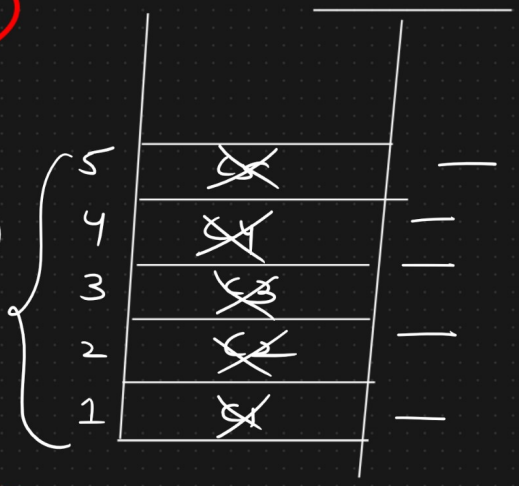
4

a

b = findPower(2, 1) → Small Problem ↳ c

↳ 2

Stack



Skewed Recursive Tree

Stack space = O(n)

↳ Worst Case

CBT Recursive Tree

Stack space = O(log n)

↳ Best & average

$2^{17} = (2^{16}) * 2 = \_\_\_\_$

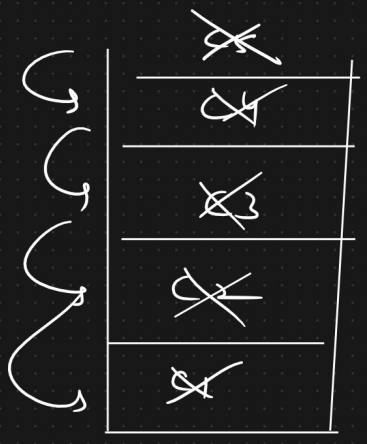
↳ Divide & conquer

65536

b = 2

result = 4

b = 4



Error

↳ Recursion in depth exceeded

↳ Recursion → n is very high

↳ Dynamic Programming

Psuedocode — T(n)

```
findPower(a, n):
    n == 0
        ↳ 1                    c
    n == 1
        ↳ a
    n > 1                      Divide — c
        ↳ mid = n // 2
T(n/2) ——— b = findPower(a, mid)
```

Small Problem

```
c — result = b * b

c — { if n % 2 == 0 :
        return result
    else
        return result * a
```

$$T(n) = T(n/2) + c$$

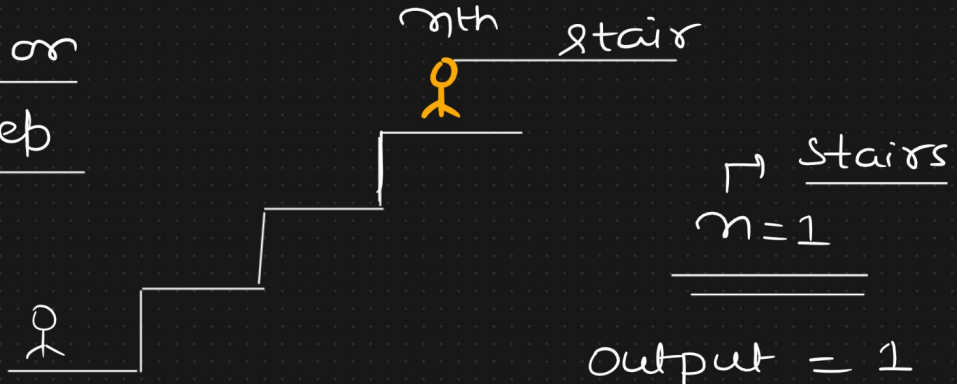
$$= O(\log_2 n)$$

---

count number of ways to reach

2 steps or

1 step



↳ Stairs

$n = 1$

Output = 1

$n = 2$

Output = 2

(1,1) and 2

$n = 4$

Output = 5

(1,1,1,1) (2,1,1)

(1,1,2) (2,2)

(1,2,1)

↳ 1 2 3 4 5 6 7

1 2 3 5 8 13 21

↳ fibonacci Series

8 9

34 55

10

89

n = 3

↳ (1,1,1)

(1,2)

(2,1)

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

$ways(n) = fib(n+1)$

$ways(1) = fib(2) = 1$

$ways(2) = fib(3) = 2$

$ways(3) = fib(4) = 3$

correlation

fibonacci Series

0 1 2 3 4 5 6 7

0 1 1 2 3 5

n = 3

↳ output = 3

n = 4

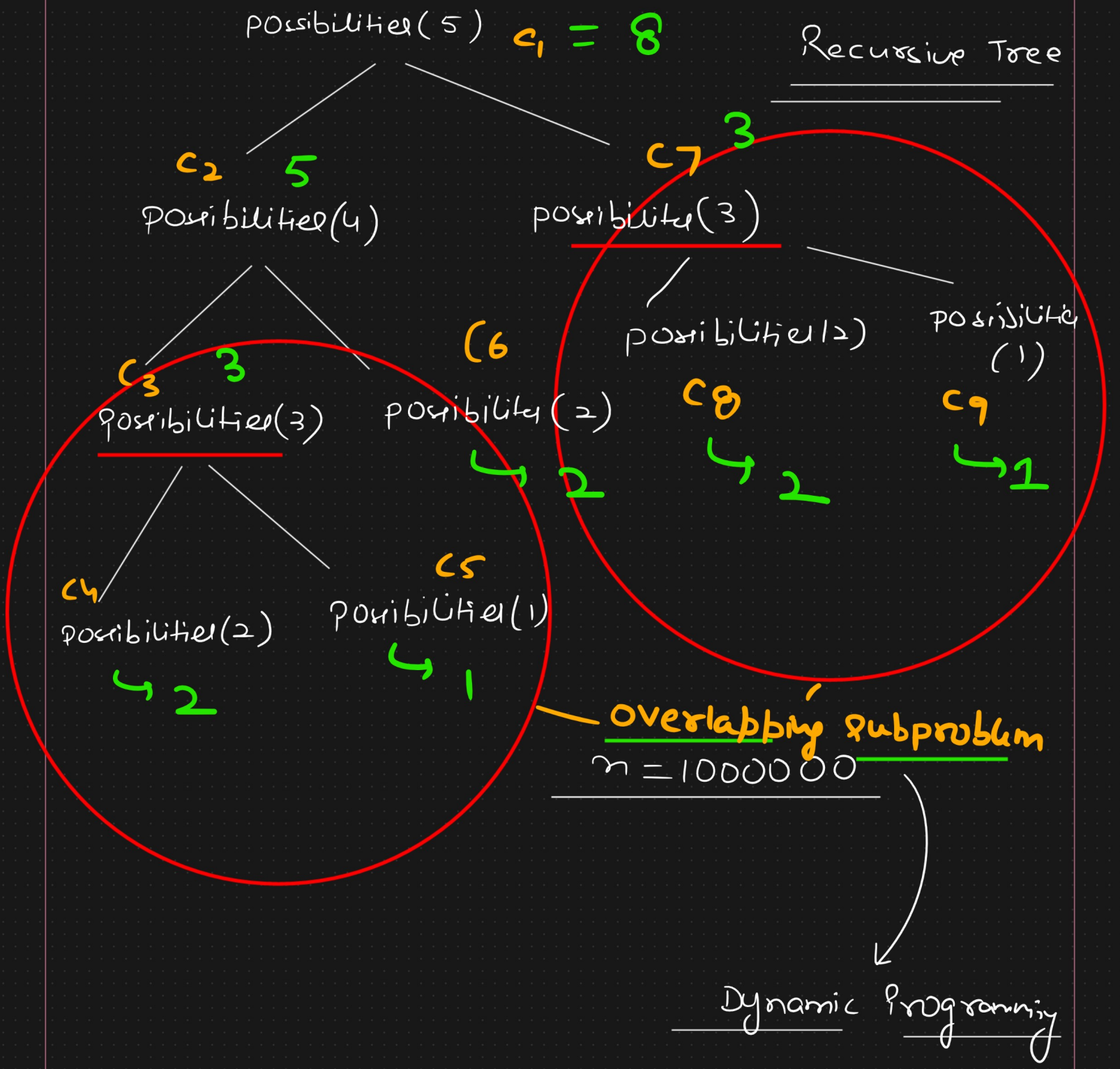
↳ output = 5

Base case condition

n == 1 ↳ 1

n == 2 ↳ 2

n > 2 ↳ possibilities (n-1) + possibilities (n-2)


possibilities(5) c1 = 8
Recursive Tree
c2 5
possibilities(4)
c7 3
possibilities(3)
c3 3
possibilities(3)
c6
possibilities(2)
4 2
possibilities(2)
c8
4 2
possibilities(1)
c9
4 1
c4
possibilities(2)
4 2
c5
possibilities(1)
4 1
Overlapping subproblem
n = 1000000
Dynamic Programming

maximum purchase
↳ product → max freq → output

⇓

clash

Redhat → 2
yellowhat → 2
blackshirt → 2

[blackshirt, redhat, yellowhat]

Deadline → 2 Days (Sunday, Monday)

↳ priyabhatia@ineuron.ai

7th Oct → friday

8-10pm

↳ Recording

Approaches → Assignment 7 & Assignment 6 → Heap

Interview

Data Structure

→ MergeSort

→ QuickSort

} Application of divide & conquer