WEEK2 WEBINAR SESSION

Agenda

- Recurrence Relation
- Substitution Method
- Recursion Tree Method
- Master Method
- Divide and Conquer

Recurrence Relation

- When should a constant be dropped from a recurrence relation?

In recurrence relations, there are two parts

- recursive part T(n/2)
- non recursive part: c or n or both

$$T(n) = \underbrace{T(n/2)}_{\text{Recursive part (you will retain this one)}} + \underbrace{\qquad\qquad\qquad}_{\text{Some terms}}$$

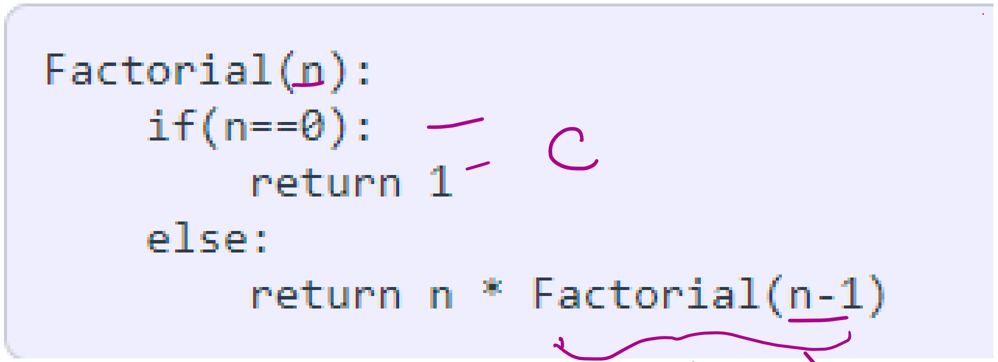
If there are multiple terms here, you can approximate these terms ignoring lower degree terms.
If there is only single term, you would retain it.
Ex:

$$n^2 + n + c \Rightarrow n^2$$
$$n + c \Rightarrow n$$
$$c \Rightarrow c$$

Recurrence Relation

- How to write recurrence relations
- How do we write recurrence relations with multiple if statements?

$T(n)$ $T(n-1)$



```
Factorial(n):
    if(n==0):
        return 1        c
    else:
        return n * Factorial(n-1)
```

$T(n-1)$

$T(n) = \underline{T(n-1)} + \underline{\boxed{c}}$

Recurrence Relation (multiple answers)

A. ~~T(n) = T(n-1)~~
B. T(n) = T(n-1) + c ✓
C. T(n) = T(n-1) + <u>theta(1)</u> ✓
D. T(n) = <u>2</u>T(n-1) + c ✗

Recurrence Relation

$\underline{\underline{\theta(1)}}$ $\underline{\underline{c}}$

- Base Case $T(n)$

```
def climbStairs(n: int):
    if (n <= 0):
        return 0
    if (n == 1):
        return 1
    if(n ==2):
        return 2
    return climbStairs(n - 1) + climbStairs(n - 2)
```

c $n \le 2$

Base case:

A. T(n) = c for all n <=2 ✓

B. T(n) = c1 for n <=0
   T(n) = c2 for n = 1 ✓
   T(n) = c3 for n = 2

C. T(n) = $\underline{\text{theta}}$(n) for n <= 2

n

$\underline{\text{theta}}(1)$ ✓

Recurrence Relation

$T(n) = T(n-1) + c$

- Multiple parameters:

```
def pow(x, n):
   if n == 1:
      return x

   if n != 1:
      return x * pow(x, n - 1)
```

$n-1$

$x * x * (n-2)$

$x * (n-3)$ . . . . . . $n=1$

Recurrence relation function notation

A. T(x,n)

B. T(n)

C. T(x)

D. None

Recurrence Relation

$T(m) = T(m-1) + \underline{c}$

- Multiple if else conditions:

```
def foo(m):
  if(m==0):  ✓
      return 0

  elif(m is odd):  ✓
      return foo(m-1) + 1 ✓
  else:  ✓
      return foo(m-1) + 2 ✓
```

$\xrightarrow{} T(m-1)$

OR

$\xrightarrow{} T(m-1)$

Recurrence relation:

A. T(m) = 2T(m-1) + c ✗

[✓ B. T(m) = T(m-1) + c]

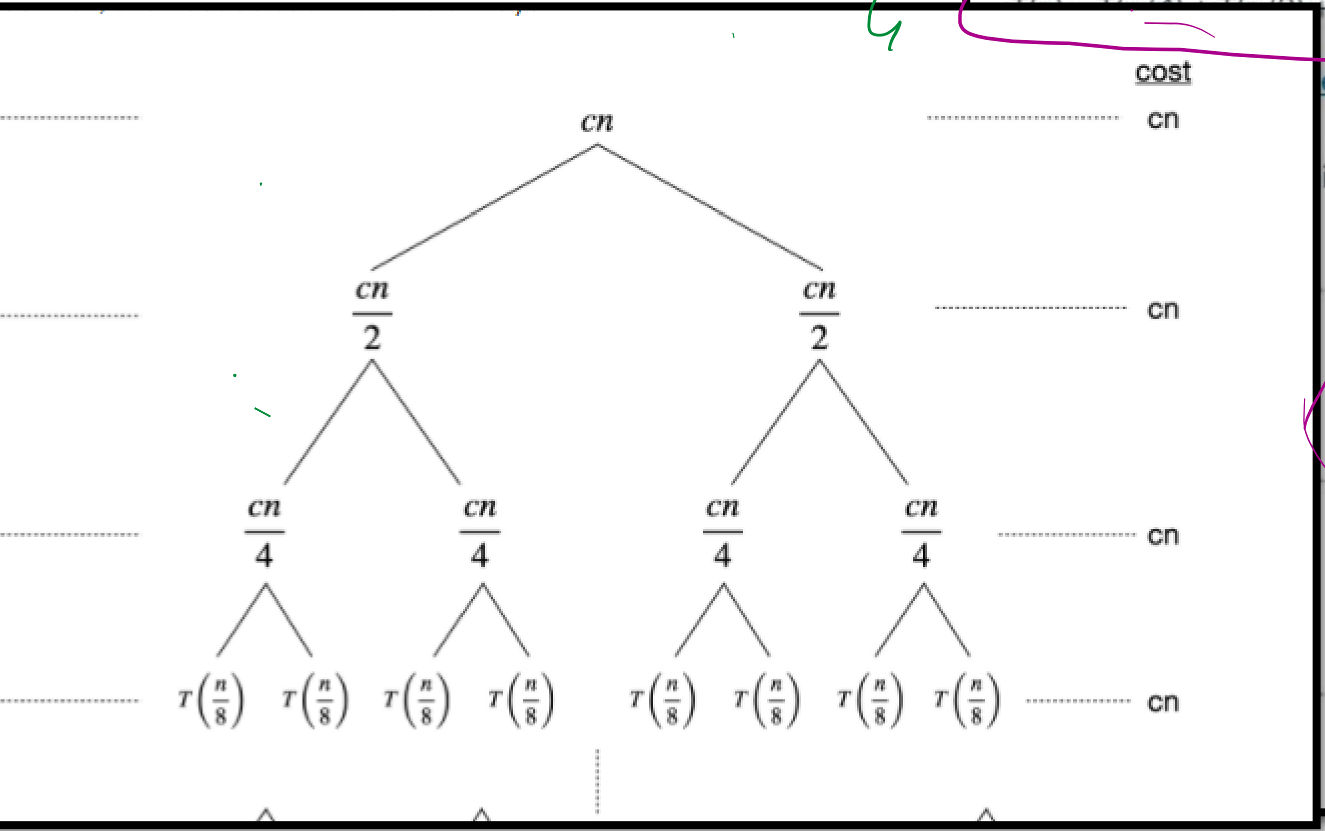
C. T(m) = 2T(m-1) ✗

D. None

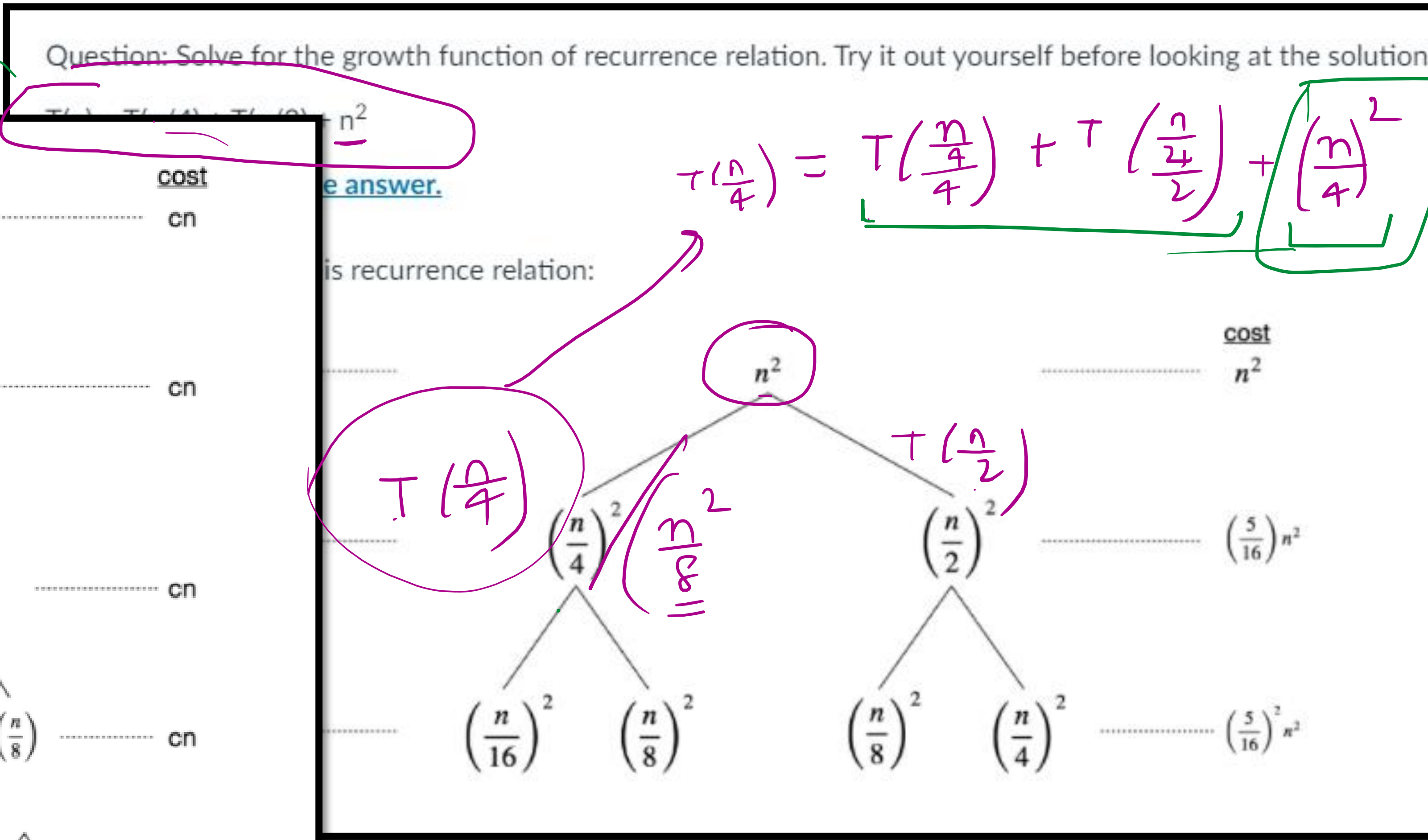
Survey Comments

- can you go over the example T(n) = T(n/4) + T(n/2) + n^2? when n^2 is divided by 4 why does it become (n/4)^2 and not n^2/8?

$T(n) = 2T\left(\frac{n}{2}\right) + cn$



cost
cn
cn
$\frac{cn}{2}$
$\frac{cn}{2}$
cn
$\frac{cn}{4}$
$\frac{cn}{4}$
$\frac{cn}{4}$
$\frac{cn}{4}$
cn
$T\left(\frac{n}{8}\right)$ $T\left(\frac{n}{8}\right)$ $T\left(\frac{n}{8}\right)$ $T\left(\frac{n}{8}\right)$ $T\left(\frac{n}{8}\right)$ $T\left(\frac{n}{8}\right)$ $T\left(\frac{n}{8}\right)$ $T\left(\frac{n}{8}\right)$
cn



Question: Solve for the growth function of recurrence relation. Try it out yourself before looking at the solution

T(n) = T(n/4) + T(n/2) + n²

e answer.

is recurrence relation:

cost
n^2
n^2
$\left(\frac{n}{4}\right)^2$
$\left(\frac{n}{2}\right)^2$
$\left(\frac{5}{16}\right)n^2$
$\left(\frac{n}{16}\right)^2$
$\left(\frac{n}{8}\right)^2$
$\left(\frac{n}{8}\right)^2$
$\left(\frac{n}{4}\right)^2$
$\left(\frac{5}{16}\right)^2 n^2$
$T\left(\frac{n}{4}\right)$
$T\left(\frac{n}{2}\right)$
$\left(\frac{n}{8}\right)^2$

$T\left(\frac{n}{4}\right) = \underline{T\left(\frac{\frac{n}{4}}{4}\right) + T\left(\frac{\frac{n}{4}}{2}\right)} + \boxed{\left(\frac{n}{4}\right)^2}$

Survey Comments

$$a^{\log_c b} = b^{\log_c a}$$

- Can you go over how to simplify the following 2^log(n)?

$$2^{\log_2 n} = n^{\log_2 2} = n^1 = n.$$

- What does it mean when it says that when T(n) is executed it spends cn cost of execution for the current execution from the following sentence? "Looking at the recurrence equation
- T(n) = 2T(n/2) + cn, we can say that when T(n) is executed it spends cn cost of execution for the current execution and then makes two recursive calls expressed by T(n/2). We can represent this in the tree form as:“

Substitution Method

Substitution method, Substitution method example (I am understanding it but it is good to see more). substitution method and recursion tree method examples with more complex T(n) functions.

$$T(n) = 3T\left(\frac{n}{2}\right) + C \quad ; T(1) = c_1$$

$$T(n) = 3T\left(\frac{n}{2}\right) + c$$

$$\hookrightarrow \left(3T\left(\frac{n}{2^2}\right) + C\right)$$

$$= 3\left[3 \cdot T\left(\frac{n}{2^2}\right) + C\right] + C$$

$$= 3^2 T\left(\frac{n}{2^2}\right) + (3 + 1) C$$

$$\hookrightarrow \left[3T\left(\frac{n}{2^3}\right) + c\right]$$

$$= 3^2\left[3T\left(\frac{n}{2^3}\right) + C\right] + (3+1)C$$

$$= 3^3 T\left(\frac{n}{2^3}\right) + (3^2 + 3 + 1) C$$

$$= 3^k T\left(\frac{n}{2^k}\right) + (3^{k-1} + \dots 3^2 + 3 + 1) C$$

$$\llcorner 1 \Rightarrow n = 2^k = k = \log_2 n$$

$$= 3^k \cdot C + 3^{k-1} \cdot C_1$$

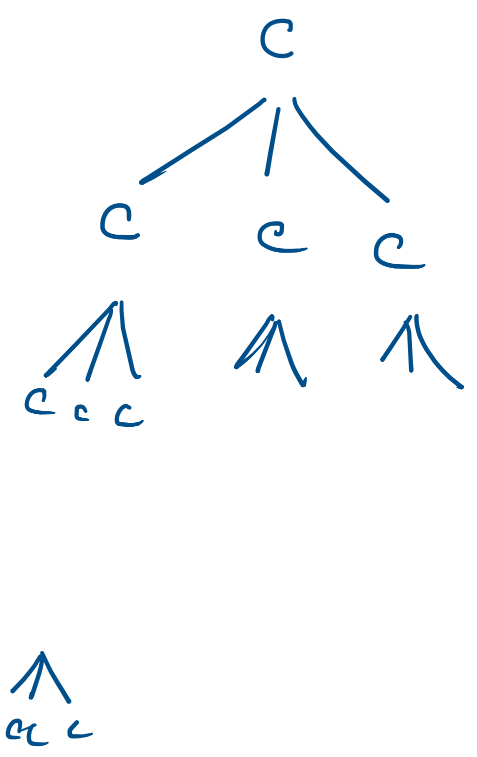
$$= 3^k [C + 3C_1] = 3^{\log_2 n} \cdot C_3 = n^{\log_2 3}$$

Recursion Tree Method

recursion tree method examples with more complex T(n) functions.

$$T(n) = 3T\left(\frac{n}{2}\right) + C \quad ; T(1) = c_1$$

level	T(?)	Cost
0	$T(n)$	C
1	$T(\frac{n}{2})$	$3C$
2	$T(\frac{n}{2^2})$	3^2C
k	$T(\frac{n}{2^k})$	3^kC



$$C + 3C + 3^2C + \dots 3^kC$$

$$= 3^k(C)$$

$$\frac{n}{2^k} = 1 \Rightarrow k = \log n$$

$$= 3^{\log_2 n} \cdot C$$

$$= n^{\log_2 3} \cdot C$$

$$= \Theta(n^{\log_2 3})$$

Masters Method

$\rightarrow$ $\boxed{T(n) = aT\left(\frac{n}{b}\right) + f(n)}$

$\boxed{n^{\log_b a} \ vs \ f(n)}$; f(n) ∈ $\Theta(n^d)$

- Time complexities of the master method.

Case 1: n^d <<< $n^{\log_b a}$
- f(n) grows slower than $n^{\log_b a}$
- T(n) ∈ $\Theta(n^{\log_b a})$

Case 2: n^d ~ $n^{\log_b a}$
- f(n) and $n^{\log_b a}$ have same order of growth
- T(n) ∈ $\Theta(n^d \log n)$

Case 3: n^d >>> $n^{\log_b a}$
- f(n) grows faster than $n^{\log_b a}$
- T(n) ∈ $\Theta(n^d)$

$\rightarrow$ $T(n) = 3T\left(\frac{n}{2}\right) + C \quad ; T(1) = c_1$

$a = 3$

$b = 2$ $\quad f(n) = C$

$n^{\log_2 3}$ vs C

$n^{1.xx}$ vs $c\,n$

$= f(n) = \; ? \quad n^0 = 1$

$= c \cdot 1$

$= n^0 c$ vs $n^{\log_2 3}$

$\Rightarrow \boxed{\Theta\left(n^{\log_2 3}\right)}$

Masters Method

Ex $T(n) = aT\left(\frac{n}{b}\right) + f(n)$

$n^{\log_b a}$ vs $f(n)$; f(n) ∈ $\Theta(n^d)$

•Time complexities of the master method.

A. Case 1: $n^d \lll n^{\log_b a}$
- f(n) grows slower than $n^{\log_b a}$
- T(n) ∈ $\Theta(n^{\log_b a})$

B. Case 2: $n^d \sim n^{\log_b a}$
- f(n) and $n^{\log_b a}$ have same order of growth
- T(n) ∈ $\Theta(n^d \log n)$

C. Case 3: $n^d \ggg n^{\log_b a}$
- f(n) grows faster than $n^{\log_b a}$
- T(n) ∈ $\Theta(n^d)$

$$T(n) = 2T\left(\frac{n}{2}\right) + Cn \quad ; T(1) = c_1$$

$a = 2$

$b = 2$

$f(n) = cn$

$n^{\log_2 2} = n$

n vs cn

$\Theta(n)$ $\quad$ $\Theta(n) \to \Theta(n)$

$n^d = n$

$\Theta(n \log n)$

More Practice Problems:

✓ 1. T(n)=T(n−1)+n
T(0)=1 $\qquad\qquad \Theta(n^2)$

✓ 2. $T(n) = 3T\left(\frac{n}{2}\right) + C \quad ; T(1) = c1$ This is solved in this deck in previous slides.

3. All problems in exploration using other methods ✓

4. CLRS book examples ✓

Divide and Conquer

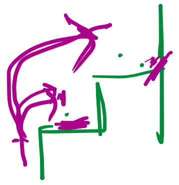
- divide and conquer
  - Quick sort
  - Merge sort
  - Binary search

$\boxed{n\log n}$

Survey Comments

- Can you walk through some recursion functions? Specifically the stair step one from the module?

- You are climbing a staircase. It takes n steps to reach to the top.
Each time you can either climb 1 or 2 steps. You are trying to find out how many distinct ways can you climb to the top. Write a recursive function that takes the number of steps as input and returns the number of distinct ways that you can climb to the top.



$(1+1+1) = 3$

Base case:

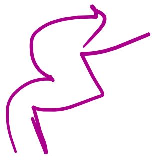
$n=1$

return : 1

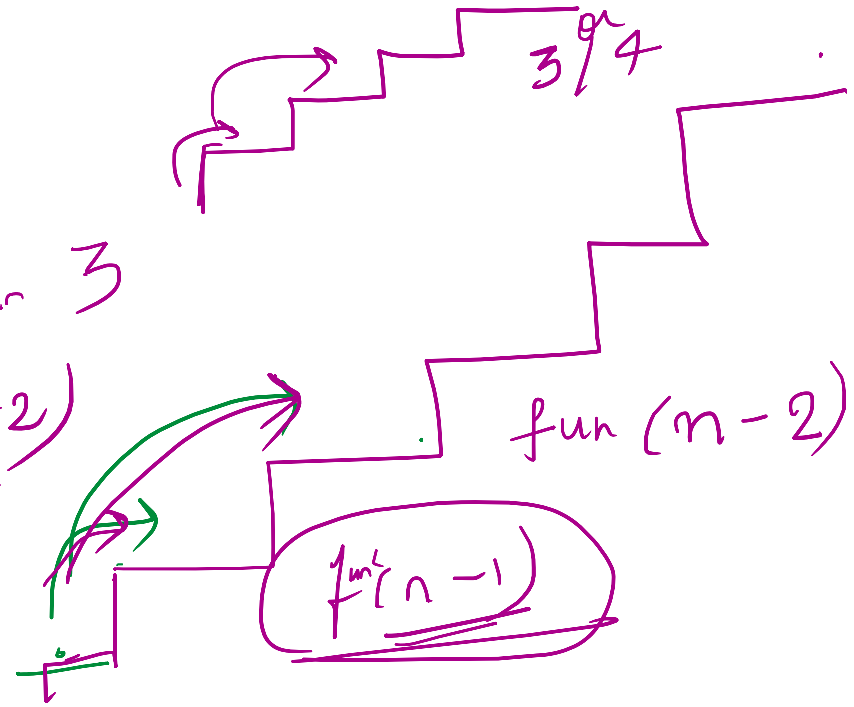
$n=2$

return 3

$\text{fun}^n(n) =$

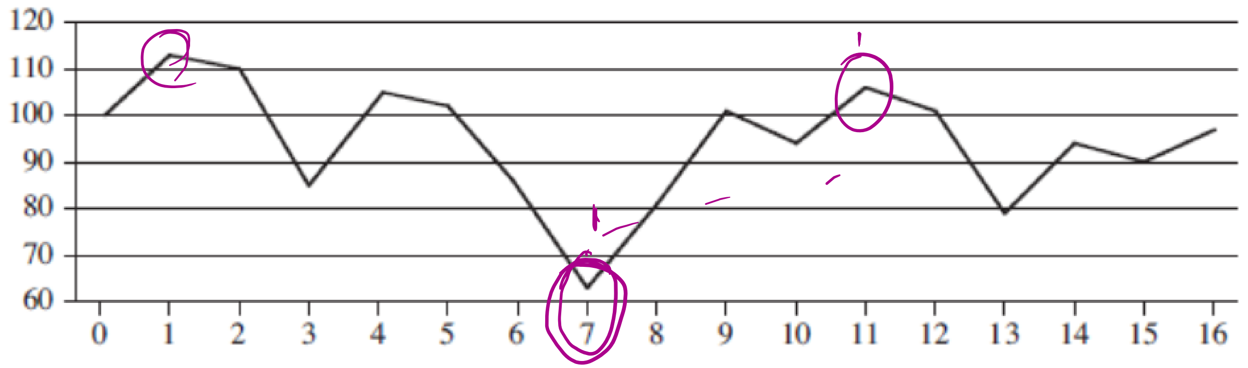


return | $\text{fun}(n-1) + \text{fun}(n-2)$



Additional Practice Problem

Suppose that you been offered the opportunity to invest in the Volatile Chemical Corporation. Like the chemicals the company produces, the stock price of the Volatile Chemical Corporation is rather volatile. You are allowed to buy one unit of stock only one time and then sell it at a later date, buying and selling after the close of trading for the day. To compensate for this restriction, you are allowed to learn what the price of the stock will be in the future. Your goal is to maximize your profit. →
Below figure shows the price of the stock over a 17-day period. You may buy the stock at any one time, starting after day 0, when the price is $100 per share.



Day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Price	100	113	110	85	105	102	86	63	81	101	94	106	101	79	94	90	97
Change		13	−3	−25	20	−3	−16	−23	18	20	−7	12	−5	−22	15	−4	7

$ 43

Optimal solution for the above example: Buy the stock just after day 7 at $63 and sell it after day 11 at $106; giving a profit of $43.