4 Analysis of Recursive Algorithms

Tuesday, September 5, 2023 12:14 PM

$n!$ $fib(n)$

- factorial

$$F(n) = n! = 1 \cdot 2 \cdot \ldots \cdot (n-1) \cdot n = (n-1)! \cdot n$$

$$0! = 1$$

or

$$F(n) \begin{cases} F(n-1) \cdot n & n > 0 \\ 1 & n = 0 \end{cases}$$

E.G.

```
FUNCTION fact(n)
    if n=0
        return 1
    else
        return fact(n-1) * n
```

basic operation

- $M(n) = \underbrace{M(n-1)}_{\text{no. of multiplications needed to compute } F(n-1)} + \underbrace{1}_{\text{multiply } F(n-1) * n}$

called "recurrances" or "recurrance relations"
- Big in discrete math and Analysis of Algorithms.

Note : infinite solutions

n	...	32	33	34	35	36	37	....
M(n)	...	8	9	10	11	12	13	....

$$M(33) = M(32) + 1$$
$$M(34) = M(33) + 1$$
$$\ldots\ldots$$

- $M(0) = 0$ ← initial condition.

Collecting:
- $M(n) = M(n-1) + 1$
- $M(0) = 0$

Let us solve in a systematic way: backward subtitution

$$M(n) = M(n-1) + 1$$

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

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

$M(n-2) = M(n-3) + 1$

$$= M(n-3) + 1 + 1 + 1$$

after i subtitution....

$$= M(n-i) + i$$

if $i = n$

$$= M(n-n) + n$$
$$= M(0) + n$$
$$= 0 + n$$
$$M(n) = n \in \Theta(n)$$

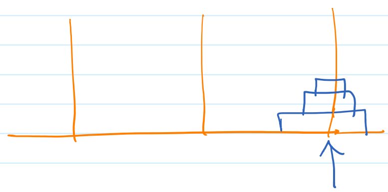
- GENERAL PLAN FOR ANALYZING THE TIME EFFICIENCY OF RECURSIVE ALGORITHMS

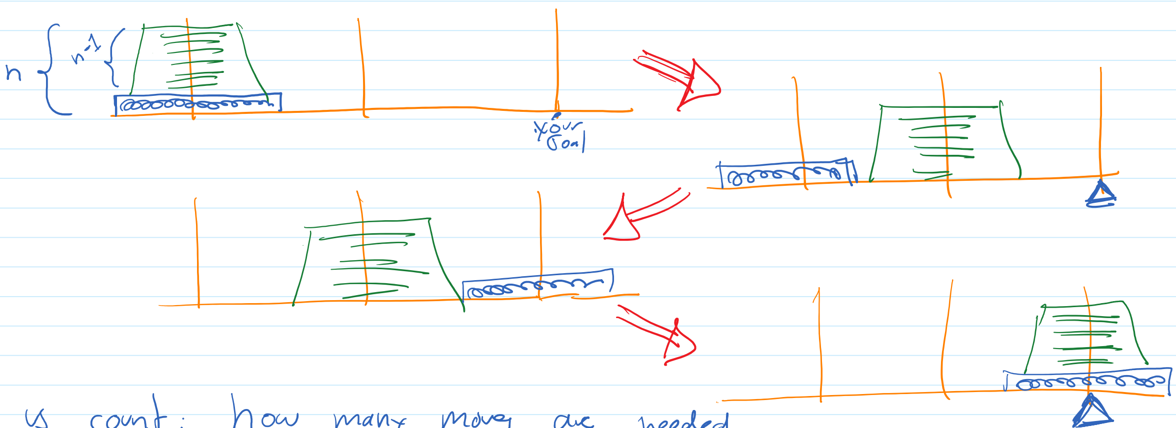
$M(n) = n \in \Theta(n)$

- GENERAL PLAN FOR ANALYZING THE TIME EFFICIENCY OF RECURSIVE ALGORITHMS

1.- Decide on the parameter(s) for the size of the input
2.- Identify basic_operation.
3.- Check whether the number of times the basic operation is executed depends only on the size of the input or on other factors
4.- Set up a recurrance relation, with an initial condition for the number of times the basic_operation is executed.
5.- Solve the recurrance (using known results) or at least, find the order-of-growth of the solution.

E.G 2 Towers of Hanoi





Let us count: how many moves are needed to move n discs.

- $M(n) = M(n-1) + 1 + M(n-1)$ recurrance
- $M(1) = 1$ initial condition.

Solve by backwards substitution.

tip

$$\sum_{i=0}^{u} a^i = \frac{a^{u+1} - 1}{a - 1}$$

$$\sum_{i=0}^{u} 2^i = 2^{u+1} - 1$$

$M(n) = 2 \cdot M(n-1) + 1$

$M(n-1) = 2 \cdot M(n-2) + 1$

$= 2 \cdot [2 \cdot M(n-2) + 1] + 1$

$= 2^2 \cdot M(n-2) + 2 + 1$

$M(n-2) = 2 \cdot M(n-3) + 1$

$= 2^2 \cdot [2 \cdot M(n-3) + 1] + 2 + 1$

$= 2^3 \cdot M(n-3) + 2^2 + 2 + 1$

$M(n-3) = 2 \cdot M(n-4) + 1$

$= 2^3 \cdot [2 \cdot M(n-4) + 1] + 2^2 + 2 + 1$

$= 2^4 \cdot M(n-4) + 2^3 + 2^2 + 2 + 1$

after i subtitutions

$$= 2^i \cdot M(n-i) + 2^{i-1} + 2^{i-2} + 2^{i-3} + \cdots + 2^1 + 2^0$$
$$= 2^i \cdot M(n-i) + \sum_{i=0}^{i-1} 2^i$$
$$= 2^i \cdot M(n-i) + 2^i - 1$$

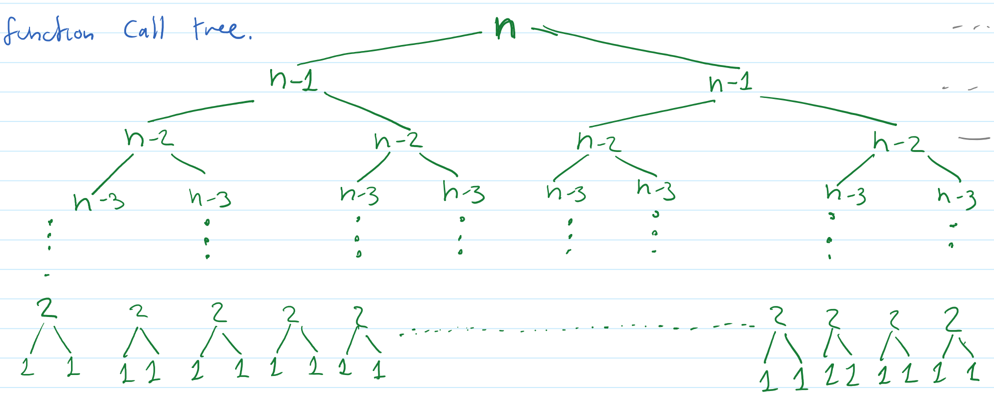
Let $i = n-1$ to reach initial conditions

$$= 2^{n-1} \cdot \underbrace{M(n-(n-1))} + 2^{n-1} - 1$$
$$= 2^{n-1} \cdot 1 + 2^{n-1} - 1$$
$$= 2 \cdot 2^{n-1} - 1$$
$$= 2^n - 1 \in \Theta(2^n)$$

- USING THE CALL TREE

Hanoi: (n)
 Move $n-1$ disks : Hanoi $(n-1)$
 Move Largest disk
 Move $n-1$ disks : Hanoi $(n-1)$

function Call tree.



$1 = 2^0$
$2 = 2^1$
$4 = 2^2$
$8 = 2^3$
$= 2^{n-2}$
$= 2^{n-1}$

$2^n - 1 \in \Theta(2^n)$ $\qquad 2^n - 1 \Leftarrow \sum_{i=0}^{n-1} 2^i$

- ANOTHER EXAMPLE = BinRec

FUNCTION BinLenRec(n)
 // returns the number of binary digits in n's binary representation.
 // PRE: n is a positive integer.
 IF $n=1$ THEN RETURN 1
 ELSE
 RETURN BinLenRec($\lfloor n/2 \rfloor$) + 1

$n = 5 \Rightarrow 3$
$n = 2 \Rightarrow 2$
$n = 1 \Rightarrow 1$

Analysis:
 basic operation : addition.

$$A(n) = \underset{\text{one addition}}{1} + \underbrace{A(\lfloor n/2 \rfloor)}_{\text{additions needed for } A(\lfloor n/2 \rfloor)} \leftarrow \text{recurrance.}$$

$$A(1) = 0 \leftarrow \text{initial condition.}$$

BinLenRec(5) = 3
$5 = 101_b$

Lets make our life easier:
 Let $n = 2^k$; $k = \log_2 n$

Under very broad conditions, ("Smoothness Rule") we can safely make this substitution.

- $A(2^k) = A(2^{k-1}) + 1$
- $A(1) = 0$

substituted:

sub. $A(2^{k-1}) = A(2^{k-2}) + 1$

sub $A(2^{k-2}) = A(2^{k-3}) + 1$

sub $A(2^{k-3}) = A(2^{k-4}) + 1$

$$A(2^k) = A(2^{k-2}) + 1 + 1$$
$$A(2^k) = A(2^{k-3}) + 1 + 1 + 1$$
$$A(2^k) = A(2^{k-4}) + 1 + 1 + 1 + 1$$

after i substitutions

$$A(2^k) = A(2^{k-i}) + i$$

to get to initial conditions: $i = k$

$$A(2^k) = A(2^{k-k}) + k$$
$$= A(2^0) + k$$
$$= 0 + k$$
$$A(2^k) = k$$

hence $A(n) = \log_2 n \in \Theta(\log n)$

— EOF —