6 Decrease and Conquer

Tuesday, October 3, 2023 10:51 AM

An algorithm design technique.

- exploit the relationship between the solution to a problem instance with the solution to a smaller instance

Variants:
- Decrease by 1
- Decrease by a constant factor. (2)
- Variable size decrease.

e.g. problem: compute a^n

$a^n \longleftrightarrow a^m \quad m < n$

- Decrease by 1: a^m where $m = n-1$ then. $a^n = a * a^{n-1}$.

$a^0 = 1$

```
FUNCTION pow(a, n)
    IF n = 0
        RETURN 1
    ELSE
        RETURN a * pow(a, n-1)
```

top-down approach.

- bottom up: approach.

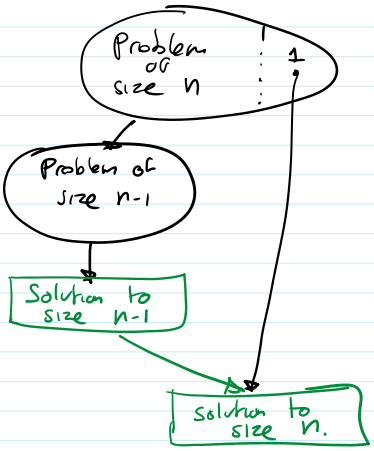
start by computing a^0 and build up to a^n

```
FUNCTION pow(a, n)
    k ⇐ 0
    res ⇐ 1
    WHILE k ≤ n DO
        res ⇐ res * a
        k ⇐ k + 1
    RETURN res.
```

res [$a^0 \to a^1 \to a^2 \to a^3 \dots a^k$

Analysis: basic-operation *

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



- Decrease by a constant factor: (2)

Problem: compute a^n

Given $a^{n/2}$ then. $a^n = (a^{n/2})^2 = a^{n/2} \cdot a^{n/2}$ if n is even.

Problem: compute a^n

Given $a^{n/2}$ then $a^n = (a^{n/2})^2 = a^{n/2} \cdot a^{n/2}$ if n is even.

$a^{\lfloor n/2 \rfloor}$

$n = 5$ a^2 then $a^n = a^{\lfloor n/2 \rfloor} \cdot a^{\lfloor n/2 \rfloor} \cdot a$ if n is odd.

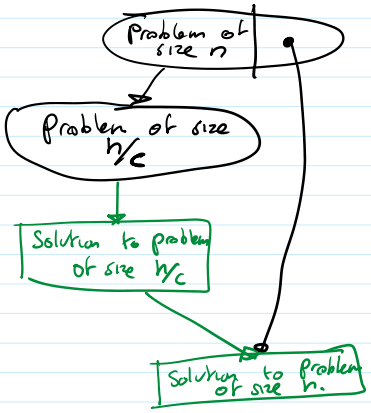
$a^5 = a^2 \cdot a^2 \cdot a$

```
FUNCTION pow(a, n)
    IF n = 0
        RETURN 1
    ELSE
        b = pow(a, ⌊n/2⌋)
        IF n is even THEN
            RETURN b*b
        ELSE
            RETURN b*b*a
```

Analysis:

	M(n)
pow(a, n)	2
pow(a, ⌊n/2⌋)	2
pow(a, ⌊n/4⌋)	2
pow(a, ⌊n/8⌋)	2
⋮	
pow(a, 0)	0

$2 \cdot \log n \in \Theta(\log n)$



- Decrease by variable size:

  e.g. Euclids Algorithm.

  Process to compute the G.C.D. of two numbers. a. b

  - $GCD(a, b) = GCD(a \bmod b, b)$ where $a > b$ $= GCD(a - b, b)$
  - $GCD(a, a) = a$

a	b	c
-	-	-
3	4	5
-	-	-
-	-	-
-	-	-

```
FUNCTION gcd(a, b)        PRE: a ≥ b
    IF a mod b = 0
        RETURN b
    ELSE
        RETURN gcd(b, a mod b)
```

```
FUNCTION gcd(a, b)
    WHILE a != b DO
    { IF a > b THEN
        a ← a - b
```

```
WHILE a != b DO
    IF a > b THEN
        a ← a - b
    ELSE
        b ← b - a
RETURN a.
```

- PROBLEM : Sorting :

Intuition: Sort an array:

- Supose somebody can sort arrays of size n-1 call that person:

insert left over element into already sorted part.

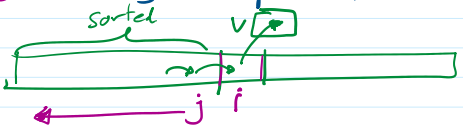
- Simplest case.


size 1 already sorted.

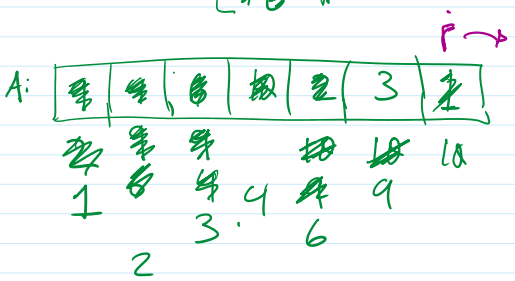

size 2

```
FUNCTION Insertion_Sort(A[0...n])
    FOR i ← 1 TO n-1 DO
        v ← A[i]
        j ← i-1
        WHILE j ≥ 0 AND A[j] > v DO
            A[j+1] ← A[j]
            j ← j-1
        A[j+1] ← v
```

Note: i : A[0..i-1] is already sorted
A[i], v : element we trying to insert
j : searching for place to insert



TRACE:



Analysis: basic operation.

j = i-1, i-2, i-3, i-4, , 0

$$C(n) = \sum_{i=1}^{n-1} \sum_{j=0}^{i-1} 1 = \sum_{i=1}^{n-1} i = \frac{(n-1)n}{2} \in \Theta(n^2)$$

Ponder:

- Worst-case scenario for insertion sort: array sorted in decreasing order.
- best-case:

array sorted in decreasing order.

- best-case:
  array is already sorted.

$$C_{best}(n) = \sum_{i=1}^{n-1} 1 = n-1 = \in \Theta(n)$$

EOF