6 Decrease and Conquer

Tuesday, October 8, 2024 11:41 AM

Algorithm Design Technique.

- Exploit the relationship between a large problem and a smaller instance of the problem

Variants:
- Decrease by 1
- Decrease by c: constant factor
- Variable size decrease.

- Decrease by 1
  - EG: Power

$$a^n = \underbrace{a \cdot a \cdot a \cdot \ldots \cdot a}_{n\text{-times}}$$

How: a^n relate? a^m $m < n$

- $a^n = a \cdot a^{n-1}$
- $a^0 = 1$

```
FUNCTION pow(a, n)
    if n = 0
        Return 1
    else return a * pow(a, n-1)
```

top down approach

- What about a bottom up approach? Start at a^0 and build up to a^n

```
FUNCTION pow(a, n)
    k ← 0
    r ← 1 // a^0
    While k ≤ n do
        k ← k+1
        r ← r * a // r = a^k
    return r // a^n
```

Analysis

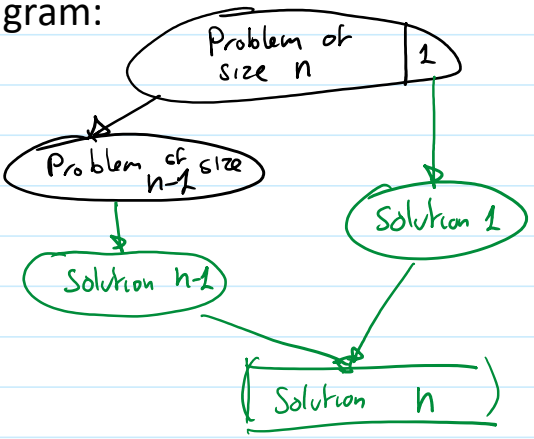
basic_operation

Analysis

basic_operation

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

- Diagram:



- Decrease by a constant factor (2)

a^n relate? a^m $\qquad$ $a^n \overset{?}{\leftrightarrow} a^{n/2}$

$$a^n = a^{\lfloor n/2 \rfloor} \cdot a^{\lfloor n/2 \rfloor} \quad \text{when } n \text{ is even}$$

$$a^n = a^{\lfloor n/2 \rfloor} \cdot a^{\lfloor n/2 \rfloor} \cdot a \quad \text{when } n \text{ is odd}$$

e.g. $a^6 = a^3 \cdot a^3$

$a^7 = a^3 \cdot a^3 \cdot a$

```
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

basic_operation *

using the call tree.

pow(a,n) 2

pow(a,⌊n/2⌋) 2

pow $(a, \lfloor \frac{n}{2} \rfloor)$ 2

pow $(a, \lfloor \frac{n}{4} \rfloor)$ 2

pow $(a, \lfloor \frac{n}{8} \rfloor)$ 2

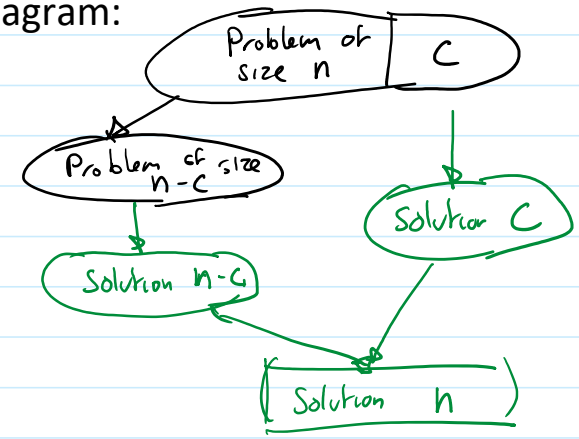
⋮

pow $(a, 0)$ 0

$2 \cdot \log_2 n = M(n) \in \Theta(\log n)$

eg. a^{100} → 100 multiplications
→ 14 multiplications

- Diagram:



- Decrease by variable size:

e.g. Euclids Algorithm:

Compute the G.C.D. of two numbers a b

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

- $GCD(a,b) = GCD(a \bmod b, b)$
  $= GCD(b \bmod (a \bmod b), a \bmod b)$
  $= GCD((a \bmod b) \bmod (b \bmod (a \bmod b)), b \bmod (a \bmod b))$
  ⋮
  $= GCD(\ ,\ )$

$= GCD(c, c)$

Pseudocode

```
FUNCTION gcd(a, b)   // a ≥ b
  IF a mod b = 0
     Return b
  else
     Return gcd(b, a mod b)
```

Trace.

```
gcd ( 521 , 67 )
gcd ( 67  , 52 )
gcd ( 52  , 15 )
gcd ( 15  , 7  )
gcd ( 7   , 1  )
return 1
```

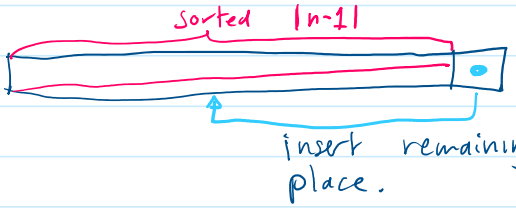
- Decrease by 1
  - EG: Sorting

Intuition.

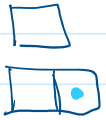
size n.

Suppose somebody can solve a smaller problem:
i.e. sort an array of size n-1
How can we use that person?

1. use them to sort the array from [0 to n-2]

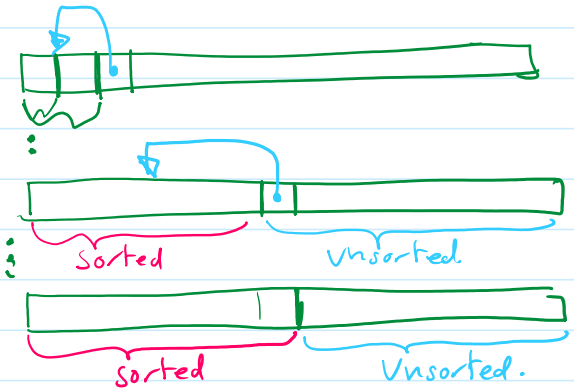


Simplest case:



size 1 already sorted

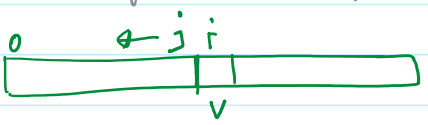
size 2 either swap, or keep.



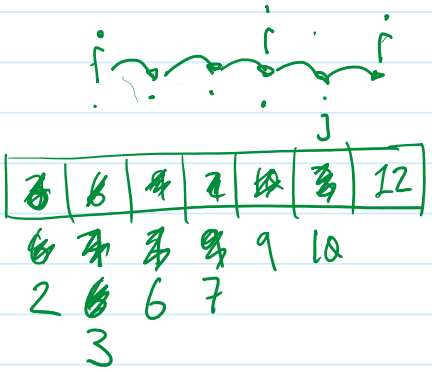
FUNCTION InsertionSort (A [0..n-1])

```
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: A[0..i-1] is already sorted.
A[i] : element we are trying to insert.
j : index j searches for a place to insert.



Trace



Trace:
- https://visualgo.net/en

Analysis: basic-operation

Consider j : i-1, i-2, i-3, ..., 0

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

Think:

- worst_case scenario: array is sorted in reverse order.
- best_case scenario: array is already sorted

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

EOF