1) Lecture 1-3

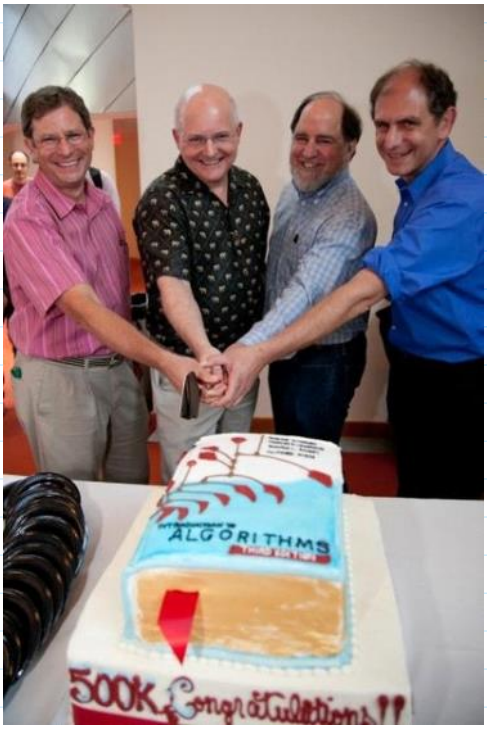
28 July 2024 13:05

Lab 524

Time 11-1 pm

1. Big Oh notation (Asymptotic analysis for computing running time/space)
2. Basic Data structure - List, Stack, Queue, Trees
3. Dictionaries, Binary Search tree, Balanced Search tree, Btree, Hash tables Heaps
4. Graph - Basic representation of graph, BFS/DFS Graph traversal algo Computing min spanning tree, shortest path

Binomial Heaps, Fibonacci heaps ..



Introduction to Algorithms

Coreman
Leiserson
Rivest
Stein

CLRS

Implementation will be in C language

Evaluation policy

①	Assignment	(3-4 Theory + 3-4 Coding)	30%
②	Lab assignment		10%
③	Mid Sem		20%
④	End Sem		25%
⑤	Surprise Quiz		10%
⑥	Attendence (lab + class)		5%

Algorithm: Outline; essence of a computational procedure; step-by-step instructions

Program: An implementation of algorithm in some programming language

Data structure: Facilates to organize data for given problem

Algorithmic Problem:

Algorithmic Problem:

Specification of Input → (Algorithm) → Specification of output as fn of input

Ex: A set of 100 real no. as task is to compute max of those nos

$S = \{1, 3, 5, \ldots, 99\}$ $|S| = 100$

Infinetly many input instance possible

" " algorithms possible for a given problem

What is a good algorithm

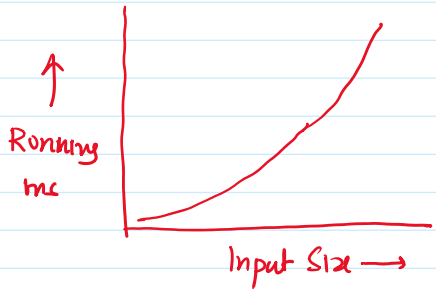
Efficiency — Running time
Memory / Space

Input: # of data points
# of bytes used to store data pts

Measuring running time empirically

System clock

(empirical running time)



Limitation of empirical running time

① Error in measuring time

② Varies based to hardware / software plateform used

③ Experiments can be ran only on a limited set of input, and this may not be indicative about all those inputs that were not considered

④ Implement the algorithm.

Remedy: Develop a general methology for analysing the running time of algo.

- Able to comment on running time by looking at high level description

- Able to comment on running time by looking at high level description of algorithm (instead of its implementation)
- take care of all possible inputs
- It should be indep. of hardware / software platform used.

Ex: computing largest no. of a given input set.

Input: A set of n numbers represented in array A

Output: The max element of A.

```
current max ← A[0]
for i = 1 to n-1 do
    if current max < A[i]
    then current max ← A[i]
return current max.
```

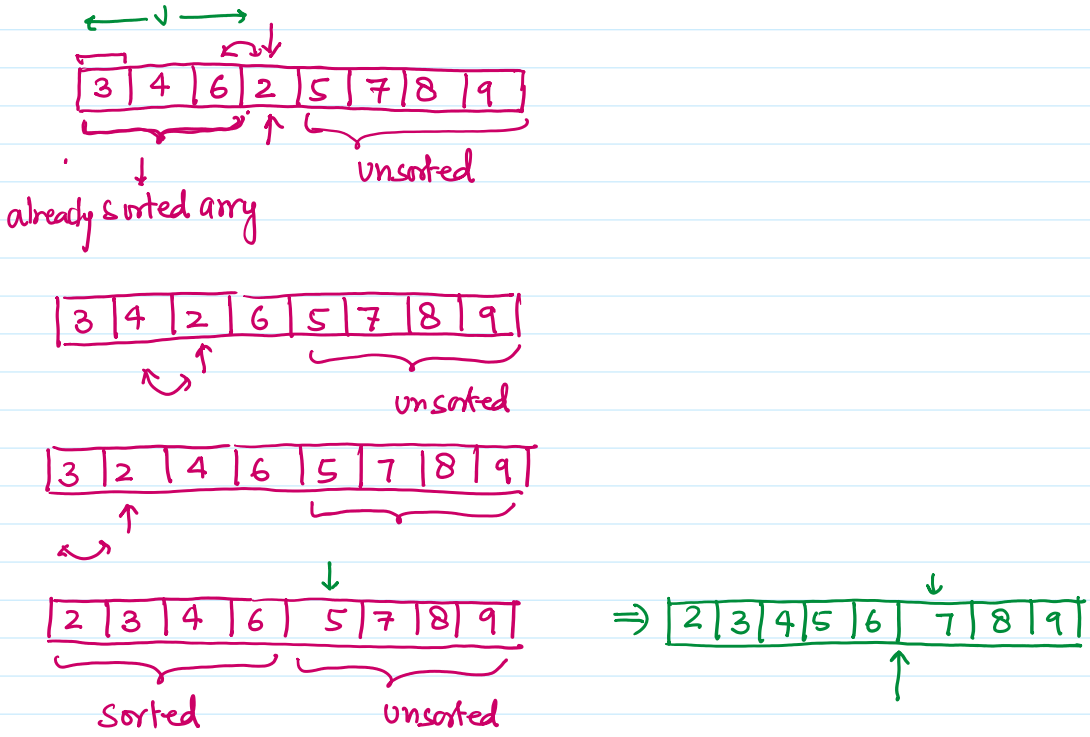
Primitive operation - Data movement (assignment)
- arithmetic and logic operation (addition, subtraction, comparison)
- Control operation (branching — if else
  call return
  loop
  :
)

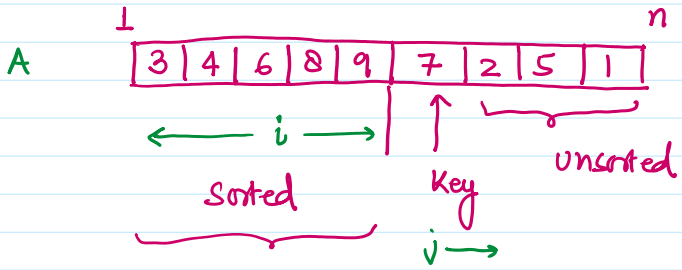
By inspecting the Algo we need to count # of primitive operation used.

Lecture 2

Input	Output
Sequence of numbers	$(b_1, b_2 \ldots b_n)$ is a permutation of $(a_1 \ldots a_n)$
$a_1 \ldots a_n$	$b_1 \leq b_2 \ldots \leq b_n$ Set of n no.

- Start empty handed
- Insert a card in the right position of already sorted card in hand
- continue until all card are sorted





		cost	# of iteration
For $j \leftarrow 2$ to n do	①	c_1	$n-1$
Key $\leftarrow$ A[j]	②	c_2	$n-1$
/* Insert A[j] into sorted seq of A[1..j-1]			
$i \leftarrow j-1$	③	c_3	$n-1$
while $i>0$ and $A[i] > key$	④	c_4	$\sum_{j=2}^{n} t_j$
do A[i+1] $\leftarrow$ A[i]	⑤	c_5	$\sum_{j=2}^{n} (t_j - 1)$
$i \leftarrow i-1$	⑥	c_6	$\sum_{j=2}^{n} (t_j - 1)$
A[i+1] $\leftarrow$ key	⑦	c_7	$(n-1)$

A[i+1] ← key (7) c_7 $(n-1)$

t_j := time to put j^{th} element at appropriate position in $A[1 \ldots j-1]$

Total running time = $c_1(n-1) + c_2(n-1) + c_3(n-1) + c_4 \sum_{j=2}^{n} t_j + (c_5+c_6)\sum_{j=2}^{n}(t_j-1) + c_7(n-1)$

$= n(c_1+c_2+c_3+c_7-c_5-c_6) + \sum_{j=2}^{n} t_j (c_4+c_5+c_6) - (c_1+c_2+c_3-c_5-c_6+c_7)$

$c' \qquad c'' \qquad c'''$

$= c'n + c''\sum_{j=2}^{n} t_j - c'''$

When array is already sorted $t_j = 1 \Rightarrow$ Time $= c'n + c''(n-1) - c'''$

$:= f'(n)$ Best case

linear fn of n

When array is sorted in reverse order $t_j = j$

Time $= c'n + c''\sum_{j=2}^{n} j - c'''$

$= c'n + c''\left[\frac{n(n+1)}{2} - 1\right] - c'''$

$= f''(n^2)$ Quadratic in n

Worst Case ||

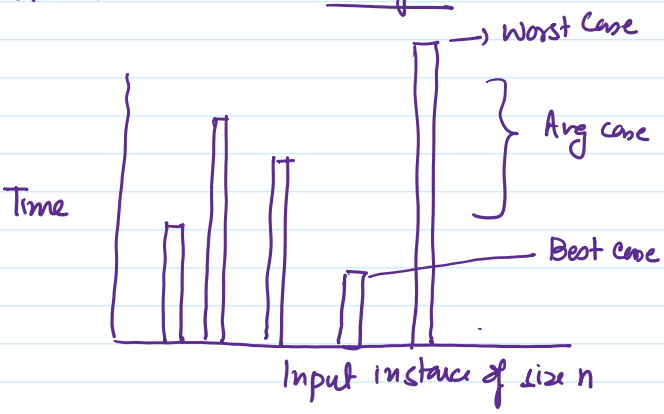
Case when $t_j = j/2$

Time $= c'n + c''\sum_{j=2}^{n} j/2 - c'''$

$:= f'''(n^2)$ Quadratic in n Average Case ||

→ Worst Case

$:= f'''(n^2)$ Quadratic in n Average case ||



Worst case is generally used. for analysis.

Avg case is often as bad as worst case.

Computing Avg case can be difficult - computing expected running time

└ Knowledge of prob distribution of input instance

Lecture 3

$$3n^3 + 4n\log^2 n + 5\log n + 10000$$

Two Features : (i) we give more importance as $n \to \infty$

(ii) $5n^3 + 8n^2 + 10n + 100$
$10n^3 + 3n^2 + 2n + 5$
} Should belong to same class

ignore constt.

Running time	Time in Sec $n=10$	$n=100$	$n=1000$	$n=10^5$
$5\log n$	11×10^{-9}	23×10^{-9}	37×10^{-9}	57×10^{-9}

$2n^2$	2×10^{-7}	2×10^{-5}	0.002	20 Sec
$5n^6$	0.005	5000 1.38 Hr	5×10^9 57 870 days	5×10^{21} 5.78×10^{16} days
2^n	10^{-6} Sec	$\approx 4 \times 10^{48}$ yrs	3×10^{284} yrs	

X

i7 10^7 Instructions per sec

Asymptotic Notation

Big Oh notation: Asymptotic Upper bound

For a given fn $g(n)$, we define $O(g(n))$ as set of fn

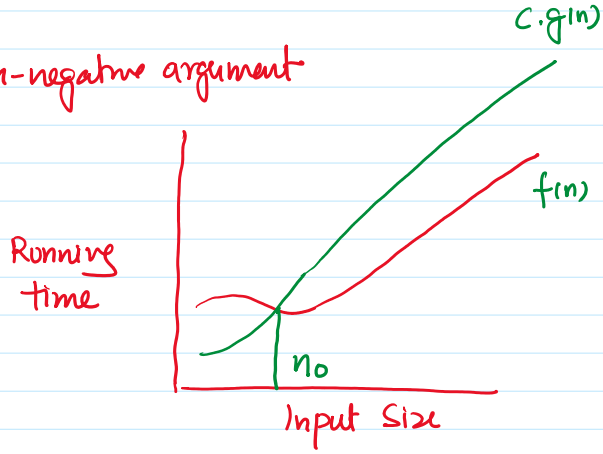
$O(g(n)) = \{ f(n) : \exists$ a positive constt c and n_0 s.t

$0 \leq f(n) \leq c\, g(n), \forall n \geq n_0 \}$ —①

$f(n)$: running time obtained by counting # primitive ops

$g(n)$: clean fn of n

$f(n)$ & $g(n)$ are non negative fn over non-negative argument

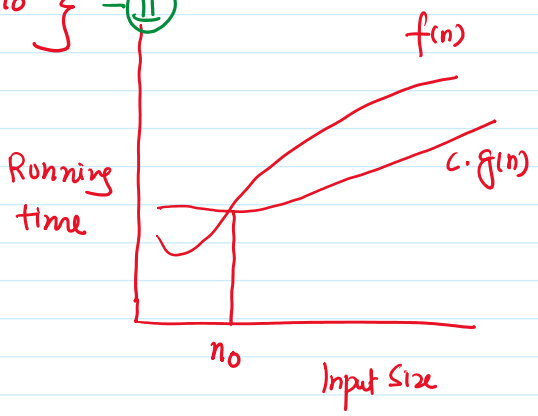


Ω notation: asymptotic lower bound:

For a given function $g(n)$, we define $\Omega(g(n))$ as

$\Omega(g(n)) = \{ f(n) : \exists$ a positive constt c and n_0 s.t

$0 \leq c \cdot g(n) \leq f(n), \quad \forall n \geq n_0$ } — (II)



Ex:1 $f(n) = 10n^2 + 5n + 100$

Aim is to find $g(n)$, c, n_0 s.t Eqn (i) satisfies

$\leq 10n^2 + 5n^2 + 100n^2$

$= 115n^2$

$n_0 = 1$, $c = 115$

$f(n) = O(n^2)$

$f(n) \in O(n^2)$ ✓

$f(n) = O(n^2)$ Abuse of notation

$f(n) = 10n^2 + 5n + 100$

$\geq 10n^2$ $\quad n_0 = 1$, $c = 10$

$f(n) = \Omega(n^2)$

Ex:2 $f(n) = 10n^3 + n \log n$

$f(n) \leq 10n^3 + n^3$

$f(n) \leq 11n^3$ $\quad \because \log n \leq n^2 \ \forall n \geq 1$

$n_0 = 1$, $c = 11$

$f(n) = O(n^3)$

$f(n) = 10n^3 + n \log n$

$f(n) \geq 10n^3$ $\quad n_0 = 1$, $c = 10$

$f(n) = \Omega(n^3)$

Ex: $f(n) = 1 + \frac{1}{n}$ $\quad f(n) \geq 1$

$f(n) \leq 2$ $\quad n_0 = 1$

$f(n) = O(1)$ $\qquad$ $f(n) = \Omega(1)$

Θ notation: f, g are non negative fn over non-negative arguments

$\Theta(g(n)) = \{ f(n) \mid \exists$ +ve const c_1, c_2 and n_0 s.t

$0 \leq c_1 g(n) \leq f(n) \leq c_2 \cdot g(n) \quad \forall n \geq n_0 \}$

Small oh notation

$o(g(n)) = \{ f(n) : \forall$ const $c > 0$, $\exists$ a const $n_0 > 0$

s.t $0 \leq f(n) < c \cdot g(n) \quad \forall n \geq n_0 \}$ — (III)

$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$

Ex: $2n \stackrel{?}{=} o(n^2)$ $\qquad$ $\lim_{n \to \infty} \frac{2n}{n^2} = 0$

$2n^2 \stackrel{?}{=} o(n^2)$ $\qquad$ $\lim_{n \to \infty} \frac{2n^2}{n^2} = 2 \neq 0$

$f(n)$ $\qquad$ $g(n)$

$2n^2 \leq c \cdot n^2 \quad \forall c$ ✗

Choose $c = 1$

$2n^2 = O(n^2)$

$\because 2n^2 \leq 3n^2 \quad n_0 = 1$

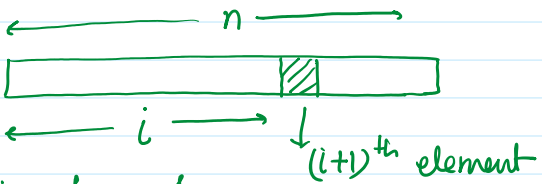
$f(n) = O(g(n))$	$f(n) = \Omega(g(n))$	$f(n) = o(g(n))$	$f(n) = \omega(g(n))$
$\lim_{n\to\infty} \frac{f(n)}{g(n)} \leq$ const	$\lim_{n\to\infty} \frac{g(n)}{f(n)} \leq$ const	$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$	$\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$
f is no larger than g	f is atleast as large as g.	f is strictly smaller than g	f is strictly larger than g

Ex. $5n+3 \stackrel{?}{=} \omega(n)$ ✗

$5n+3 = \omega(1)$ ✓

Lecture -4

Insertion Sort [Bounding worst case running time]



$T(n)$:= worst case running time of insertion sort on input of size n

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

$= T(n-2) + c(n-2) + c(n-1)$

$= T(n-3) + c((n-3) + (n-2) + (n-1))$ $\quad T(1) = c_1$

$\vdots$

$= c(\underbrace{(n-1) + (n-2) + (n-3) + (n-4) \cdots 2}_{\text{poly of deg 2}}) + c_1$ $\quad c, c_1$ are consts

$= c\left[\frac{n(n-1)}{2}\right] + c'$

$= \frac{c}{2}n^2 - \frac{c}{2}n + c'$

$= O(n^2) = \Theta(n^2)$

$= \Omega(n^2)$

$\frac{c}{2}n^2 - \frac{c}{2}n + c'$

$< \frac{c}{2}n^2 + c'$

$< \frac{c}{2}n^2 + c'n^2$

$= (c + c')n^2$

$= \Omega(n^2)$

$< \frac{c}{2} n^2 + c' n$

$= \left(\frac{c}{2} + c'\right) n^2$

$\frac{c}{2} n^2 - \frac{c}{2} n + c'$

$> \frac{c}{2} n^2 - \frac{c}{2} n$

$> \frac{c}{4} n^2$

$f(n) = \sum_{i=1}^{k} a_i n^i = \Theta(n^k)$ $a_i \in \mathbb{R}$

Bounding best case running time

$T(n)$:= best case running time of insertion sort for input of size n

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

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

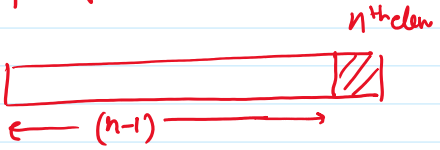
$\vdots$

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

$= c_1 + (n-1)c$

$= O(n)$ $= \Theta(n)$

$= \Omega(n)$



Linear Search

Input: An unsorted array A of size n, query element q

Output: if $q \in A$?

$T(n)$:= worst case running time on input of size n

```
for i = 1 to n
    if (A[i] == q)
        return 'Sucess'

if (i == n)
    return "Fail"
```

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

$T(n) = \Theta(n)$

Binary Search

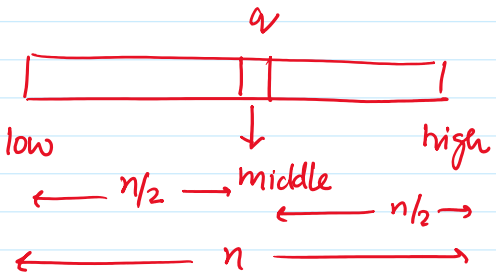
Input: A sorted array of size n, query element q

Input: A sorted array of size n, query element q

Output: If $q \in A$?

$T(n)$:= worst case running time on input of size n

$$T(n) = T(n/2) + c$$
$$= T(n/2^2) + c + c$$
$$= T(n/2^3) + c + c + c$$
$$= T(n/2^4) + c + c + c + c$$
$$\vdots$$
$$= T(n/2^k) + kc$$



if $n = 2^k \Rightarrow k = \log_2 n$

$n/2^k = 1$

$$= T(1) + kc$$
$$= c_1 + c \cdot \log_2 n \quad \because T(1) = c_1$$
$$= O(\log_2 n) = \Theta(\log_2 n)$$
$$= \Omega(\log_2 n)$$