8/1/18

→ Time Complexity

→ Measured as a function of input size, worst case

→ Correctness [loop invariants]

→ Induction

HW Deadline Thu (11 th)

Hint : Induction

• Asymptotic Notation

T(n)

$$f(n) = \overset{\text{big-O}}{O}(g(n))$$

$$f(n) \le c \cdot g(n) \ \forall \text{ suff. large } n$$

$$f(n) = \overset{\text{Small-o}}{o}(g(n))$$

$$\text{If } \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0 \qquad \text{if } \lim_{n \to \infty} \frac{g(n)}{f(n)} = 0$$

$$f(n) = O(g(n)) \qquad g(n) = O(h(n))$$

$$\Rightarrow f(n) = O(h(n))$$

$\log_2 n$, $2n+3$, $\log_2 \log_2 n$, $\frac{n}{\log n}$, $\sqrt{n}$, $(\log_2 n)^{\log_2 n}$

$f_1(n)$, $f_2(n)$, $f_3(n)$, $f_4(n)$, $f_5(n)$, $f_6(n)$

$\to 2^{\log_2 n}$

3^n, n^4 $\qquad (\log_2 n)^{\log_2 n}$

$f_7(n)$, $f_8(n)$

$$f_1 < f_6$$

$$f_3 < \cancel{f_1} < f_5 < f_4 < f_2 < f_8 < f_7$$

$$\log_2 \log_2 n = o(\log_2 n) \qquad \cancel{\log_2 n = o((\log_2 n)^{\log_2 n})}$$

$$f_3 < f_1 < f_6 <$$

$$(\log_2 n)^{\log_2 n} = 2^{\log_2 n (\log_2 \log_2 n)}$$

$$f_3(n) < f_1(n) < f_5(n) < f_4(n) < f_2(n) \quad f_6(n) < f_7(n)$$

$$< f_8(n) <$$

$\Rightarrow f_3 < f_1 < f_5 < f_4 < f_2 < f_8 < f_6 < f_7$

⇝ Divide and Conquer Strategy

1. Divide I/P into smaller pieces.
2. Solve the problem on smaller pieces. [Use Recursion]
3. Combine / use soluti from step 2 to solve problem on original input.

⇒ Binary search, merge sort.

Merge Sort :- I/P : $A[1], \ldots, A[n]$

$A[1] \ldots A[\lfloor \frac{n}{2} \rfloor] \quad A[\lfloor \frac{n}{2}+1 \rfloor], \ldots A[n]$

$\downarrow$ Recursively sort each half.

$A[1] \le A[2] \le \ldots A[n]$

$\downarrow$ $A_1[n]$ ~~$+ B[m]$~~

$B[1] \le B[2] \ldots \le B[m]$

Combining the two solution

Let $T(n)$ denote the time complexity of merge sort on i/p of size n.

$$T(n) = \underbrace{T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right)}_{\text{Time for recursive Calls}} + \underbrace{c \cdot n}_{\text{Time for merging}}$$

Method 1

Expand

✓ Let just assume

$$T(n) = 2T(n/2) + c \cdot n$$

$$= c \cdot n + 2 \cdot \left[c \cdot \frac{n}{2} + 2T(n/4)\right]$$

$2cn + 4T(n/4)$

$$= c \cdot n + 2 \cdot \left[c \cdot \frac{n}{2} + 2\left[c \cdot \frac{n}{4} + 2T(n/8)\right]\right]$$

$3cn + 8T(n/8)$

K times unrivaling

$$T(n) = k \cdot c \cdot n + 2^k \cdot T\left(\frac{n}{2^k}\right)$$

when $\frac{n}{2^k} \le 1$, $T\left(\frac{n}{2^k}\right) = 0$

$k \simeq \log_2 n$

$T(n) = c \cdot n \cdot \log_2 n + 2^k \cdot 0$

$= c \cdot n \cdot \log_2 n$

$T(n) = \theta(n \log_2 n)$

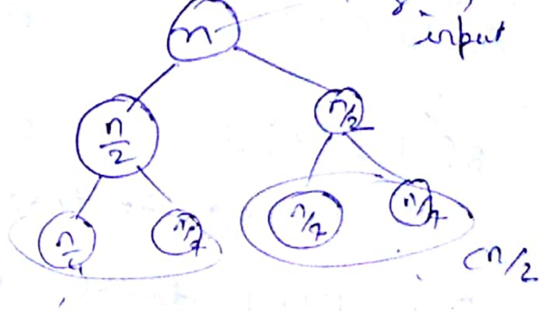
Theta

Method 2 :- Recursion Tree method

Merging + $c \cdot n$

$c \cdot n$. height of tree

$\underline{(c \cdot n \cdot \log_2 n)}$



Level 1 $\quad cn$

level 2 $\quad \frac{cn}{2} + \frac{cn}{2} = cn$

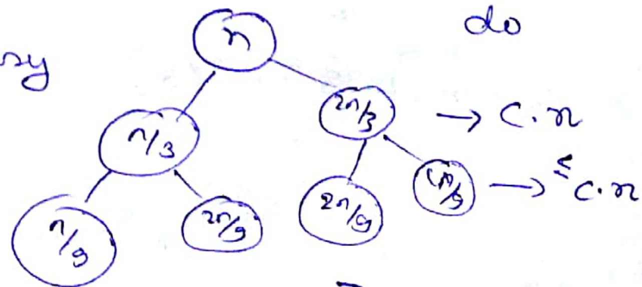
level 3 $\quad \frac{cn}{4} + \frac{cn}{4} + \frac{cn}{4} + \frac{cn}{4} = \frac{cn}{4} \times 4 = cn$

# of level $= \log_2 n$

$\underbrace{a_1, \ldots a_{n/3}}_{n/3} / \underbrace{a_{\frac{n}{3}+1}, \ldots a_n}_{2n/3}$

$T(n) = T(n/3) + T(2n/3) + c \cdot n$ → expansion is quiet to do

by using recursion tree it's easy



height of tree

no. of level $\Rightarrow \log_{(3/2)} n$

$\Rightarrow \left[c \cdot n \cdot \log_{(3/2)} n\right]$

Q → Finding min & Second least elt with $\frac{3n}{2}$ Comparison

D & C strategy.

$$T(n) = 3\,T(n/4) + \sqrt{n}$$

find $T(n)$

17-1-18

→ Ex :-

1. $T(n) = 2T\left(\frac{n}{3}\right) + n$, $T(1)$ is constant.

$$= n + 2\left[\frac{n}{3} + 2T\left(\frac{n}{9}\right)\right]$$

$$= n + 2\left[\frac{n}{3} + 2\cdot\left[2T\left(\frac{n}{27}\right) + \frac{n}{9}\right]\right]$$

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

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

$n = 3^k$

$$T(n) = (3^k - 2^k)\,3 + 2^k T(1)$$

$$= \Theta(n)$$

⇒ 2. I/p : $a_1, a_2, a_3, a_4, \ldots\ldots\ldots a_{n-1}, a_n$

O/p : Smallest & 2nd smallest elements.

Step 1 :- Pair up the no's & compare elements of each pair.

Step 2 :- Now consider the set of min. elements of each pair.

Step 3 :- Find min & second least element of these $\frac{n}{2}$ elements recursively.

Step 4 :- If $\begin{Bmatrix} a_i \\ a_{i+1} \end{Bmatrix}$ = min, + $\{a_{i+1}\}$

$$T(n) = a\,T\left(\frac{n}{b}\right) + f(n)$$

$a \geq 1$, $b \geq 2$ integers,

$a = 4$, $b = 10$

a # recursive calls, on ip/s of size $\frac{n}{b}$

n → $\frac{n}{10}$ $\frac{n}{10}$ $\frac{n}{10}$ $\frac{n}{10}$

⇒ f(n)

Ex :-

$$T(n) = \left\lfloor \frac{n}{2} \right\rfloor + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + 4$$

$$T(n) \simeq \frac{n}{2} + T\left(\frac{n}{2}\right) + 4 = \left(4 + \frac{n}{2}\right) + \left(4 + \frac{n}{4}\right) \cdots \left(4 + \frac{n}{2^k}\right) + T\left(\frac{n}{2^k}\right)$$

$$= 4k + (n) + T(1) \qquad \frac{n}{2^k} \le 2$$

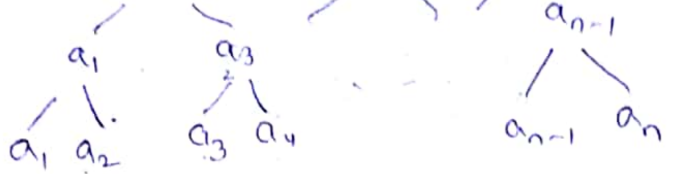
$$k \sim \log_2 n$$

$$= 4\log_2 n + n + T(1)$$

$$\frac{n}{2} + \frac{n}{4} + \frac{n}{8} \cdots \le n$$

$$T(n) = n + 4\log_2 n + T(1)$$

Another way



Check the path of smallest elements and the second smallest will be on same path.

$$(n-1) + \lceil \log_2 n \rceil$$

$$T(n) = aT\left(\frac{n}{b}\right) = a\left[aT\left(\frac{n}{b^2}\right)\right]$$

$$n = b^k$$

$$\cdots\; a^k\, T\left(\frac{n}{b^k}\right) \qquad k = \log_b n$$

$$T(n) = f(n)$$

$$a^{\log_b n}\,[T(1)] = \text{const.}$$

$$\boxed{T(n) = O\left(n^{\log_b a}\right)}$$

$a = 2,\ b = 2$

$T(n) = O(n)$

$T(n) = 2T(n/2)$

Thm : (Master) :-

If $f(n) >> n^{\log_b a}$; $T(n) = f(n)$

If $n^{\log_b a} >> f(n)$, $T(n) = n^{\log_b a}$

If $f(n) = \Theta(n^{\log_b a})$, $T(n) = n^{\log_b a} \cdot \log n$

$\&\ af\left(\frac{n}{b}\right) \le c \cdot f(n)$ $\qquad = f(n) \log n$

Ex :- $T(\ldots \frac{\ldots}{3})$

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

$\Rightarrow n^{\log_2 2}, n \Rightarrow n \log n$

Ex :-

I/P A seq. of no's

Buy once, Sell once.

30, 80, 100, 20, 10, 50, 40

Day 1 2 3 4 5 6 7

Stock

$\left\{ \underset{j>i}{\text{Maximum}} \; A[j] - A[i] \right\}$

$\underset{i,j}{\text{Max}} \; A[i] + A[i+1] \ldots A[j]$

Soln 1 :- Find max difference among all pairs $\Theta(n^2)$ steps.

Soln 2 :- D & C

$a_1, a_2, \ldots \ldots a_{\frac{n}{2}} \;|\; a_{\frac{n}{2}+1}, \ldots \ldots a_n$

Buy & sell this half | Buy & Sell in this half

Buy ———— Sell

$\text{Max}\left(a_{\frac{n}{2}+1}, \ldots a_n\right) - \text{Min}\left(a_1, a_2, \ldots a_{\frac{n}{2}}\right)$

$$\text{OPT}(a_1, \ldots a_n) = \text{Max}\left\{ \text{OPT}(a_1, \ldots a_{\frac{n}{2}}),\; \text{OPT}(a_{\frac{n}{2}+1}, \ldots a_n),\; \left(\underbrace{\text{Max}\{a_{\frac{n}{2}+1}, \ldots a_n\} - \text{Min}\{a_1, \ldots a_{\frac{n}{2}}\}}_{\text{linear time}}\right)\right\}$$

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

$\boxed{T(n) = O(n \log_2 n)}$

Ex :- Find a linear time algorithm (using Induction) ?

Ex. 1 :- $T(n) = 3T(n/2) + n$: $n^{\log_2 3}, n$

Ex. 2 :- $T(n) = 3T(n/4) + \sqrt{n}$ $n^{\log_4 3}, \sqrt{n}$

18-1-18

Ex 1 : $T(n) = 3T(\frac{n}{2}) + n$

Compare $n^{\log_2 3}$ $(3^{\log_2 n})$ vs n

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

Ex 2 : $T(n) = 3T(\frac{n}{4}) + \sqrt{n}$

$n^{\log_4 3}$ vs $\sqrt{n}$

$\log_4 3 > \log_4 2 = \frac{1}{2}$

$T(n) = \Theta(n^{\log_4 3})$

Integer Multiplication

I/P : $A = (A_1, A_2, A_3, \dots A_n)_2$ $[A_i \in \{0, 1\}]$

$B = (B_1, B_2, B_3, \dots B_n)_2$

O/P :- Find $A \cdot B$.

$A_1 A_2 A_3 \dots A_n$
$B_1 B_2 B_3, \dots B_n$

$A \cdot B_n$
$2A \cdot B_{n-1}$
$4A \cdot B_{n-2}$

Takes $\sim 2n^2$ operation

$A \cdot B$
$= B_n \cdot A + 2B_{n-1} A$
$+ 4B_{n-2} \cdot A \dots$
$\dots + 2^{n-1} B_1 \cdot A$

Add n # integers each $\leq 2n$ bits long

Best known algo. complexity $O(n \log n \{ \quad \})$

$A = A_1 A_2 \dots A_{\frac{n}{2}} A_{\frac{n}{2}+1} \dots A_n$
$B = B_1 B_2 \dots B_{\frac{n}{2}} B_{\frac{n}{2}+1} \dots B_n$

$A = A_n + 2A_{n-1} + \dots 2^{n-1} A_1$

Karatsuba's method

$A_L = A_1 A_2 \dots A_{\frac{n}{2}}$, $B_L = B_1 B_2 \dots B_{\frac{n}{2}}$

$A_R = A_{\frac{n}{2}+1} \dots A_n$, $B_R = B_{\frac{n}{2}+1}, \dots B_n$

$A = 2^{\frac{n}{2}} A_L + A_R$ $\quad$ $B = 2^{n/2} B_L + B_R$

$A \cdot B = 2^n A_L B_L + (A_L B_R + A_R B_L) 2^{n/2} + A_R B_R$

Find A_L, B_L, $A_L \cdot B_R$, $A_R \cdot B_L$, $A_R \cdot B_R$ recursively + Add 4 integers

$\downarrow$
$\frac{n}{2}$ bits

4 recursive calls

$T(n) = 4T(\frac{n}{2}) + n$

$\Rightarrow O(n^2)$

$n^{\log_2 4}$ vs n

~~Identity~~ Idea :- Make 3 recursive calls instead of 4.

$$(A_L + A_R)(B_L + B_R)$$

$$= A_L B_L + (A_L B_R + A_R B_L) + A_R B_R$$

Find $(A_L + A_R)(B_L + B_R)$ recursively

& $A_L B_L$ and $A_R B_R$ recursively

$\cancel{C = A_L B_R + A_R B_L}$

$$C = (A_L + A_R)(B_L + B_R) - A_L B_L - A_R B_R$$

$$\Rightarrow T(n) = 2T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}+1\right) + \Theta(n)$$

$$T(n) = 3T\left(\frac{n}{2}\right) + \Theta(n) \longrightarrow T(n) = \Theta(n^{\log_2 3})$$

Can we do better ?

$A = A_1 A_2 A_3$ $\quad B = B_1 B_2 B_3$

$\downarrow \downarrow \downarrow$

$\frac{n}{3}$ bits

~~Hint~~ :- H. W :-

n people $\rightarrow$ round-robin

$P_1 \rightarrow P_2 \rightarrow P_3 \cdots \rightarrow P_{n-1} \rightarrow P_n$ $\quad$ P_i won against P_{i+1}

$P_1 \rightarrow P_2 \rightarrow \cdots \rightarrow P_{n-1}$

Consider P_n

If $P_{n-1} \rightarrow P_n$, add P_n at end

Else Find i, the first person such that

insert Q

$P_{i-1} \rightarrow Q \rightarrow P_i$.

$a_1 \le a_2 \le \cdots \le a_{n-1} \mid a_n$

Find i such that $a_{i-1} \le a_n \le a_i$

$\downarrow$ Binary search $O(\log n)$

$P_1 \to P_2 \cdots \to P_n \mid P_{n+1} = Q$

$Q \to P_{\frac{n}{2}}$ — fit Q in first half

$P_{\frac{n}{2}} \to Q$ — ↳ fit Q into 2nd half

$\exists$ smallest $i \le \frac{n}{2}$: $Q \to P_i$

⇒ I/P :- 12, 18, 30, −2, 0, 4, 8

Original sequence :- −2, 0, 4, 8, 12, 18, 30

Sorting

Rotate by t times

A rotated sorted sequences.

O/P :- Find K, the # of rotations.

i) Find position of smallest ~~position~~ element.

⇒ compare the 1st element and middle element

• 22-1-18

HW Deadline :- 25 Jan (Thursday).

• SELECTION PROBLEM

I/P :- An array $A[1 \ldots n]$ of integers, and $1 \le K \le n$

O/P :- The k^{th} smallest element of A. (A_K)

18, 32, 0, −5, 11, 6, 20, 10

A[1 2 3 4 5 6 7 8]

$A_1 = -5$ $A_2 = 0 \ldots$

$\ldots A_5 = 11$ $A_6 = 18$

$A_8 = 32$

Algo 1 :-

Sort the array, and pick the k^{th} element in the sorted array.

↳ $O(n \log n)$

Algo 2 :-

Build a heap & find the k^{th} smallest element.

↳ $O(n + k \log n)$

linear time for building heap

Algo 3 :-

Repeatedely find the i^{th} smallest ellement from $i=1$ to k.

$\hookrightarrow O(k \cdot n)$

Find median :-

SELECT $(A, \lceil \frac{n}{2} \rceil)$ $\qquad A[\lceil \frac{n}{2} \rceil]$ = median

$\hookrightarrow \Theta(kn)$

Goal :- A $O(n)$ algorithm to find k^{th} smallest element.

• Algo A that find the median elt. of any list

$\Downarrow$

Algo to find k^{th} smallest element for any k.

$n = 1000$, $A[1, 2, \ldots 1000]$

$k = \underline{736^{th}}$.

736

k^{th} smallest element

500 ele $\quad$ 500 th ele

500 501

250 $\quad$ 250

125 $\quad$ 125

① Find A_{501} (median of original list)

② Find f_1 : $\{x : x \le A_{501}\}$ → $|f_1| = 500$

f_2 : $\{x : x \ge A_{501}\}$ $|f_2| = 499$

$\downarrow$

$O(n)$

idea :- add 472 elements and then 736^{th} element is median

1000+472

63

The 736 element of A is $\underline{235}^{th}$ element of f_2 $\qquad 736-(500+1)$

Recurse the process

find the median entry is $f_2 \to 499$

Median → 249^{th} elements ↔ 249 elements

250

249 $\qquad$ 249

124 ↙ ↘ 124 $\qquad 235 - 125 = 110$

62

61 $\quad$ 6 $\quad$ 62 → $110 - (62+1) = 47$.

Algo to find an ~~algor~~ approx median → [Not at the extremes]

median → $\frac{n}{2}^{th}$ elt.

↳ (exact middle)

:- linear time algo to find kth smallest elt.

I/P :- $A[1, 2, 3, \dots n]$ $\rightarrow$ SELECT (A, k) $\rightarrow T(n)$

1. $A[1]$ $A[2]$ $A[3]$ | $A[4]$ $A[5]$ $\rightarrow$ Sort this group

$A[6]$ $A[7]$ $A[8]$ $A[9]$ $A[10]$ "

"

"

"

$A[n-4]$ $A[n-3]$ $A[n-2]$ $A[n-1]$ $A[n]$

2. $A[2] \le A[4] \le A[3] \le A[1] \le A[2]$

Time $= O(\frac{n}{5} \log 5) = O(n)$

look the middle element of each group.

$\frac{n}{5}$ elements

$\rightarrow$ Call select $(\underbrace{A[3], A[8], \dots A[n-2]}_{n/5}, \frac{n}{10})$ $\rightarrow T(n/5)$

(i.e find median elt. of medians)

M

Claim :- M is an approximate median of $A[1, 2, \dots n]$

How many ele. $\le M$?

$\frac{3n}{10} = \frac{n}{10} + \frac{2n}{10}$ $\leftarrow$ elem

atleast element must be less than M

$\frac{n}{10}$ ele

M

$\frac{n}{10}$ elements

atleast $M \ge (\frac{3n}{10})$ element is A

atleast $M \le (\frac{3n}{10})$ element is A

3. $S = \{x : x \le M\}$ $|S| \ge 3n/10$

$T = \{x : x \ge M\}$ $|T| \le 3n/10$

If $k \le |S|$ $\longrightarrow$ select (S, k)

$\therefore$ k^{th} element in S

$\longrightarrow \theta(n)$

else:

$\therefore$ find select $(T, k - |S|)$

recursive call

$\max(|S|, |T|) = 7n/10$

$T(|S|)$ or $T(|T|)$

$|S| \le 7n/10$

$|T| \le 7n/10$

clearly atmost $T(7n/10)$

$$T(n) = \underbrace{\theta(n)}_{\text{sorting group of 5}} + \underbrace{T\left(\frac{n}{5}\right)}_{\substack{\text{Recursive call} \\ \text{[select (median, median)]}}} + \underbrace{\theta(n)}_{\substack{\text{partition around } M \\ [<M \mid M \mid >M]}} + \underbrace{T(7n/10)}_{\substack{\text{Recursive call for} \\ k^{th} \text{ smallest}}}$$

$$T(n) = \theta(n) + T(n/5) + T(7n/10)$$

$a = 1/2 \quad b = 1/3 \qquad a+b = 5/6$

Ex:- $T(n) = T(n/2) + T(n/3) + \theta(n)$

$T(n) = T(2n/3) + T(n/12) + \theta(n) \qquad a+b = \frac{3}{4}$

$T(n) = T(n/5) + T(7n/10) + \theta(n) \qquad a+b = \frac{9}{10}$

recursion tree

$L_1 = [1 \cdot \ldots\ n]$

$L_2 = [(a+b)n]$ $\Rightarrow$ $\dfrac{n(1-(a+b))}{1-(a+b)}$ $\qquad \dfrac{n(1-\frac{9}{10})}{1-9/10}$

$L_3 = [(a+b)^2 n]$

$n(1 + (a+b)n + (a+b)^2 n \ldots$ $\qquad \underline{10n}$

linear

Ex:- check the groups

$\rightarrow$ size $\rightarrow 3$

$\rightarrow$ size $\rightarrow 7$

25-1-18

Linear Time Sorting

- Merge Sort, Heap Sort : $O(n \log n)$
- Any sorting algo. must make $\geq \Omega(n \log n)$ comparison in the worst case.

I/P : $A[1, \ldots n]$
$A[i] \in \{0, 1\}$ Goal : Sort A

Count # zeroes $\leq n$ steps

$A[i] \in \{0, 1, 2, \ldots 100\}$

$\Rightarrow$ have a Counter Array of size 101 $\rightarrow O(n)$

I/P :- $A[1, \ldots n]$ $n \sim 10^6$

$A[i] \in \{0, 1, 2, \ldots k-1\}$ $k \sim 10^6$

Idea = (1) Initialize a Counter array of size k $\Leftarrow$ Counting Sort

(2) for $i = 1$ to n
  Increment $C[A[i]]$

(3) $C[0]$ zeroes.
$C[1]$ 1's.
⋮
$C[k-1]$. $(k-1)$'s

when $k = n$
$A[1 \ldots n]$; $A[i] \sim \{0, 1, \ldots n-1\}$

Running time : $\Theta(n+k)$
linear time

but if $k \geq n$
$n = 10^6$ $k = 10^{12}$
then $O(n^2)$
Time Complexity
$\therefore$ Better to use merge sort.

if $A[i] \in \{0, 1, \ldots (n^2-1)\}$

Counting Sort :- $A[1, 2, \ldots n]$

If $A[i] \in \{0, 1, \ldots k-1\}$

→ Counting Sort ⇒ $O(n+k)$

→ linear if $k = O(n)$

Goal :- $k = n^2$

→ Still we need to have linear time algo.

Idea :- Sort by digit :- (not by leading digit, but infact by least significant digit)

3 digit no's

135	312	312	135
278	843	135	154
154	154	843	278
843	135	154	312
312	278	278	843

1) • Sort by last digit

• Sort by 2nd last digit (Ignore all other digit)

Stable Sort

Radix Sort

⇒ Each number has d digits. $A_1, A_2, \ldots A_d$

• For $i = d$ to 1 ;

(stable) sort by i^{th} digits → Count Sort

• Now the no's are sorted

↑ 1st digit ↑ d^{th} digit

Ex :- Sorting words :-

DOG	SEA	TAB	BAR
SEA	TEA	EAR	DOG
MOB	MOB	TAR	EAR
EAR	TAB	BAR	FOX
TAB	DOG	SEA	MOB
TEA	PIG	TEA	PIG
TAR	EAR	PIG	SEA
PIG	TAR	MOB	TAB
BAR	BAR	DOG	TAR
FOX	FOX	FOX	TEA

Time Complexity :-

$A[1, \ldots n]$

$A[i].key \in \{0, 1, \ldots k-1\}$

$A[i].data$

Count Sort → $C[0] = \#\ 0's$

$C[1] = \#\ 1's$

⋮

$C[k-1] = \#\ (k-1)^{th}$ no's

Store indices ←

Linked list ←

To sort each coloumn

time = $\Theta(n+b)$

$\downarrow$

be base

$= \Theta\{\underbrace{d}_{\text{\# of digits}}(n+\underbrace{b}_{\text{base}})\}$

b count's

$\{0, 1, \ldots b-1\}$

$\downarrow$

Each digit

English alphabet $b = 26$

Binary ; $\Rightarrow b = 2$

decimal ; $b = 10$

for $n = 10^6$; $A[i] \in \{1, \ldots\ldots, 10^{12}\}$

$\downarrow$

$0 - 999999999999$

Base b representation of x

~~no. of~~ representation

$\downarrow$

where $b = 10^6$

$x = q \cdot 10^6 + r$

$q \leq 10^6$, $r < 10^6$

Write in base $b = 10^6$

$a[0] = (q_0, r_0)$

$a[1] = (q_1, r_1)$

$a[2] =$

$\vdots$

$a(n) = (q_n, r_n)$

by quoitent

Sorting by remainder

two digit no.

$A \in \{0, 1, \ldots n^2-1\}$

$A[i] = nq + r$

$q, r \leq n$

In base n ;

$(0, \ldots n-1) \mid (0, \ldots n-1)$

q $\quad$ r

$O(n)$ $\quad$ $O(n)$

$\Rightarrow$ time $= \Theta(2n)$

Sort first by r,

then by q.