CS2443 - ALGO

* MULTIPLICATION Algos

Mid 1 - 20
Mid 2 - 25
Final - 45
Cls - 10

→ Multiply two n-digited nos (a, b)
Algo:- add b a times

```
x = 0
for i = 1 → b
    x += a
return x
```

claim:- ∀ any $i \geq 0$
at the end of i^{th} iteration, x contains value i.a

→ For many algo. Proof will be by induction

Base case $i = 0 = 0 \cdot a$

sub condition: Assume $j < i$

$x = x + a$

$= (i-1)\,a + a = i.a.$

Strong induction / weak induction → anything can be used

Proof by induction:- true for all $j < i$

runtime:- b operations (of add^n of 2 N-digit nos)

⇒ bn operations of add^n of 2 1-digit nos)

$10^{n-1} \leq b \leq 10^n - 1$

$\Rightarrow \leq T(n) \leq (10^{n-1} \leq b \leq 10^n - 1)\, n.$

for $n = 31 \Rightarrow T(n) \approx 10^{20}$ hrs.
(for a system that can do 1 million ops/sec)

So, good algo required.

→ Basic multiplication (also long multiplication)

$1122 \times 5678 = (2 \times 5678 + 20 \times 5678 + 100 \times 5678 + 1000 \times 5678)$

```
 1122
×5678
```

→ Running time = $5n^2$ ($2n^2 + 3n^2$)
$6n + 4n$

Masters theorem

→ Karatsuba Multiplication:-

- Break it to $\frac{n}{2}$ no.s

x	y
56	78
(11)	(22)
u	v

if length of both i/p ⇒ axb.

78 × 221 = 156 + 1716

step 1:- 56 × 11 = 616 (x × u)

step 2:- 78 × 22 = 1716 (multiply recursively) (y × v)

step 3 = (x+y)(u+v)

(134)×(33) = 4422

134 × 33 = 402 + 4022 = 4422

step 4 = ③ - ② - ①

= 4422 - 1716 - 616

= 2090

1716 − 616 = 1100 + 32 = 1132

4422 − 2332 = 2090

step 5:- 616 0000

1716

209 000

Correctness:-

$a = x \times 10^{n/2} + y$

$b = u \times 10^{n/2} + v$

$axb = 10^{n}(xu) + 10^{n/2}(xv + uy) + yv$

Time complexity:-

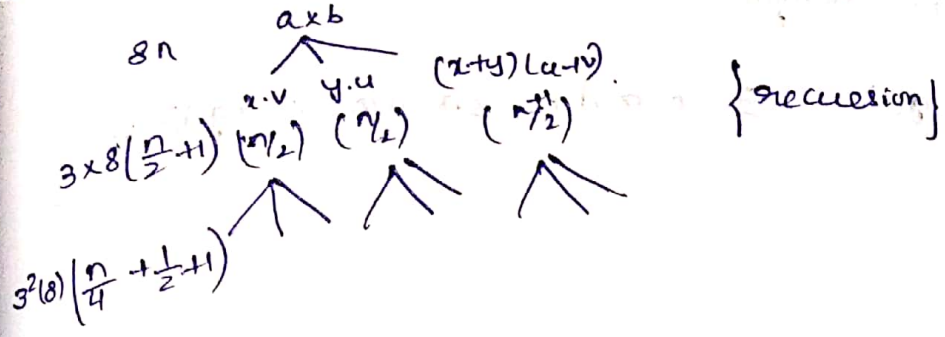
$T(n) = 3T\left(\frac{n}{2}+1\right) + 8n \leq cn^{1.5849}$

→ $T(n) = 3\left[3T\left(\frac{\frac{n}{2}+1}{2}+1\right)\right] + 8n$

$8\left(\frac{n}{2}+1\right)$

$T(n) = 3^2 T\left(\frac{n}{2^2} + \frac{1}{2} + 1\right) + 3 \times 8\left(\frac{n}{2}+1\right) + 8n$

=



$3^l(8)\left(1+\cdots+\frac{1}{4}+\frac{1}{2}+1\right)$

$l = \log_2^n$

$8n + 3\times 8\left(\frac{n}{2}+1\right) + 3^2 8\left(\frac{n}{4}+1+\frac{1}{2}\right)\cdots$

$8n\left(1+3\times\frac{1}{2}+3^2\times\frac{1}{2^2}+\frac{3^3}{2^3}+\cdots+\left(\frac{3}{2}\right)^{\log_2^n}\right.$

$+ 8\left[3+\left(3^2+\frac{3^2}{2}\right)+\left(3^3+\frac{3^3}{2^2}+\frac{3^3}{4}\right)\cdots\right]$

$\leq 8n\left(\frac{3}{2}\right)^{\log_2^n}\left(1+\frac{2}{3}+\left(\frac{2}{3}\right)^2+\cdots\right) + A.$

$\frac{1}{1-\frac{2}{3}}.$

$8n\left(\frac{3}{2}\right)^{\log_2^n} = 8n\left(\frac{3}{2}\right)^{\frac{\log_{10}^n}{\log_{10}^2}}$ $\quad 8n\ \frac{3^{\log_2^n}}{\left(2^{\log_2^{10}}\right)^{\log_{10}^n}}$.

$\leq 24\, n^{1.58}$

adding $A \Rightarrow 26\, n^{1.58}$

Asymptotic analysis:-

$\left(O(f(n))\right) = \{ g(n) \quad \exists\ c \text{ and } n_0 \text{ s.t } f(n) \le c\,g(n),\ \forall\ n \ge n_0$

$\left(\begin{matrix} \ge \\ n^{1.5} = o(n^2) \end{matrix}\right)$ $\Omega(f(n)) = \{ g(n) \quad \exists\ c \text{ and } n_0 \text{ s.t } g(n) \ge c f(n)$ $\forall\ n \ge n_0$

$\left(\begin{matrix} = \\ \Theta(f(n)) \end{matrix}\right) = \{ g(n) \quad \exists$

$O(f(n))$

→ $T(n) = a\,T(n/b) + d(n)$ → fundamental recurrance because it can be converted to

$S(n) = S(n-1) + d(n)$ → driving function

$S(n) = S(0) + \sum_{i=1}^{n} d(i)$

$R(n) = a\,R(n-1) + d(n)$

$\frac{R(n)}{a^n} = \frac{R(n-1)}{a^{n-1}} + \frac{d(n)}{a^n}$

Let $S(n) = \frac{R(n)}{a^n}$

$S(0) = R(0)$

$S(n) = S(n-1) + \frac{d(n)}{a^n}$

$= S(0) + \sum_{i=1}^{n} \frac{d(i)}{a^i}$

$$\boxed{R(n) = a^n R(0) + \sum_{i=1}^{n} d(i)\, a^{n-i}}$$

$T(n) = T(n/b) + d(n)$ let $n = b^k$

$T(b^k) = T(b^{k-1}) + d(b^k)$

$$S(k) = S(k-1) + d(b^k)$$

$$= S(0) + \sum_{i=1}^{k=\log_b^n} d(b^i)$$

$$\boxed{T(n) = T(1) + \sum_{i=1}^{n} d(b^i)}$$

→ $T(n) = aT\left(\frac{n}{b}\right) + d(n)$.

$T(b^k) = aT(b^{k-1}) + d(b^k)$

$R(k) = aR(k-1) + d(b^k)$

$S(k) = S(k-1) + \frac{d(b^k)}{a^k}$

$= S(0) + \sum_{i=1}^{k} \frac{d(b^i)}{a^i}$.

Let $n = b^k$

$R(k) = T(b^k)$

$S(k) = \frac{R(k)}{a^k}$.

$$R(k) = a^k R(0) + \sum_{i=1}^{k} d(b^i)\, a^{k-i}$$

$$\boxed{T(n) = a^k T(1) + \sum_{i=1}^{k} d(b^i)\, a^{k-i}} \rightarrow (i)$$

$a^k = n^{\log_b^a}$

$a^k = a^{\log_b^n}$

$= \left(a^{\log_a^n}\right)^{\log_b^a}$

$= n^{\log_b^a}$.

**

$$\boxed{T(n) = n^{\log_b^a} T(1) + \sum_{i=1}^{k} d(b^i)\, a^{k-i}}$$

$T^*(n) = 3T^*\left(\frac{n}{2}\right) + 8n$

$T^*(n) = n^{\log_2^3} + \sum_{i=1}^{k} 8 \times 2^i \times 3^{k-i}$

$= n^{1.5849} + 3^k \cdot 8 \sum_{i=1}^{k} \left(\frac{2}{3}\right)^i$

$= 24\left(n^{1.5849}\right)$.

$T(n) = aT\left(\frac{n}{b}\right) + d(n)$

$\boxed{T(n) = n^{\log_b^a} T(1) + \sum_{i=1}^{k} d(b^i)\, a^{k-i}}$

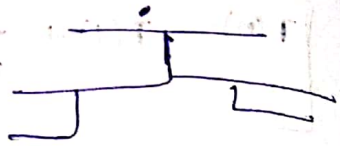
$n = b^k$

$\log_b^n = k$

→ $T(n) = 3T\left(\frac{n}{2}+1\right) + 8n$

(Induction)

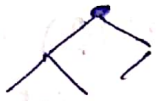
$T(n) = T(n/2) + 1$



→ $Q(n) = T(n+\alpha)$

$= 3T\left(\frac{n}{2} + \frac{\alpha}{2} + 1\right) + (8n + 8\alpha)$

$= 3Q\left(\frac{n}{2} - \frac{\alpha}{2} + 1\right) + 8n + 8\alpha.$



Putting $\alpha = 2$

$Q(n) = 3Q\left(\frac{n}{2}\right) + 8n + 16$

use the previous result i.e., (1)

we get $Q(n)$.

but $Q(n) = T(n+2)$.

$\Rightarrow T(n) = Q(n-2)$.

$Q(n) = c \cdot n^{1.5849}$

$T(n) = Q(n-2) = c \cdot (n-2)^{1.5849}$

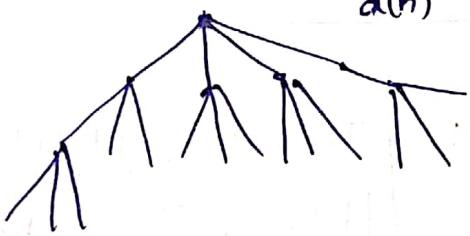
→ $\boxed{T(n) = n^{\log_b^a} T(1) + \sum_{i=1}^{k} d(b^i)\, a^{k-i}}$

→ To prove masters theorem, ? (Recurrance tree)

$= n^{\log_b^a} T(1) + d(n) + a\, d\left(\frac{n}{b}\right) + a^2 d\left(\frac{n}{b^2}\right) + \ldots$ → (2)

$n^{\log_b^a}$ → no. of leaves in the tree

$d(n)$



$a\, d\left(\frac{n}{b}\right)$

$a^2 d\left(\frac{n}{b^2}\right)$

$a^k d\left(\frac{n}{b^k}\right)$

Masters theorem:

(1) If $d(n) = \theta\left(n^{\log_b a - \varepsilon}\right)$

for $\varepsilon > 0$

then $T(n) = \theta\left(n^{\log_a b}\right)$

$d(n) = n^c$ where c is less than $\log_b a \Rightarrow c < \log_b a$

$\Rightarrow b^c < a$

expanding (2). we get,

$= n^{\log_b a} T(1) + n^c + a\left(\frac{n}{b}\right)^c + a^2\left(\frac{n}{b^2}\right)^c$

$+ n^c \left(\frac{a}{b^c}\right)^k (1 + \dots)$

$n^{\log_a b}$

$n^c \frac{a^k}{(b^c)^k} = n^c \frac{a^{\log_b n}}{\left(b^{\log_b n}\right)^c} = n^c \frac{a^{\log_b n}}{n^c}$

$= a^{\log_b n}$

$= \left(a^{\log_a n}\right)^{\log_b a}$

$= n^{\log_b a}$.

---

(2) $d(n) = \theta\left(n^{\log_b a}\right)$ [is the lower bound]

$T(n) = \theta\left[n^{\log_b a} \log n\right]$

Since there are $\log n$ levels in the tree. and every level is same.

③ $d(n) = \Theta\left(n^{\log_b a+\varepsilon}\right)$ and $a\,d\left(\frac{n}{b}\right) \le z\,d(n)$ $\forall n \ge n_0$, $z \le 1$

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

→ $T(n) = \sqrt{n}\,T(\sqrt{n}) + n$

$T(n) \le a\,n\log n$

$T(1) = 1$ $\quad n \ge f(a)$

$T(n) = \sqrt{n}\,a\sqrt{n}\left(\log n^{1/2}\right) + n$

$= \frac{a}{2} n\log n + n.$

$\le a n\log n$

Ex:- Try to do with recurrance tree method (or) get a tight bound using R.T.

9/1/2020

→ Multiplication algorithms:-

→ a x b → adding a, b times $\Omega(10^n)$

long multiplication $O(n^2)$

Karatsuba mult $O(n^{1.5849})$

→ $T(n) = T\left(\frac{n}{2}\right) + 1$.

$n^{\log_2 1}$ $\quad n^{\log 1}$

$T(n) = \left(T\left(\frac{n}{4}\right) + 1\right) + 1$

$= T\left(\frac{n}{8}\right) + 1 + 1 + 1$

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

$= T(1) + \log n.$

$n = 2^k$

$\log n = k$

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

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

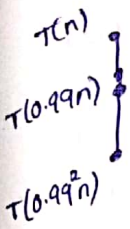
→ $T(n) = T\left(\frac{9n}{10}\right) + 1$

$T(n) = O(\log n)$

→ $T(n) = T(0.99n) + n$.

=

Method 2 –



$n(1 + 0.99 + 0.99^2 + \dots)$

$n\left(\frac{1}{1-0.99}\right)$

$= O(n)$

→ $T(n) = 3T\left(\frac{n}{3}\right) + n$

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

$= 3^2 T\left(\frac{n}{3^2}\right) + 3 \cdot \frac{n}{3} + n.$

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

$= 3^3 T\left(\frac{n}{3^3}\right) + 3^2 \cdot \frac{n}{3^2} + 3 \cdot \frac{n}{3} + n.$

$= 3^3 T\left(\frac{n}{3^3}\right) + 3n.$

$= 3^K T\left(\frac{n}{3^K}\right) + Kn.$

$n = 3^K$

$= nT(1) + (\log n)\, n$

$\log n = K.$

$\Rightarrow T(n) = O(n \log n)$

Method 1 –

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

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

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

$n = 2^k.$

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

$n = 2^k$

$= T(1) + 2^k +$

$= T(1) + n + \frac{n}{2^1} + \dots + \frac{n}{2^{k-1}}$

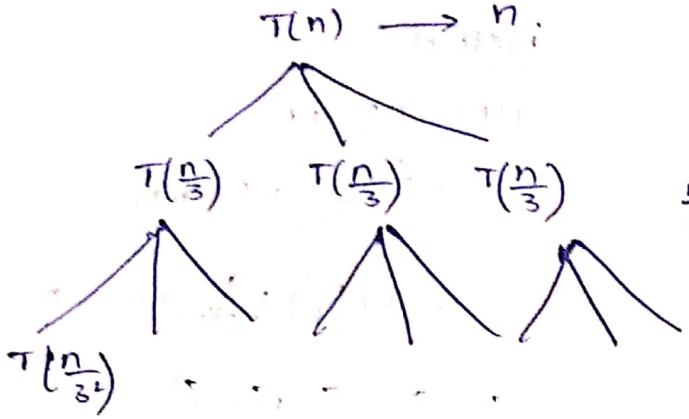
$+ n\left(\frac{1}{2^0} + \frac{1}{2} + \dots + \frac{1}{2^{k-1}}\right)$

$+ n\left(\frac{1 - \left(\frac{1}{2}\right)^K}{\frac{1}{2}}\right)$

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

$= O(n).$

Recurrence tree:



$\frac{n}{3}+\frac{n}{3}+\frac{n}{3}=n$

⇒ total operations = n.

⇒ n operations in each level.

Total operations = n·l (l = depth of tree)

$l = \log n$.

⇒ T.C. = $O(n \log n)$

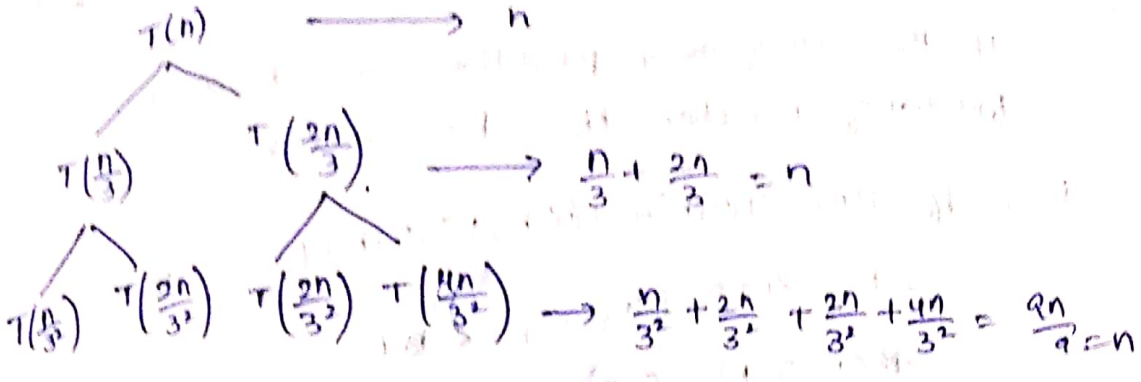
→ $T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + n$.

$T(n) = \left[T(\frac{n}{3^2}) + T(\frac{2n}{3^2}) + \frac{n}{3}\right] + \left[T(\frac{2n}{3^2}) + T(\frac{4n}{3^2}) + \frac{2n}{3}\right] +$

$= T(\frac{n}{3^k}) +$

with recurrance tree method it would be easy to

(P.T.O)

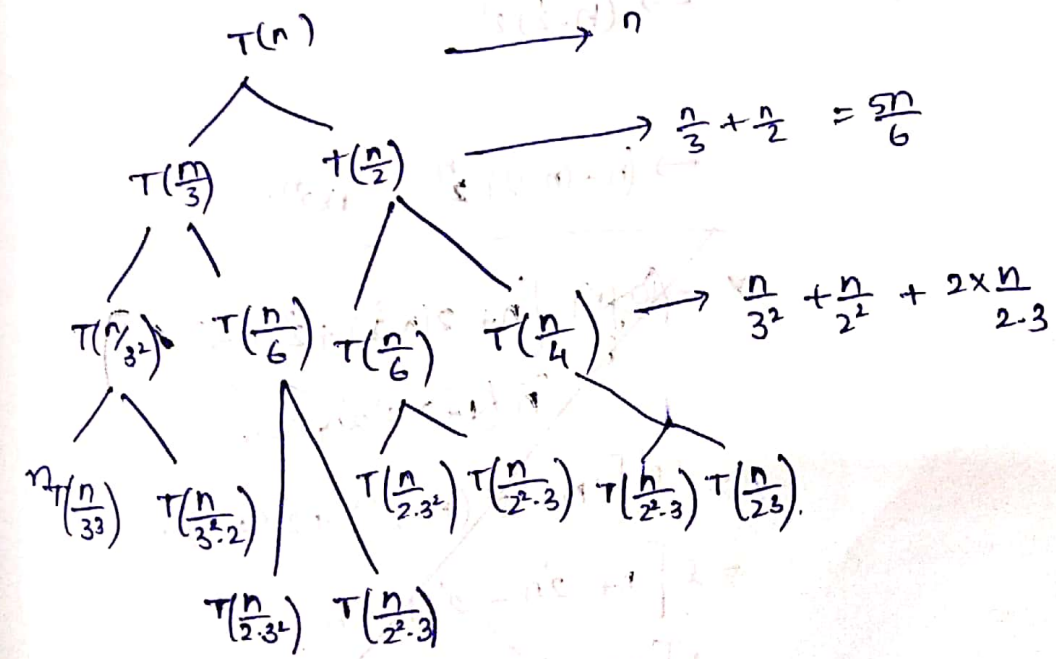


(+)

= n × l (l = length of tree)

T.C ⇒ O(n log n)

→ T(n) = T(n/3) + T(n/2) + n.



(+) O(n)

Observation:-

If the sum of new partitions is $< n$ or a fraction of n, then the T.C is $O(n)$

i.e., If $T(n) = T(\alpha_1 n) + T(\alpha_2 n) + T(\alpha_3 n) + \cdots + n$.

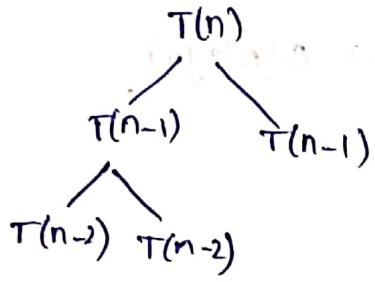
S.T. $\alpha_1 + \alpha_2 + \alpha_3 \cdots + < 1$

then $T(n) = O(n)$.

→ $T(n) = T(n-1) + n$

$= O(n^2)$

→ $T(n) = 2T(n-1) + n$.



→ n

→ $2(n-1)$

→ $(n-2^1)2^2$

→ $(n-(n-1))2^n = 1 \times 2^n$.

$= (\log n)\left[n + 2(n-2^1) + 2^2(n-2) \cdots\right]$

$n(1+2+2^2+\cdots)$

$= l\left[n + 2^1(n-2^0) + 2^2(n-2) + \cdots\right]$

$= l\left[n + 2^1 n - 2^1 + 2^2 n - 2^3 + \cdots\right]$

$= l\left[n(2^0 + 2^1 + 2^2 + \cdots) - (2^1 + 2^3 + \cdots)\right]$

$l = \log n$. $= l\left[n\left(\frac{2^l - 1}{1}\right) - \left(2\left(\frac{2^l - 1}{3}\right)\right)\right]$

$\therefore$ T.C $= O(2^n \cdot n) / O(2^n)$

→ $T(n) = 3T\left(\frac{n}{2}\right) + n$

$$O\left(n^{\log \frac{3}{2}}\right) = O\left(n^{1.5849}\right)$$

Observation :- Here the ~~At~~ $T(n)$ is divided such that the number of levels we go down decreases i.e., for eg: $\frac{n}{2} + \frac{n}{2} + \frac{n}{2}$ in above example is $\frac{3n}{2} > n$.

In these cases, the ~~T(C)~~ T.C goes from exponentiation to polynomial

$T(n) = 4T\left(\frac{n}{2}\right) + n$ $\quad T(n) = aT\left(\frac{n}{b}\right) + n$

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

we observe that as a ↑'s ⟹ (TC) ↑'s.

→ sort n elements :-

By selection sort $O(n^2)$

Implement using min. Heap

also binary search tree.

Merge sort also.

→ ~~Merge sort~~

Me

$\log n$ [n

Mergesort (A[1,... n]) $n \geq 1$

if n=1 then return

else $m = \lfloor \frac{n}{2} \rfloor$

mergesort (A[1,2,.. m])

mergesort (A[m+1, ..., n])

Merge (A[1,?..., n], m)

Merge (A[1, 2, ..., n], m)

i ← 1

j ← m+1

for k = i to n — O(n)

if i > m then B[k] = A[j]

j++;

else if j > n then B[k] = A[i]

i++;

else if A[i] < A[j]

B[k] = A[i]

i++;

else B[k] = A[j] ; j++

$T(n) = T(\frac{n}{2}) + O(n)$ & $T(1) = 1$.

Proof by induction:

Assume that given A[1,.. n] , m ST A[1..., m] & A[m+1, ..., n] are sorted then it will give us a sorted array B when n=1.

By Induction hypothesis.

A[1, ..., m] sorted → ①

A[m+1, ..., n] sorted → ②.

By the assumption "Merge" is correct and statement ①&② A[1,.. n] is sorted at the end of algo

→ At the beginning of each iteration of the for loop

① B[1,... k] is sorted

② A[i] and A[j] are least elements in the sub arrays A[1,.. m] and A[m+1,.. n] that are not being copied into B.

→ quick sort

→ Partition.

16/2/20 By choosing 5 rows:- find the k^{th} smallest element.

Quick select (A[1,... n], k)

using quick select $\left(m[1,\ldots,\frac{n}{5}], \frac{n}{10}\right)$

find a pivot q

re partition (A[1,2,.. n], q)

If $k=\gamma \Rightarrow$ return γ

if $k<\gamma$ return Q.S (A[1,.... $\gamma-1$), k)

else if $k>\gamma$ return Q.S (A [$\gamma+1,\ldots,n$], $k-\gamma$)

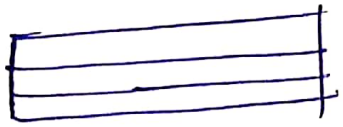
$T(n) = T(n/5) + T\left(\frac{7n}{10}\right) + cn.$

$\Rightarrow O(n)$

$|A[1,\ldots \gamma-1]| \leq \frac{7n}{10}$

By choosing 3 rows. , we get $T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + cn$

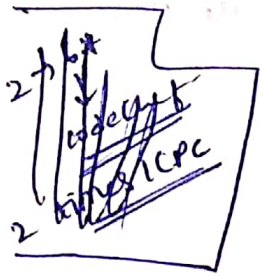
$\Rightarrow O(n \log n)$



Try with 7 rows.

→ In quick sort, for partition pivot, if we took

pivot = Quick select (A[1,2,3,..n], n/2) then the worst case T.C is O(nlogn)



Worst case running time:-

Let A be an algorithm. then the function $T: N \to N$ is the worst case running time.

$T(n) = max$

3

29/01/2020

IS:-

Insertion sort (A[1,..,n])

if n>2

IS (A[1,2,..n-1])

to insert A[n] to the sorted subarray A[1,..,n-1]

key ← A[n]

i = n-1

while (i>0 & A[i] > key)

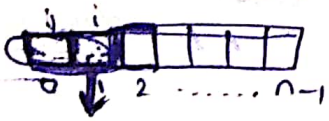
A[i+1] = A[i]

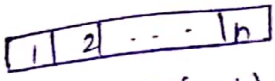
i = i-1

endwhile

A[i+1] = key.

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





$R_1(n) = R_1(n-1) + 4$

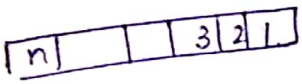
$= 4n$

$T(n) = \max_{B[1,\ldots n]} \{ \#\text{ step} = \text{takes } B[1,\ldots n] \}$

$\geq$ # step IS takes ——

$\geq R_1(n) = \Omega(n^2)$

Merge
Quick
Insert
Select
decision tree
comparison sort
nlogn
Radix sort
counting sort



$R_2(n) = R_2(n-1) + n$

$= \Theta(n^2)$.

$R_2(n) = O(n^2)$ & $R_2(n) = \Omega(n^2)$.

→ can we get sorting in time $o(n \log n)$ (little o)

→ $A[1\ldots n]$ $\xrightarrow{10^{9000}}$

construct an array $a[i]$ such that $i \in A$

↓

- Frequency array

$O(n)$ complexity.

→ Radix sort: ↑ using this to sort

456
981
324
582

981
582
324
456

324
456
981
582

324
456
582
981

↓

one digit considered

Running time? code

$\Theta(n)$

can be achieved by using 2 digits also.

Q) $A[1,2,\ldots n]$ s.t $A[i] \in \{1,2,\ldots n^2\}$ Can we sort this using $O(n)$?

no. of digits needed to represent a $(n^2) = \log_{10}(n^2)$

$\leq 2\log n$

$T(n) = 2b\log_{10}(cn)$

$= O(n \log n)$

	$\frac{1}{2}\log n$	$\frac{1}{2}\log n$	
$A[i]$	$b_1 \ldots b_{l/2}$	$b_{l/2+1} \ldots \ldots b_n$	→ no. of values $= \frac{\log n}{\log 10}$ $= n$

$b_{l/2+1} \ldots b_n$

$b'_{l/2+1} \ldots b_n$

$\vdots$

$\Big\}$ n numbers and each number $\in [0,n]$

$10^{l/2} - 1$ max number $\leq n-1$

→ Construct an array $C[0,\ldots 10]$ s.t $C[i] \leftarrow$ # of elements $\leq i$

for $i = n \rightarrow 1$

if $A[i] = j$

$B[C[j]] = A[i]$;

$C[j] = C[j] - 1$; → to preserve the order in radix sort

RAM: Random Access machine.

→ RAM model
→ Turing machine model.

Ω - Omega

→ Backtracking algo:-

SUBSETSUM

~~Input~~ Input => Natural nos (N)
Target number T ∈ N.

$$\exists P, Y \subseteq X \text{ s.t } \sum_{y \in Y} y = T$$

Eg:- {2, 4, 8, 6, 7} T = 10.

Sub set that sums to 10 ??

Algo:

```
if(T==0) return Yes;
else
    let x ∈ X
        with C = SS(X \ {x}, T-x)
        without ← SS(X \ {x}, T)
                  (x)
```

$$SS(X,T) = \begin{cases} 0 & \text{if } T=0 \\ SS(X \setminus \{x\}, T-x) \vee SS(X \setminus \{x\}, T) \end{cases}$$

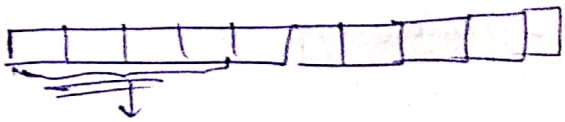
→ Text segmentation Problem:-

I/P A[1, ..., n]

Qn:- Is there a valid segmentation

IsWord(i,j) returns true if A[i][A[i+1]...A[j] is meaningful.

→



SPLIT TABLE:

SPLIT TABLE (j) = { True if A[j..n] has valid segmentation
else
False

$$\text{SPLIT TABLE}(j) = \begin{cases} \bigvee_{i=j+1}^{n} (\text{ISWORD}(j,i) \wedge \text{SPLITABLE}[i,n]) & j<n \\ \text{True} & \text{if } j>n \end{cases}$$

Let A[1...n] be global array.

SPLITABLE(K) ⟶ for sub array A[K...n]

↓

if K>n return true

for each i ∈ K+1 to n.

$t_1 \leftarrow$ ISWORD (K, i-1)

$t_2 \leftarrow$ SPLITABLE (i, n)

if ($t_1 \wedge t_2$) return true.

→ (Every recurrence has base case. (some times forgot)

Running time:-

Let us assume ISWORD() takes O(1) time

then, $\boxed{T(n) = \sum_{i=1}^{n-1} T(k) + c}$.

(-) $T(n-1) = \sum_{k=1}^{n-2} T(k) + c$

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

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

→ $O(2^n)$ ⟶ Backtracking

Other method:- Guess any function and prove by Induction

→ longest increasing subsequence:-

→ subset sum:-

Input $x \in N$, S.T

If there is a subset of x that adds to T

$$subset(x,T) = \begin{cases} True & \text{if } T=0. \\ False & \text{if } T<0 \end{cases}$$

$$ss(x \setminus \{x\}, T) \vee ss(x \setminus \{x\}, T-x)$$

* SS (x[1...,n], T)

if T==0

return TRUE;

if T < 0

return FALSE;

return (SS.(X[1.. n], T) ∨ SS(x[1...,n], T-x)

$T(n) = 2T(n-1) + 4$

$= O(2^n)$

→ Text segmentation:-

SPLITTABLE (j)

If j > n return True

for k ← j to n

If ISWORD (j,k) ∧ SPLITTABLE (k+1)

return True;

→ FAST SPLTTTABLE (A[1.... n])

S.T [1... n+1] = set to false

S.T [n+1] = TRUE

for (j ← n to 1).

for K ⇔ i ← n.

If (ISWORD (i,k) ∧ ST[k+1])

S.T[1] ← TRUE

Running time: $O(n^2)$ [DYNAMIC PROGRAMMING]

→ $a_1, a_2 \ldots, a_n$

substring $a_i \cdot a_{i+1} \ldots a_j$ for $0 \le i \le j \le n$

subsequence $a_{i_1}, a_{i_2}, \ldots a_{i_r}$ where $i_1 < i_2 < i_3 \ldots < i_r$

Longest increasing subsequence:

Input: $a_1, a_2 \ldots, a_n \in \mathbb{Z}$

Output: The length of the largest increasing subsequence.

FOOD → MONEY

EDIT DISTANCE

FOOD → MONEY

EDIT (m,n)

1) Insert
2) Remove
3) Replace

S1: A L G O R _ I _ T H M

S2: A L T R U I S T I C

If there is gap in s_1 then insert the one below it (Hint)

If there is gap in below one (s_2) then delete the corresponding char in s_1

if $A[m] = B[n]$

$$Edit[m,n] = \begin{cases} Edit(m-1, n-1) & \text{if both indices are filled} \\ Edit(m-1, n)+1 & \text{if gap in } s_2 \text{ (delete in } s_1) \\ Edit(m, n-1)+1 & \text{if gap in } s_1 \text{ (insert in } s_1) \end{cases}$$

Lemma :- let $A[1..m]$, $B[1,2..n]$ are two strings

If $A[m] = B[n]$ then $ED[m,n] = ED[m-1, n-1]$