⇒ DIVIDE AND CONQUER

→ Divide the problem into $k > 2$ sub problems.

→ Solve the sub-problems.

→ Combine the sol^n of the sub-problems to arrive at a sol^n to the original problem.

$$T(n) = \sum_{i=1}^{n} T(ni) + D(n) + C(n)$$

- **Merge sort**

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

Using master's theorem

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

- **Quick sort**

(assuming good pivot)

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

$$\Rightarrow \boxed{T(n) = O(n \log n)}$$

Matrix multiplication

→ Use divide conquer as follows:-

Divide the matrices into 4 parts.

$$\begin{bmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{bmatrix} \times \begin{bmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{bmatrix} = \begin{bmatrix} C_{00} & C_{01} \\ C_{10} & C_{11} \end{bmatrix}$$

$C_{00} = A_{00}B_{00} + A_{01}B_{10}$ $\quad C_{10} = A_{10}B_{00} + A_{11}B_{10}$

$C_{01} = A_{00}B_{01} + A_{01}B_{11}$ $\quad C_{11} = A_{10}B_{01} + A_{11}B_{11}$

~~$T(n) = 4T(n/2)$~~

A_{ij} is an $n/2 \times n/2$ submatrix for $0 \le i, j \le n$

→ 8 subproblems.

$D(n) = O(1)$

$C(n)$ = 4 matrix additions each of size $n/2 \times n/2$

$= 4\left(\frac{n}{2}\right)^2 = O(n^2)$

$T(n) = 8T(n/2) + O(n^2)$

$\Rightarrow \boxed{T(n) = O(n^3)}$

⇒ There are 2 complex no. $a+ib$ & $c+id$ to be multiplied.

$= (ac-bd) + i(ad+bc)$

- Suppose multiplications are expensive than addition.
- Can we save one multiplication & use more add/sub.?

$= (a+b)c - b(c+d) + i\left((a+b)c - a(d-c)\right)$

$P1 = ab \quad P2 = cd \quad P3 = (a+b)(c+d)$

⇒ 3 multiplications only.

⇒ So we can do less matrix multiplication & more matrix addition & matrix add^n is easier than matrix mul^n.

$P1 = A11\,(B12 - B22)$

$P2 = (A11 + A12)\,B22$

$P3 = (A21 + A22)\,B11$

$P4 = A22\,(B21 - B11)$

$P5 = (A11 + A22)(B11 + B22)$

$P6 = (A12 - A22)(B21 + B22)$

$P7 = (A11 - A21)(B11 + B12)$

$C_1 =$

$C_2 =$

$C_3 =$

$C_4 =$

$T(n) = 7T(n/2) + O(n^2)$

$T(n) = 7T(n/2) + n^2$

$a = 7 \quad b = 2$ [Case I]

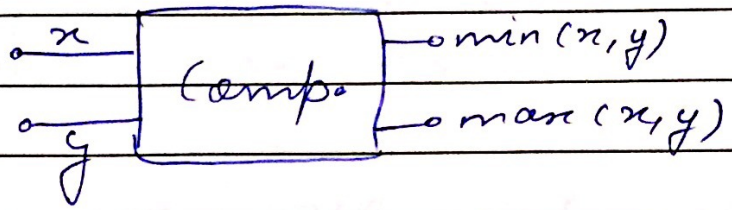
$T(n) = O(n^{\log_2 7}) = \Theta(n^{2.8})$

⇒ Bitonic Sequence

A sequence that monotonically increases & then decreases monotonically.

⇒ We will consider sorting again. But this time as a network/ hardware for sorting.

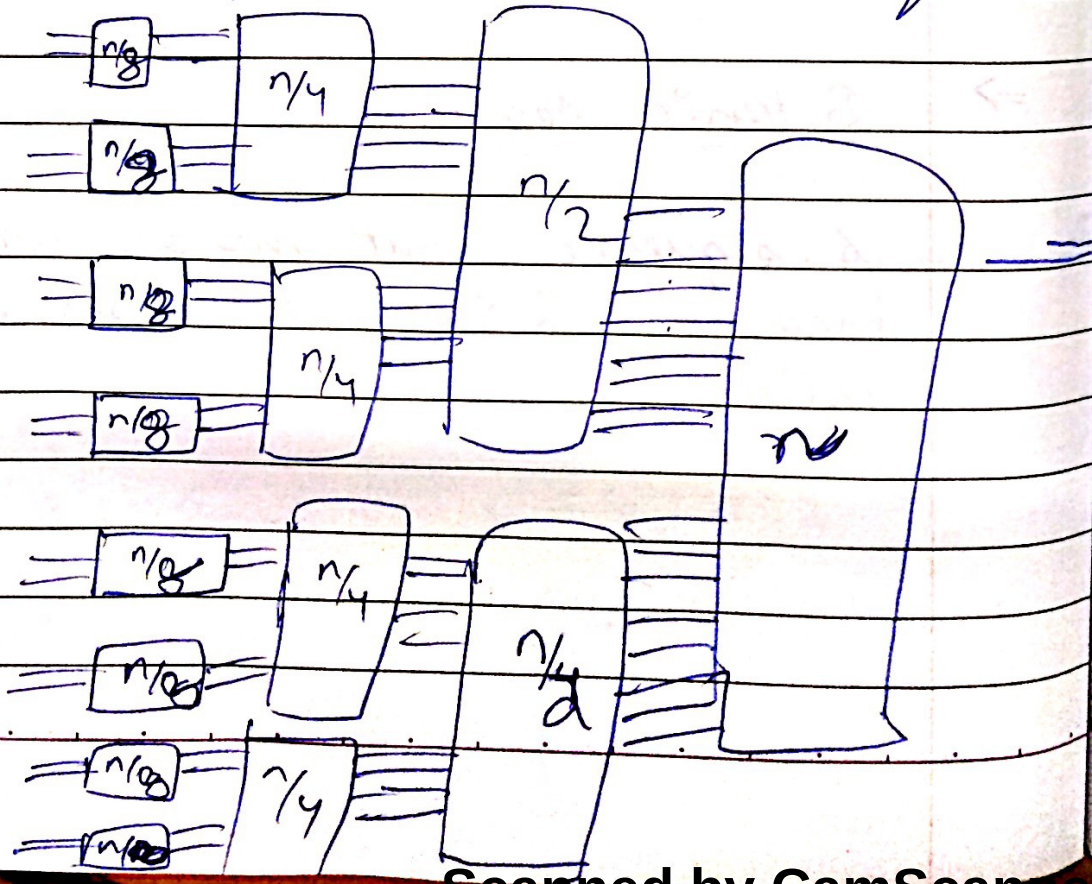
Using comparator



First we construct a sorting network that can sort bitonic sequences.

⇒

Q. If X & Y are 2 sorted sequence, how to create a bitonic sequence?



$T(n) = \Theta(\Phi^n)$, where Φ is golden ratio.

Global array X[100]

Begin

~~[illegible]~~ Store the element in an array of n.

if $x[n] = fibo[n]$

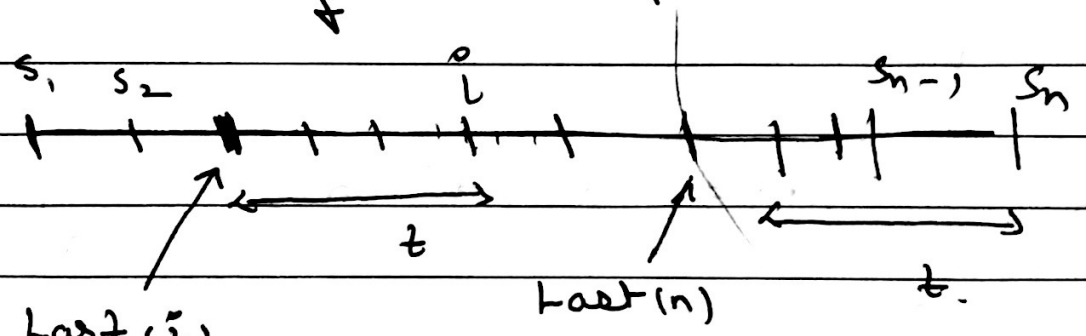
if $x[n] == 0$

then $fibo(n) = x[n-1] + x[n-2]$

~~[illegible]~~

End

⇒ The Railway station problem



best possible revenue out of railway station.

$\leftarrow R(n) = \max\{a_n + R(\text{Last}(n)), R(n-1)\}$

a_n ↙ revenue out of S_n

1. Find the runtime of the progra.

Programme MaxRevenue (int index)

Input : $a_1, a_2 \ldots\ldots, a_n$

Begin

if index = 1 return a_1.

else return max { an + MaxRevenue(last_n), MaxRevenue(n-1) }

}

T(n) = T(n-1) + T(Last(n)) + O(1)

T(n) = 2T(n-1) + O(log n)

Take worst case for Last(n)

T(n) = O(2^n)

Memoization is used to avoid the recomputing of prog.

Modified pro.

Input : a1, a2 -- an

Begin

int M[n] ; #-1

if index = 1 return a1 ;

else

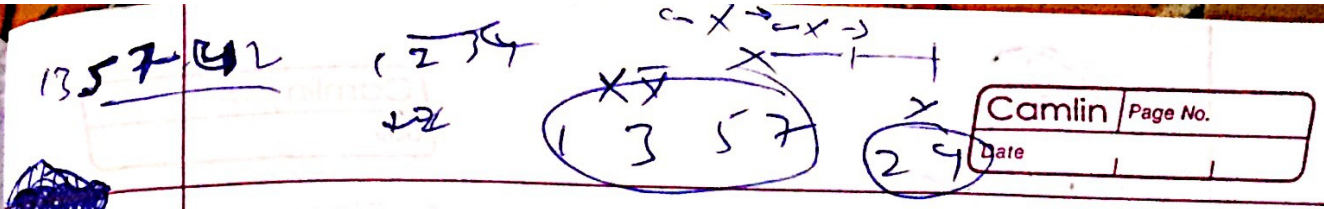
if M[index] == -1 then.

M[index] = max{ an + MaxRev (Last(n)), MaxRev (n-1)}

return M[index]

End

program runs in O(n) time



Total no. of comparators =

$$n\log n + 2\cdot n/2 \log n/2 + 4\cdot n/4 \log n/4 + 8\, n/8 \log n/8$$

$$+ \text{---------} \; n/2\,(2)\log 2$$

$$= \sum_{i=0}^{\log n} \left(\frac{n}{2^i}\log\frac{n}{2^i}\right)2^i = \sum_{i=0}^{\log n} n \log\frac{n}{2^i}$$

$$= \sum_{i=0}^{\log n} n(\log n - i) = O(n\log^2 n)$$

⇒ Divide & conquer

Date

(4)

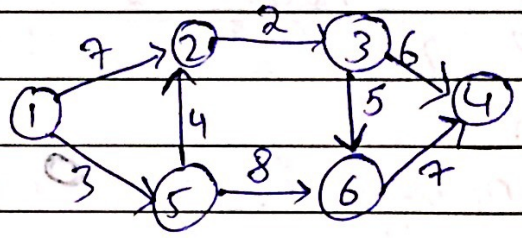
create subproblems

⇒ Dynamic Programming

↳ 'sledge hammer of algo'

⇒ Shortest path in DAG → directed acyclic graph

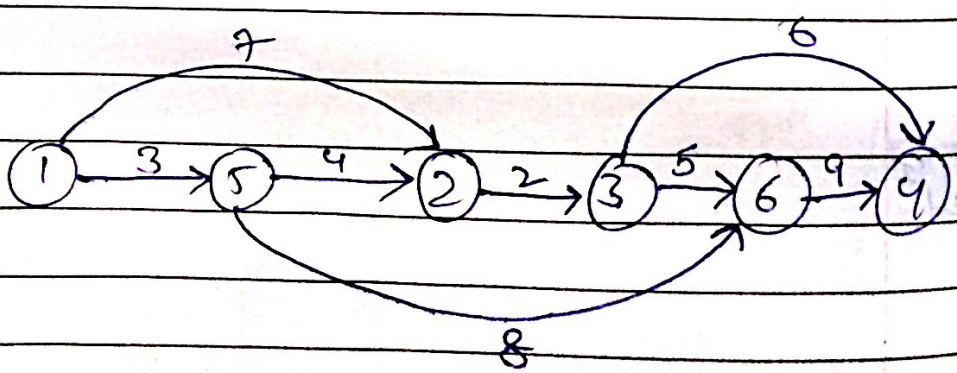
G = (V, E), acyclic, digraph



DAG has a topologically sorted order

topological sorting

1 5 2 3 6 4



Q. Find shortest path from ① to ⑥.

Now there are only two ways to ⑥ i.e. from ③ & ⑤.

So we write a recurrence.

$$L(6) = \min(5 + L(3), 8 + L(5))$$

(3 → 9, 5 → 3)

$L(3) = 2 + L(2)$	$L(2) = \min(7 + L(1), 4 + L(5))$
$L(5) = 3 + L(1)$	$= 7.$

(7; 5 → 3)

$$L(u) = \min\left(w(v,u) + L(v)\right)$$

$$\forall\ v \text{ s.t. } (v,u) \in E$$

Base case

$L(1) = 0$

=> Longest increasing subsequence (LIS)

seq: $a_1, a_2, a_3, \ldots, a_n$

Subseq. : $a_{i_1}, a_{i_2}, a_{i_3}, \ldots a_{i_j}$

s.t. $i_1 < i_2 < i_3 < \ldots < i_j$

inc. subseq. $a_{i_1} < a_{i_2} < a_{i_3} < \ldots < a_{i_j}$

Eg. 9, 7, -2, 3, 6, 5

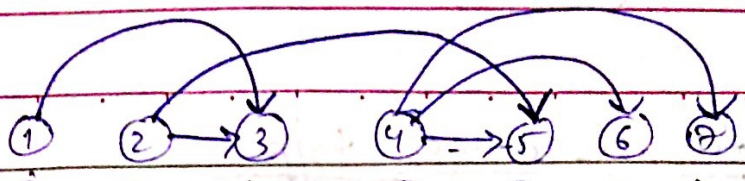
$L(i)$ = length of longest increasing subsequence ending at index i.

$L(1) = 1$

$L(j) = \}$

Draw edge (a_i, a_j) if $i < j$ & $a_i < a_j$

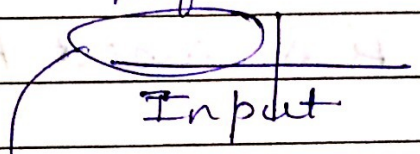
$$L(j) = \max_{(i,j) \in E(G)} (1 + L(i))$$



Q. 8, 6, 10, -3, 7, 4, 1

$L(1) = 1$	$L(5) = 2$
$L(2) = 1$	$L(6) = 2$
$L(3) = 2$	$L(7) = 2$
$L(4) = 1$	

$L(3) = \max(1 + L(1), 1 + L(2)) = 2$

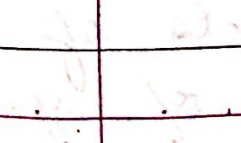


⇒ assume the problem is only prefix & solve this sub problem..

prefixes may be different

Eg. if there are 2 seq

Edit distance



chain matrix Multiplication

poly triangulisation

Edit distance type

Q. Min. no. of delete, insert operations to convert one string to another?

5

Eg.

HEART, ~~FE~~ EARTH

(2) opers.

10

Delete H & insert H.

alignment → add blanks.

15

HEART—
—EARTH

cost of alignment → no. of pos. which have a mismatch.

20

→ (2) cost of alignment

25

$E(i, j)$ = edit distance b/w first 'i' symbols of x & first 'j' symbols of y.

Camlin | Page No.
Date

We want to compute $E(n, m)$

$x = x_1\ x_2 - - - x_n$

$y = y_1\ y_2 \text{ —— } y_n$

$x_1, x_2 - - -, x_i$

$y_1, y_2 \text{ ——}, y_j$

$E(i, 0) = i$

$E(0, j) = j$

(⊗) The problem can end only in 3 cases.

x_i or — or x_i

— y_j y_i

min

$$E(i,j) = \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ E(i-1, j-1) + 1 \text{ if } x_i \neq y_j \\ E(i-1, j-1) + 0 \text{ if } x_i = y_j \end{cases}$$

n+1

	x_1	x_2	———	x_n
y_1	✓	✓		
y_2	✓			
⋮				
m+1				
y_m				

Eg whats the E.D. b/w →

SOUND & UNDER.

SOUND

		x_1	x_2	x_3	x_4	x_5
U	y_1	1	2	2	3	4
N	y_2	2	2	3	2	3
D	y_3	3	3	3	3	2
E	y_4	4	4	4	4	3
R	y_5	5	5	5	5	4

SOUND - -

- - UNDER

Q = Fractional knapsack problem. (??)

↳ can we do this using prev. approach?

Q: If NO then how would you solve this??