B - Tech 4th Semester

End - Term Examination

May 2021

Q1, 2, 3, 4, 5

Subject: Analysis and Design of Algorithms [CS2201]

Date: 19th may 2021

Name: Abhiroop Mukherjee

Exam Roll No.: 510519109

G-Suite ID: 510519109.abhirup@students.iiests.ac.in

No. of Sheets Uploaded: 12

Q2) a) For finding the connected components of a graph, we can employ two different types of data structure.

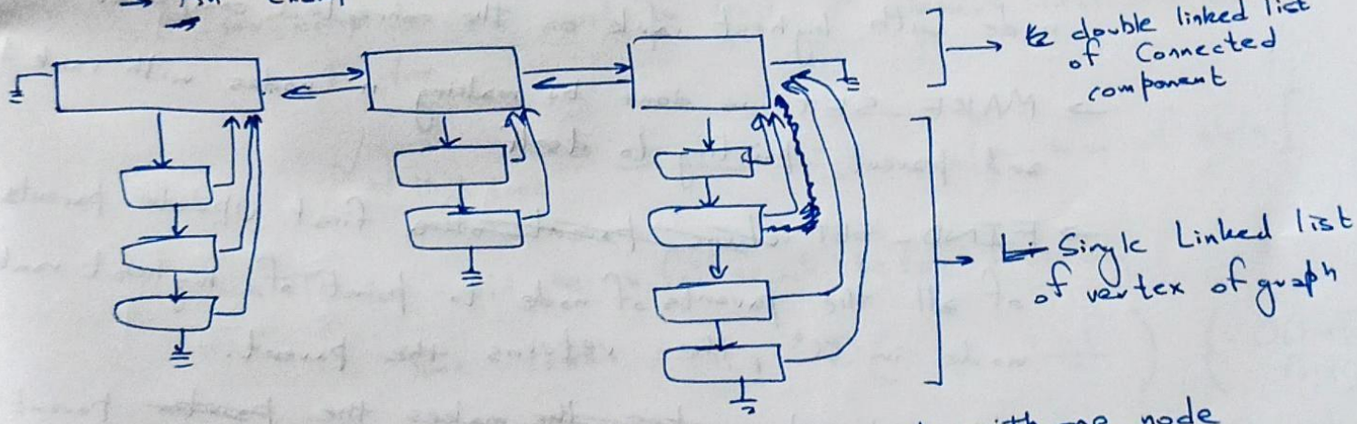
i) Link List Based

ii) Tree Based.

I) Link list based

→ Here every connected component is described as a Linked List

→ An example of LL based ~~#~~ Graph



→ We initially start with n CC's, each with one node [where n → no. of vertices] [done by MAKE_SET()]

→ we do FIND_SET() by going over all the Connected Components, in a CC, we go through all the nodes, when we find our required node, we return it's representative element [which is the Connected Component Node]

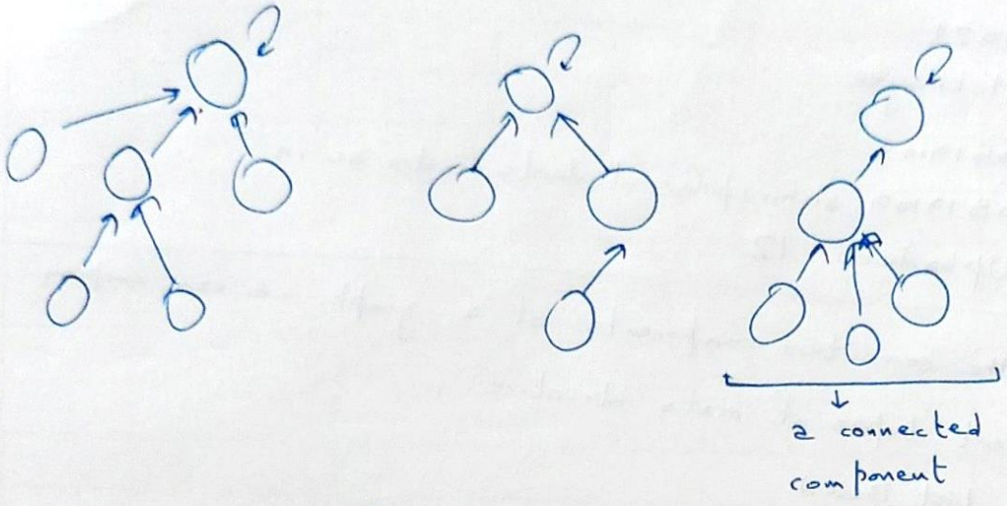
→ UNION here, shifts the lower length Vertex linked list to ~~high~~ be appended after higher length Linked list, also while changing their representative element. This is ommited if the vertices ~~they~~ are in same linked list.

→ we do UNIONS repetedly for m edges, and in the end we get connected components of the graph

II) Tree Based

→ Here we develop a tree to signify a connected component

→ Eg of tree based implementation



→ every ○ is a ~~stor~~ node, ~~with~~ along with it's rank

→ rank is a metric that is used to approximate size of tree with the node as it's root

→ every node has a parent pointer which (should) points to the node with highest rank on the connected component

→ MAKE_SET is done by making n nodes with rank 1 and parent pointing to itself

→ FIND_SET ~~returns parent of a~~ first update parents of all the parents of node to point of highest rank node in CC, the returns the parent.

→ UNION just ~~makes the~~ makes the ~~pointer~~ parent of lower ranked node to point to higher rank node, if ranks are equal, ~~the~~ one node's ranks get incremented.

Time Complexity Analysis

→ LL based structure requires update of parent pointer of all the nodes of a connected component.

It's time complexity comes out to be $O(\lg n)$

→ Tree Based Structure doesn't require such big update, it just ~~do~~ updates those which are required ~~to~~ to satisfy implementation

This saves up a lot of time

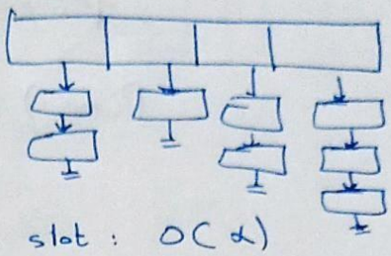
Time Complexity = $O(\lg^* n)$

→ where $\lg^* n$ is no. of $\lg$ operation required to reduce n to 1

2 & b) Dynamic Set can be implemented in two ways

i) Chaining : Uses Linked-List to handle collision
ii) Open addressing : Uses functions to calculate next probe in case of collision

I) Searching in Chaining



- i) Failure.
  - i) Time to find slot: $O(1)$
  - ii) Time to go through linked list in slot: $O(\alpha)$

$\therefore$ Time Complexity: $O(1+\alpha)$

where α is called the load factor

$$\alpha = \frac{\text{no. of elements in DS}}{\text{no. of slots in DS}}$$

$\rightarrow \alpha > 1$ generally for chaining.

ii) Success

→ consider we want to find ~~i~~ element inserted at i^{th} position

$$\alpha_{\text{at that time}} = \frac{i-1}{m} \quad [m \rightarrow \text{no. of slots}]$$

→ There would be a fail search before inserting.

$\therefore$ Time taken $= T_i = O\left(1 + \frac{i-1}{m}\right)$ $\left(\because O(1+\alpha)_{\text{fail}}\right)$

→ now this ~~i~~ i have equal probability to be any element

→ consider total n elements inserted, so

$$p_i = \frac{1}{n}$$

→ now Average Time Complexity $= \sum_{i=0}^{n} p_i T_i$

$$= \frac{1}{n} \sum_{i=1}^{n} \left[1 + \frac{i-1}{m}\right]$$

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

$$= 1 + \frac{n^2 - n - 2n}{2nm}$$

$$= 1 + \frac{n-3}{2m}$$

$$\approx O(1+\alpha) \quad \left[\alpha = \frac{n}{m} \text{ here}\right]$$

$\therefore$ Time Complexity same for both success & failure.

II) Open Addressing

i) Failure

⇒ Probability that exactly i probes access occupied slots

$$P_i = 0 \quad \text{for } i > n$$

$$P(n=i) = P(n \geq i) - P(n \geq i+1) \qquad i < n$$

$$= P_0 \sum [\text{Probability of at least } i \text{ probes required to access occupied slots}]$$

$$= \sum_i Q_i$$

→ Probability of at least i probes required to access occupied slots

$$\Rightarrow Q_i = \underbrace{\frac{n}{m}}_{\text{first insertion}} \times \underbrace{\frac{n-1}{m-1}}_{\text{second insertion}} = \frac{n-2}{m-2} \times \cdots \frac{n-i+1}{m-i+1}$$

$$\leq \left(\frac{n}{m}\right)^i \leq \alpha^i$$

$\therefore$ ~~$P(n=i)$~~

→ now $\sum_{i=0} i\, P(n=i) = \sum i \left[P(n \geq i) - P(n \geq i+1) \right]$

$$= \sum_{i=1} Q_i$$

→ Expected no. of probes for unsuccessful search $= 1 + \sum_i i P_i$

$$\leq \sum_i Q_i$$

$$\leq \frac{1}{1-\alpha} \quad [\text{GP sum}]$$

ii) Success.

→ consider $(i+1)^{th}$ element being inserted

$$\alpha_{\text{that time}} = \frac{i}{m}$$

→ It took $\frac{1}{1-\frac{i}{m}}$ no. of unsuccessful probes

→ Expected no. of probes for successful search

$$= \frac{1}{m} \sum_i \frac{1}{1-\frac{i}{m}} = \frac{m}{n} \sum_i \frac{m}{m-i}$$

$$= \frac{1}{\alpha} \left[H_m - H_{m-n} \right]$$

where $H_i = \sum_{j=1}^{i} \frac{1}{j}$

→ it been found that $\ln(i) \leq H_i \leq 1 + \ln(i)$

$$\therefore \frac{1}{n} \sum_{i} \frac{1}{i-m} = \frac{1}{n}\left[\sum_{i=1}^{m} \frac{1}{m} - \sum_{1}^{n-m} \frac{1}{n-n}\right]$$

$$\leq \frac{1}{n}\left[\ln m - \ln(n-m)\right]$$

$$\leq \frac{1}{n}\left[1 + \ln \frac{m}{n-m}\right]$$

$$\leq \frac{1}{n}\left(1 + \ln \frac{1}{1-\alpha}\right)$$

2 C) → we can define polynomial by two ways

i) Coefficient Representation

ii) Point Value Representation

let $A(n) = a_0 + a_1 n + \cdots a_n n^n$

coefficient based = $[a_0\ a_1 \cdots a_n]^T = A$

point value based ⇒

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & n_0 & n_0^2 & \cdots & n_0^n \\ 1 & n_1 & n_1^2 & \cdots & n_1^n \\ \vdots & & & & \\ 1 & n_{n-1} & n_{n-1}^2 & \cdots & n_{n-1}^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}$$

OR $Y = V A$

→ Coefficient Based Multiplication

$A(n) = a_0 + n(a_1 + n(a_2 + \cdots))$

$B(n) = b_0 + n(b_1 + \cdots n(b_2 + \cdots))$

~~→ This take $O(n^2)$ Time complexity.~~

$$C(n) = A(n)\,B(n)$$
$$= a_0 b_0 + (a_0 b_1 + b_1 a_0)n + (a_2 b_0 + a_1 b_1 + b_2 a_0)n^2 + \cdots$$

here $C[j] = \sum_{k=0}^{j} a[k]\, b[j-k]$ ⟶ convolution.

This ~~take~~ takes $O(n^2)$ Time.

Point Value Representation

→ we have Y_A, Y_B

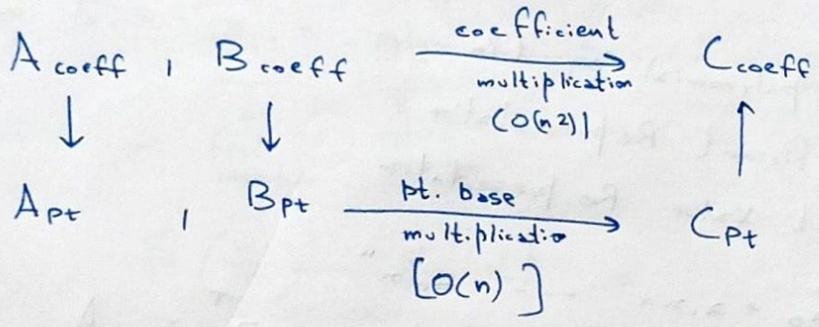
need to find Y_C

$$Y_C = Y_A \cdot Y_B$$

$$\therefore \quad Y_{Ci} = Y_{ai} \times Y_{bi}$$

→ This is element wise multiplication, so Time Complexity $O(n)$

→ So to use this property we do the following.



→ now we can do $A_{coeff} \leftrightarrow A_{pt}$, $B_{coeff} \leftrightarrow B_{pt}$

$C_{coeff} \leftrightarrow C_{pt}$ in $O(n \lg n)$ using roots of unity.

Q3) a) given $p = 561$ [It's charmichel's number]

$a = 2$

→ Do miller Rabin's Test

$p - 1 = 560 = 2^4 \times 35$

i) → $2^{35} \pmod{561} = \left[2^{32} \pmod{561} \times 2^{2} \pmod{561} \times 2^{1} \pmod{561} \right] \bmod 561$

$[35 = \underset{2^{32}}{1}\ \underset{2^{16}}{0}\ \underset{2^{8}}{0}\ \underset{2^{4}}{0}\ \underset{2^{2}}{1}\ \underset{2^{1}}{1}]$ $= \left[103 \pmod{561} \times 4 \pmod{561} \times 2 \pmod{561} \right] \bmod 561$

$= 824 \pmod{561}$

$= 263 \pmod{561}$

now $2 \pmod{561} = 2$

$2^2 \pmod{561} = 4$

$2^4 \pmod{561} = 16$

$2^8 \pmod{561} = 256$

$2^{16} \pmod{561} = 460$

$2^{32} \pmod{561} = 103$

ii) → now $263^2 \pmod{561} = 166$

$166^2 \pmod{561} = 67$

$67^2 \pmod{561} = 1$ → became 1 before 4 squares

∴ Composite.

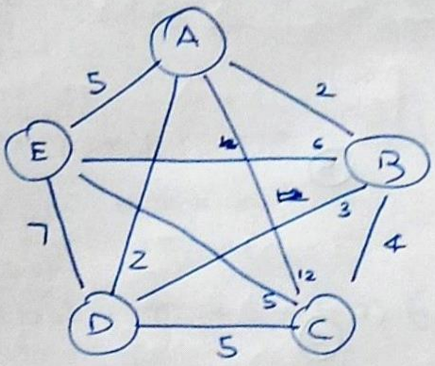
b) Roll no: 109

∴ n = 1
y = 0
z = 9

∴

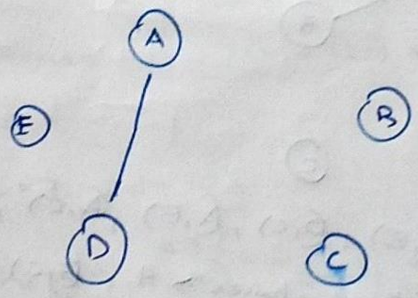
Vertex	A	B	C	D	E
A	–	2	12	2	5
B	–	–	4	3	6
C	–	–	–	5	5
D	–	–	–	–	7
E	–	–	–	–	–

So graph

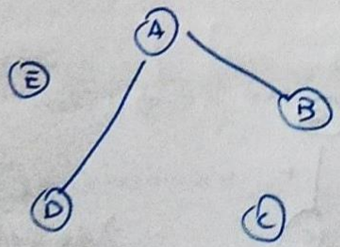


b) Kruskal's Algorithm : Take edges with lowest cost till no loop

i) (A,D)

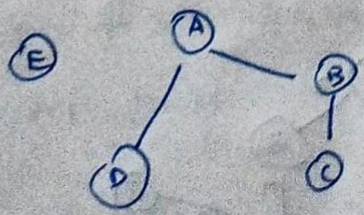


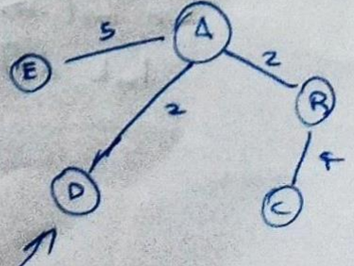
ii) (A,B)



iii) (B,D) can't be taken, makes cycle

iv) (B,C)





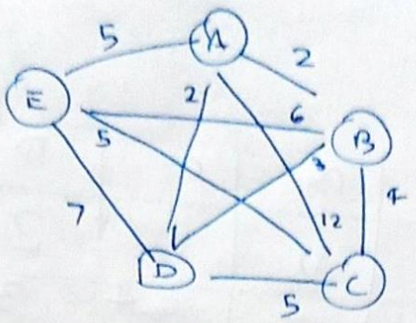
v) (B,C) can't be taken, make cycle

vi) any of (A,E) or (C,E) can be taken, taking (A,E)

vii) Taking any more will make cycle, so end.

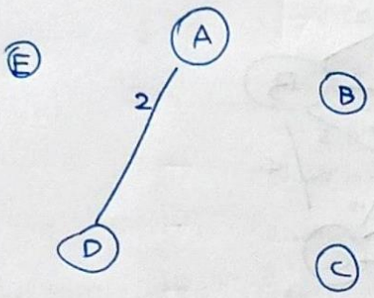
7

3)c) Redrawing Graph

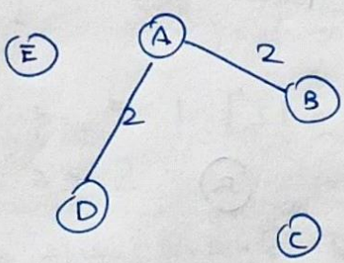


Prim's start with min. cost edge, keep adding $\overset{min}{\wedge}$ edge possible with current vertex, which doesn't create cycle.

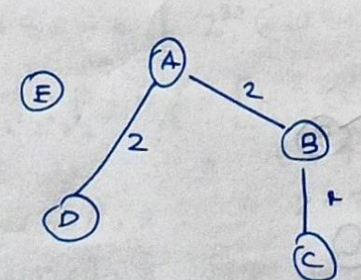
i) start with any of (A,D) or (A,B), starting with (A,D)



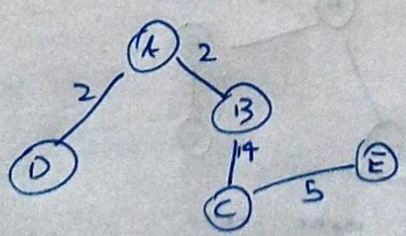
ii) (A,B) [possible choices were (A,E), (E,D), (A,B), (A,C), (D,B), (D,C), picked lowest cost]



iii) possible choices (D,E), (D,C), (A,E), (A,C), (B,C), (B,E)
min cost out of these choices ⇒ 4, (B,C)



iv) possible choices (A,E), (B,E), (C,E), (D,E)
min cost ⇒ 5 (A,E), & 5 (C,E), picking (C,E) just to make graph look different



2) given $p = 11$, $q = 13$ roll no. = 109, $n = 1$, $y = 0$, $z = 9$

→ first we calculate $n = pq = 143$

→ then we calculate $\phi(n) = (p-1)(q-1) = 10 \times 12 = 120$

→ e given → $e = 10y + z$ * 9 (near 9)

→ now we need e to be coprime with 120

let $e = 7$, as gcd $(120, 7) = 1$

→ now we need to find d, multiplicative inverse of $7 \pmod{120}$

∴ $120 = 17 \times 7 + 1 \longrightarrow$ or $1 = 120 - 17 \times 7$

→ take mod 120 on both sides

$1 \pmod{120} = 0 + (-17 \times 7) \pmod{120}$

$= [(-17 \bmod 120)(7 \bmod 120)] \pmod{120}$

$= [(103 \bmod 120)(7 \bmod 120)] \bmod 120$

$= (103 \times 7) \bmod 120$

∴ multiplicative inverse of $7 \pmod{120}$ is 103

∴ we have $n = 120$, $e = 7$, $d = 103$

so we give $P = (7, 143)$ as public key

$S = (103, 143)$ as private key

b) assuming we have only public key in access, we know that

i) $pq = 143$

ii) $e = 7$

so we first try to divide 143 as multiplication as two primes,

we get $143 = 11 \times 13$

so let $p = 13$

$q = 11$

so $\phi(n) = 12 \times 10 = 120$

now we find d as multiplicative inverse of $7 \pmod{120}$ which comes out as 103

c) we need to send 'N' as message and
'A' as signature [name is Abhivaat]

let us use ASCII values

so N = 78 = message = M
A = 65 = signature = our d (private key)
(let our n also be 143)

and we have public key (e, n) = (7, 143)

i) so we first encrypt message

$P(M) = C = M^e \pmod n$

$= 78^7 \pmod{143}$

$= [78^4 \pmod{143} \times 78^2 \pmod{143} \times 78 \pmod{143}] \pmod{143}$

$7 = 111$

now $78 \pmod{143} = 78 \pmod{143}$

$78^2 \pmod{143} = 78^6 \pmod{143}$

$78^4 \pmod{143} = 78^8 \pmod{143}$

$\Rightarrow$ so $P(M) = [78 \times 78 \times 78] \pmod{143}$

$= 9 \pmod{143}$ = C (let)

$= 9$ = C (let) $= 78$

ii) now we give signature with our private key.

$S(C) = C^d \pmod n$

$= 9^{65} \pmod{143}$ [our private key be (65, 143) (d, n)]

$= [(9^{64} \pmod{143})(9 \pmod{143})] \pmod{143}$ [in reality this n will be different]

$65 = 1000001$

$9^{64} 9^{32} 9^{16} 9^8 9^4 9^2 9^1$

now $9 \pmod{143} = 9 \pmod{143}$

$9^2 \pmod{143} = 81 \pmod{143}$

$9^4 \pmod{143} = 126 \pmod{143}$

$9^8 \pmod{143} = 3 \pmod{143}$

$9^{16} \pmod{143} = 9 \pmod{143}$

$9^{32} \pmod{143} = 81 \pmod{143}$

$\therefore S(C) = [81 \times 9] \pmod{143}$

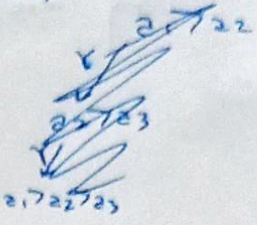
$= 14$

$S(c) = -7z^2 \pmod{193}$

$= 78$

(12) e) For a comparison based sorting of n elements, we create a comparison tree

→ no. of terminal nodes = n!



→ height of tree: n=1 → h=1 (trivial)

n=2 → h=2

n=3 → h=3

[terminal nodes 6, between 4 & 8]

generalizing

for n → if n! between 2^{b-1} & 2^b

$h = b$

height = $\lceil \lg(n!) \rceil$

or $h \geq \lg(n!)$

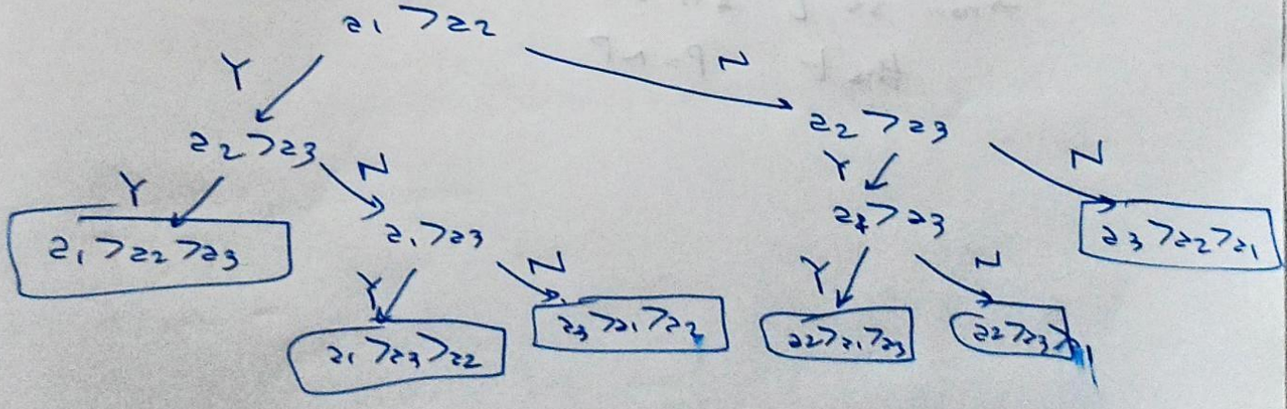
using stirling's approx = $n! \geq \left(\frac{n}{e}\right)^n$

$\therefore h > \lg\left(\frac{n}{e}\right)^n$

$h > n \lg n$

→ now h is no. of operation taken for n element

∴ lower bound = $\Omega(n \lg n)$



b) given $T(n) = 2T(\frac{n}{2}) + O(n^2)$

using master formula $T(n) = a\,T(\frac{n}{b}) + O(n^c)$

$\log_b a = 1$

$c = 2$

as $\log_b a < c$

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

c) To prove that if any NP complete set is P-time solvable, then $P = NP$

→ suppose problem $L \in P$ & $L \in NPC$

→ using propert of NPC, that $L' \leq_p L$ for every $L' \in NP$

we can say some other problem $L' \leq_p L$

→ Then such L' also belongs to P

→ now this conversion $L' \leq_p L$ doesnt decrease complexity,

is *

→ now as L' is also NP, so & $L' \leq_p L$, we can say

that $P = NP$