MTL 342.
Assignment-1.

Rahul Singh
2019MT10285.

Q 1.1 :)

(i) Show:- Definition 1 is equivalent to Definition 2.

Proof :-

Definition 1 $\Rightarrow$ Definition 2.

Assume Definition 1 holds meaning
There exists $C > 0$, s.t. $\forall n \in \mathbb{N}$.

$$f(n) \leq C \cdot g(n).$$

Now, we pick $n_0 = 1$ and with factor as C,
then $f(n) \leq C\, g(n)$ holds $\forall\, n \geq n_0 = 1$

Def^n 2 holds. This is trivial.

Now, Def^n 2 $\Rightarrow$ Def^n 1.

Assume Def^n 2 holds meaning
There exist C, n_0 s.t $f(n) \leq C \cdot g(n)$ $\forall\, n \geq n_0$.

Now, for $1 \leq n \leq n_0 - 1$, $n \in \mathbb{N}$.

Pick another factor $C' = \max\left(C, \max \frac{f(n)}{g(n)}\right)$

Now $C' \geq C \Rightarrow$

$f(n) \leq C\, g(n) \leq C' g(n)$ [where f & g are positive functions]

$f(n) \leq C' g(n)$ $\forall\, n \in \{1, 2, \ldots, n_0 - 1\}$.

Hence $f(n) \leq C' g(n)$ $\forall\, n \geq 1$.

Def^n 1 holds. Hence proved.

(ii)

(a) $n^2 = O(n^2 \ln n)$.

True.

Proof:- $\forall n \geq 3 \quad \ln(n) > 1$.

Now Pick $n_0 = 3$ and $c = 1$.

$n^2 \leq 1 \cdot n^2 \ln(n) \quad \forall n \geq 3$.

So, $n^2 = O(n^2 \ln n)$.

(b) If $f(n) = O(g(n))$ then $2^{f(n)} = O(2^{g(n)})$.

False.

Proof:- Pick $f(n) = 2\log_2 n$ and $g(n) = \log_2 n$.

Now if we choose our factor $c = 3$ and $n_0 = 1$.

then $2\log_2 n \leq 3\log_2 n \quad \forall n \geq 1$.

But, $2^{2\log_2 n} \neq O(2^{\log_2 n})$.

Assume This holds then $\exists c > 0$ s.t

$2^{2\log_2 n} \leq c \cdot 2^{\log_2 n} \quad \forall n \geq n_0$.

$2^{\log_2 n^2} \leq c \cdot 2^{\log_2 n} \quad \forall n \geq n_0$.

$n^2 \leq c \cdot n$

But this is a contradiction, since for any large enough 'c', there exist 'n' s.t

$n \geq c$.

So, It is false.

(c). $\sum_{i=1}^{n} i^8 = \Theta(n^9)$.

True.

Proof:-

claim 1:- $\sum_{i=1}^{n} i^8 = O(n^9)$.

$1^8 + 2^8 + 3^8 + \cdots + n^8 \leq \underbrace{n^8 + n^8 + \cdots n^8}_{n \text{ terms}}$.

$\leq n \cdot n^8$

$1^8 + 2^8 + \cdots + n^8 \leq n^9$

Pick $c = 1$ & $n_0 = 1$.

$\sum_{i=1}^{n} i^8 \leq 1 \cdot n^9 \quad \forall\, n \geq n_0 = 1$.

$\sum_{i=1}^{n} i^8 = O(n^9)$. — (1)

Now claim 2:- $\sum_{i=1}^{n} i^8 = \Omega(n^9)$.

$1^8 + 2^8 + 3^8 + \cdots + n^8$

By removing all elements before $(n/2)^8$, we get a lesser sum.

$\geq (n/2)^8 + \cdots + \underbrace{(n-1)^8}_{\geq (n/2)^8} + \underbrace{n^8}_{\geq (n/2)^8}$.

$\geq \underbrace{(n/2)^8 + (n/2)^8 + \cdots + (n/2)^8}_{\text{'}n/2\text{' terms}}$.

$\geq (n/2) \cdot (n/2)^8$

$\geq \left(\frac{n}{2}\right)^9 \geq n^9 \cdot \frac{1}{2^9}$.

Now Pick $c = \frac{1}{2^9}$, $n_0 = 1$.

$\sum_{i=1}^{n} i^8 \geq c \cdot n^9 \quad \forall\, n \geq n_0$.

$$\sum_{i=1}^{n} i^8 = \Omega(n^9). \quad \text{---(2)}$$

Hence From Equation (1) & (2), we get.

$$\boxed{\sum_{i=1}^{n} i^8 = \Theta(n^9)}$$

(d) $\sum_{i=1}^{n} \sqrt{i} = \Theta(n^{3/2})$.

True.

Proof:- claim 1:- $\sum_{i=1}^{n} \sqrt{i} = O(n^{3/2})$.

Now,

$\sqrt{1} + \sqrt{2} + \cdots + \sqrt{n} \leq \underbrace{\sqrt{n} + \sqrt{n} - \cdots + \sqrt{n}}_{n \text{ terms}}$.

$\leq n \cdot \sqrt{n}$.

$\sum_{i=1}^{n} \sqrt{i} \leq n^{3/2}$

Pick $c = 1$ & $n_0 = 1$.

$\sum_{i=1}^{n} \sqrt{i} = 1 \cdot n^{3/2}$ $\forall n \geq n_0 = 1$.

$\sum_{i=1}^{n} \sqrt{i} = O(n^{3/2})$ ---(1)

claim 2:- $\sum_{i=1}^{n} \sqrt{i} \in \Omega(n^{3/2})$.

$\sqrt{1} + \sqrt{2} + \cdots + \sqrt{n}$

Remove all element before $(n/2)^{1/2}$ term,

$\sum_{i=1}^{n} \sqrt{i} \geq \underbrace{(n/2)^{1/2} + \cdots + (n-1)^{1/2} + \sqrt{n}}_{\text{'}n/2\text{' terms}}$.

$$\sum_{i=1}^{n} \sqrt{i} \geq (n/2)^{1/2} + (n/2)^{1/2} + \cdots + (n/2)^{1/2}.$$

$$\geq n/2 \cdot (n/2)^{1/2}.$$

$$\sum_{i=1}^{n} \sqrt{i} \geq (n/2)^{3/2}$$

Pick $c = \frac{1}{2^{3/2}}$ & $n_0 = 1$.

$$\sum_{i=1}^{n} \sqrt{i} \geq \frac{1}{2^{3/2}} n^{3/2} \quad \forall n \geq n_0 = 1.$$

$$\boxed{\sum_{i=1}^{n} \sqrt{i} = \Omega(n^{3/2})} \quad \text{---} (2)$$

From (1) & (2), we get

$$\boxed{\sum_{i=1}^{n} \sqrt{i} = \Theta(n^{3/2})}$$

(iii)

Pick

$$f(n) = \begin{cases} n & \text{if } n \text{ is even} \\ 1 & \text{if } n \text{ is odd} \end{cases}$$

$$g(n) = \begin{cases} 1 & \text{if } n \text{ is even} \\ n & \text{if } n \text{ is odd.} \end{cases}$$

Now $\frac{f(n)}{g(n)} = n$ when n is even. — (1)

$\frac{g(n)}{f(n)} = n$ when n is odd. — (2)

Here both equations (1) & (2) cannot be bounded.

Question 1.2.

⇒.

Assume tree is rooted at 's'.

Define $p(v)$:= Length of longest path passing through a vertex 'v'.

Claim:- $\max p(v) \quad \forall v \in V$ gives the diameter.

Proof:-

Define $h(v)$:= Length of longest path from the vertex 'v' to a node in its subtree.

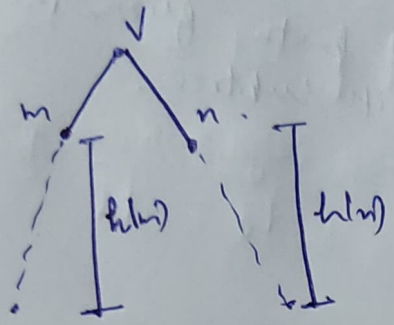
Pseudo code to compute $p(v)$.

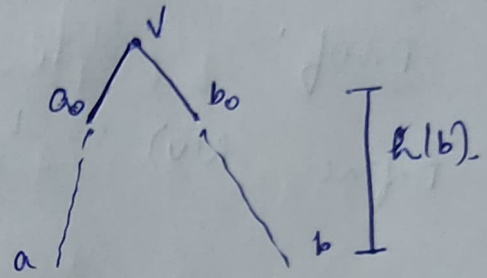
```
function dfs(v):
    for all children w of v:
        dfs(w)
        compute longest and second longest path from children of v.
    end for.
    h(v) = |Longest path from children of v| + 1.
    if v is a leaf:
        h(v) = 0
    end
    if v has atleast 2 children:
        p(v) = |Longest path| + |Second longest path| + 2.
    end
    if v has only 1 child:
        p(v) = |Longest path| + 1.
    end
end function.
do dfs(v) for all v ∈ V.
dia = max(p(v)) for all v ∈ V.
return dia.
```

Running Time := $\underbrace{O(m+n)}_{\text{for dfs}} + \underbrace{O(n)}_{\text{for computing maximum}}$.

$= O(m+n)$.

Claim:- $\beta(v) = \begin{cases} h(m) + h(n) + 2 & \text{if } v \text{ has atleast 2 childrens.} \\ h(m) + 1 & \text{if } v \text{ has one child.} \end{cases}$





Assuming m & n are v's two children with longest and second longest path.
(here second longest path might be equal to longest path).

Proof by contradiction:

Assuming the longest path is $a \to v \to b$.
where a_0 and b_0 are corresponding childrens of 'v'.

Now, $|a_0 \to a| \le h(a_0)$ & $|b_0 \to b| \le h(b_0)$

$|a \to v \to b| = |a \to v| + |v \to b| + 2$.

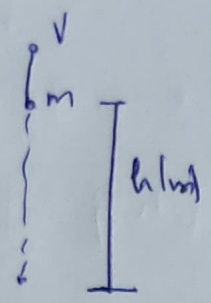
$\le h(a_0) + h(b_0) + 2$.

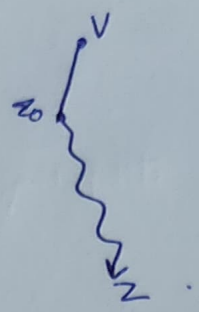
$\le h(m) + h(n) + 2$. (contradiction),

(As $h(m)$ & $h(n)$ are longest & second longest path length)

But in case of single child, $h(n) = 0$ (second longest path length).

Now m is the child with longest path length.





Assuming ~~v~~ longest path is $v \to z$ with

~~this holds~~ $|v \to z| > h(w) + 1$ ..

But $|v \to z| = |v \to z_0| + |z_0 \to z|$

$\leq 1 + h(z_0)$ ∵ (By def^n of 'h')

$\leq 1 + h(m)$ ∵ (As $h(m)$ is longest path among child of v)

(Contradiction)

Now we have proved $p(v)$ gives the longest path length containing 'v' (or passing through v).

Now $\max_{v \in V}(p(v))$ = gives the longest path length for all v vertices considered.

= Diameter of a tree.

Question 1.3

Solution. Algorithm :-

Step 1 :- start BFS from arbitrary vertex 's'.

Step 2 :- If one of the layers contains only one vertex, then that vertex is output.

Time :- $O(n+m)$ ← (BFS takes atmost this amount of time).

Claim :- Such a vertex exist.

Proof by contradiction.

Assume there no such vertex, meaning all layers of the BFS tree have atleast 2 vertices.

Since, there are $m > n/2$ layers apart from 's'.

$\{L_1, L_2 - - - L_{m'}\}$.

where m = distance between s and t.

No. of vertices $= 1 + |L_1| + |L_2| + - - + |L_m|$.
$\geq 1/2 \quad \geq 1/2 \quad \geq 1/2$.

$\geq 1 + 2m \quad \because (m > n/2)$

$\boxed{n \geq 1+n.}$

Contradiction.

Show :- Such a vertex is the answer we need.

Let v belong to the single vertex layer L_i.

Claim : There will be no path a vertex in X and a vertex in Y on deleting 'v'

where $X = \bigcup_{j<i} L_i$, $Y = \bigcup_{j>i} L_i$.

Proof :- By Contradiction.

Assume there is a path from $s \in X$ to $t \in Y$ on deleting 'v'.

Let function $I(x) :=$ Index of Layer in which 'x' belongs.

NOW $I(s) < i$ ($\because s \in X$)

$I(t) > i$ ($\because t \in Y$).

$\Rightarrow \exists w$ (vertex) s.t $\boxed{I(w) = i}$.

But 'v' was the only element in L_i & v was deleted

$\Rightarrow$ $\boxed{w = v}$.

Contradiction.

Pseudo code :-

$L_0 = \{s\}$.

$i = 0$

for v in L_i

for each neighbor t of v.

if not marked[t]

$L_{i+1} = L_{i+1} \cup \{t\}$

marked[t] = True

$i = i+1$.

end

End

Time = O(n+m)

for $i = 1$ to n

if $|L_i| = 1$

return u, where $u \in L_i$

break

end

Time = O(n)

Total time = $O(n+m)$.

Question 1.4

(a)

→ Counter example.

Take 40 paisa.

Greedy approach,

$40 = 25 + 10 + 5$. (3 coins)

Optimal,

$40 = 20 + 20$ (2 coins).

(b) Holds

Proof by Induction.

Induction Base step :-

Greedy Algorithm is optimal on the coin set $1, 4$

Trivial.

Hypothesis :-

Greedy is optimal on coin set $1, 2, 2^2 \dots 2^j, 4$.

Induction step :-

Show :- Greedy algo is optimal on coin set $1, 2 \dots, 2^{j+1}$.

Let the sum to be made by coins be 'X'.

Case 1 :- $X < 2^{j+1}$,

Greedy Algo picks coins from set $1, 2 \dots 2^j$.

By hypothesis,

Greedy Algo is optimal.

Case 2 :- $X = 2^{j+1}$

Greedy select 2^{j+1} coin.
Optimal will also select 2^{j+1} coin as there is no other choice.

So, Greedy algo is optimal.

Case 3 :- $X > 2^{j+1}$.

Now let
$n_{j+1} \geq 0$ be the coins of 2^{j+1}, Greedy algo takes.
$m_{j+1} \geq 0$ " " " " " , Optimal algo takes.

Claim :- Greedy algo does not do worse than optimal solution, meaning No. of coins by greedy algo $\leq$ No. of coins by optimal way.

Proof by contradiction, Assume optimal way takes strictly less coin.
WLOG $m_{j+1} < n_{j+1}$.
where n_{j+1} is the maximum no. of coins that can be taken.

Then
$$0 \leq X - m_{j+1} \cdot 2^{j+1} > 2^{j+1} \quad \text{---(1)}$$
$$0 \leq X - n_{j+1} \cdot 2^{j+1} < 2^{j+1} \quad \text{---(2)}$$

Now by case 1 analysis, we can say $X - n_{j+1} \cdot 2^{j+1}$ gives optimal solution by greedy algo.

Let $X - m_{j+1} \cdot 2^{j+1}$ be Y
$X - n_{j+1} \cdot 2^{j+1}$ be Z.

clearly, from eqn (1) & (2), we can say,
$Z < Y$.

Now, we calculate no. of coins reamined by Z & Y by optimal way.

Since $z < y$,

There is a possibility that

$O(z) > O(y)$

$[O(z) - O(y)] > 0$ — ③

[$O(x)$ represent no of coins required if we go by optimal way]

but we know.

$m_{j+1} < n_{j+1}$. $\Rightarrow [n_{j+1} - m_{j+1}] > 0$. — ④ .

So, total coins by optimal way . $m_{j+1} + O(y)$.

" " by greedy way $n_{j+1} + O(z)$.

Adding ③ & ④ .

$[O(z) - O(y)] + [n_{j+1} - m_{j+1}] > 0$.

$O(z) + n_{j+1} > O(y) + m_{j+1}$.

Total coins by optimal < total coins by greedy

(Contradiction .) .

Hence for $x > 2^{j+1}$, greedy algo is optimal .

Question 1.5

Solution.

(a) Algo is Incorrect.

Counterexample:-

Pick

$l = 1$	3
$p = 0.2$	0.8

&

Now, according to Algorithm, file with length 2 is before length 3 file.

Expected access time $= 1 \times 0.2 + 4 \times 0.8$

$= 3.4.$

Now optimal solution would be file with length 3 is accessed before.

$= 3 \times 0.8 + 4 \times 0.2$

$= 3.2.$

(b) Algo is Incorrect.

Counterexample.

Pick

$l = 80$	2
$p = 0.7$	0.3

According to algo

Expected access time $= 80 \times 0.7 + 82 \times 0.3$

$= 80.6.$

Now optimal solution, in this case file with $p = 0.3$ is chosen before file with $p = 0.7$.

Expected Access Time $= 2 \times 0.3 + 82 \times 0.7$

$= 58.$

(c)

Algorithm is correct.

Proof :- By exchange argument.

Since $\frac{l_j}{p_j} > \frac{l_i}{p_i}$

$$l_i p_j - l_j \cdot p_i < 0 .$$

This exchange reduces the expected access time.

Thus, this algo of arranging in $\frac{l_j}{p_j} > \frac{l_i}{p_i}$ gives optimal solution.

Question 1.6

(a)

Counterexample :-

Pick $A = \{1.5, 2, 2.5, 3, 3.5\}$.

Now, using the algo approach, it might select $[2,3]$ as it covers the most number of points in A.

After removing these points,

$A = \{1.5, 3.5\}$.

Then we have to take two intervals $[1,2]$ and $[3,4]$

So, in total 3 intervals are required.

But optimal approach, choose $[1.5, 2.5]$ and $[3,4)$ intervals.

Only 2 intervals are required.

(b) This strategy is optimal.

Proof :- By contradiction.

Consider the strategy does not produce an optimal solution.

Let the optimal solution be O.

Let the solution that strategy outputs be S.

Then we have $|O| < |S|$.

Let the solution that strategy outputs $\{c_1, c_2 \dots c_{|S|}\}$ ordered by the left end of the intervals.

Let the optimal solution be $\{k_1, k_2 \dots k_{|O|}\}$ ordered by the left ends of the intervals.

Now $|O| < |S|$.

Then $L(K_{|O|}) \leq L(C_{|O|}) < L(C_{|O|+1}) - 1$

∵ [where $L(C)$ represent left end of interval C]

Proof :- $L(K_i) \leq L(C_i) \quad \forall i \leq \min(|O|, |S|)$.

Base step :- $L(K_1) \leq L(C_1)$.

In our strategy, $L(C_i) \in A \quad \forall i \in [1, 2, \ldots, |S|]$

This means $L(C_1) = a_j$ where $a_1 \in A$ (given in question)

Now $L(K_1) \leq a_1$

If $L(K_1) > a_1$, then optimal solution will require another to cover first element a_1 of A.

where $\{K_1, K_2 \ldots K_{|O|}\}$ are sorted according to start time.

hence $L(K_1) \leq a_1 = L(C_1)$

so, $L(K_1) \leq L(C_1)$.

Hypothesis :- $L(K_i) \leq L(C_i) \quad \forall i \leq m < \min(|O|, |S|)$.

Induction step :- $L(K_{m+1}) \leq L(C_{m+1})$.

In our strategy, we choose

$L(C_{m+1}) = a_j$

This means $a_j > L(C_m) + 1$ ∵ (a_j does not belong to previous interval).

Now Assume $L(K_{m+1}) > a_j$

But $a_j > L(C_m) + 1 \geq L(K_m) + 1$

This means a_j is not included in any of K_i's intervals as

$L(K_m) + 1 \leq a_j < L(K_{m+1})$.

As a_j is not included blw K_m and K_{m+1} Intervals.
& $\{K_1 \cdots K_{101}\}$ are sorted by ~~slot~~ start time.

So, This is contradiction as optimal solution left a_j.

NOW:-

$$L(C_i) < L(C_{i+1}) - 1 \quad \forall\ i \in [1, 2 \cdots |S|].$$

Reason:- C_i's are Intervals.

$\therefore$ ~~L(C_{i+1})~~ C_{i+1} is ~~not~~ an Interval.

So, $L(C_{i+1}) > L(P) + 1$.

$$\boxed{L(C_i) < L(C_{i+1}) - 1}$$

NOW we come back to ~~our~~ Induction proof.

we know

$$L(K_{101}) \leq L(C_{101}) < L(C_{101+1}) - 1.$$

In Our strategy, $L(C_j) \in A \quad \forall\ j \in [1, \cdots |S|]$.

So, $L(C_{101}+1) \in A$.

But $L(C_{101}+1) > L(K_{101}) + 1$.

Since K is sorted,

$L(C_{101}+1) \in A$ is missed out by optimal solution.

Contradiction:

Hence proved that strategy always produce optimal solution.