Exam Date: 29/2/2024 MSE Solutions. Acad. Yr: 2023-24

Sem IV Design & Analysis of Algorithms (20CP209T)

Q1 (A) — (i) — $\Theta(\log_a \log_b n)$ — [1M]

$T(n) = T(n^{1/a}) + 1 \qquad T(b) = 1$

$= T(n^{1/a^2}) + 1 + 1$

$= T(n^{1/a^3}) + 1 + 1 + 1$

$\vdots$ After i^{th} iteration

$= T(n^{1/a^i}) + 1 + 1 + \dots + 1$

$= T(b) + 1 + 1 + \dots + 1$

$= i \cdot 1 = i$

[2M]

$n^{\frac{1}{a^i}} = b \longrightarrow \log_b \Rightarrow \frac{1}{a^i} \cdot \log_b n = 1$

Again $\log_b n = a^i$

Taking $\log_a \Rightarrow$ $\boxed{i = \log_a \log_b n}$ — (i)

Q1 (B) — (iii) I or III or IV but not II — [1M]

Let $f(n) = \sum_{i=0}^{n} i^3 \quad \therefore f(n) = \frac{n^2(n+1)^2}{4} = \frac{n^4 + 2n^3 + n^2}{4}.$

Thus:

[2M]

✓ $f(n) = \Theta(n^4)$ as $c_1 n^4 \le \frac{n^4 + 2n^3 + n^2}{4} \le c_2 n^4$ $\quad c_1 = 1 \quad c_2 \ge 2$

✓ $f(n) = O(n^5)$ as. $\frac{n^4 + 2n^3 + n^2}{4} \le c_1 n^5 \quad c_1 = 1$

✓ $f(n) = \Omega(n^3)$ as $\frac{n^4 + 2n^3 + n^2}{4} \ge c_1 n^3 \quad c_1 = 1$

✗ $f(n) \ne \Theta(n^5)$ as $c_1 n^5 \le n^4 + 2n^3 + 2n^2 \le c_2 n^5$

Q1 C (1M) (i) (ii) and (iii) all are correct but not (iv)

$f(1) = 1$ $\quad f(2n) = 2f(n) - 1$ $\quad f(2n+1) = 2f(n) + 1$

[2M]

for $n=1$ $\quad f(2) = 2f(1) - 1 = 1$ $\quad f(3) = 2f(1) + 1 = 3$

for $n=2$ $\quad f(4) = 2f(2) - 1 = 1$ $\quad f(5) = 2f(2) + 1 = 3$

for $n=3$ $\quad f(6) = 2f(3) - 1 = 5$ $\quad f(7) = 2f(3) + 1 = 7$

for $n=4$ $\quad f(8) = 2f(4) - 1 = 1$ $\quad f(9) = 2f(4) + 1 = 3$

for $n=5$ $\quad f(10) = 2f(5) - 1 = 5$ $\quad f(11) = 2f(5) + 1 = 7$

i) $f(2^n - 1) = 2^n - 1$ $\quad$ for $n=1$ $\quad f(1) = 1$ $\quad$ Base proved

Assume true for $m = n-1$

$f(2^{n-1} - 1) = 2^{n-1} - 1$

for $m=n$ $\quad f(2^n - 1) = 2 \cdot f\left(\frac{2^n - 2}{2}\right) + 1$

$= 2 \cdot f(2^{n-1} - 1) + 1$

$= 2(2^{n-1} - 1) + 1$

$= 2^n - 1$ — Proved

ii) $f(2^n) = 1$ $\quad$ for $n=0$ $\quad f(1) = 1$ $\quad$ Base proved

Assume true for $m = n/2$

$f(2^{n/2}) = 1$

for $m=n$ $\quad f(2^n) = 2 \cdot f(2^{n/2}) - 1$

$= 2 \cdot 1 - 1$ $\quad$ Proved

iii) $f(5 \cdot 2^n) = 2^{n+1} + 1$

for $n = 0 \rightarrow f(5) = 2^{0+1} + 1 = 3$. Base proved

Assume true for $m = n-1$

$$f(5 \cdot 2^{n-1}) = 2^n + 1$$

for $m = n$

$$f(5 \cdot 2^n) = 2 \cdot f\left(\frac{5 \cdot 2^n}{2}\right) - 1$$

$$= 2 \cdot f(5 \cdot 2^{n-1}) - 1$$

$$= 2 \cdot [2^n + 1] - 1$$

$$= 2^{n+1} + 1 \text{ — Proved}$$

iv) $f(2^n + 1) = 2^n + 1$

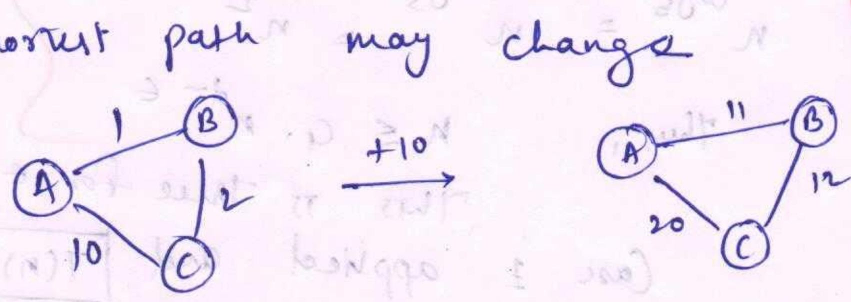
for $n = 0 \rightarrow f(2) = 2$ — Base case failed

Hence (i) (ii) & (iii) are correct

Q1(D) — (i) — only X — [1M]

MST does not change as weights are increased or doubled. Since each weight is distinct, there will be exactly 1 MST. Thus if weights are increased/doubled, they increase in proportion, means edges will be in same sorted order.

But. Shortest path may change [2M].



Q2 (A) $\boxed{O(n)}$ — [1M]

In selection sort, we select minimum element and swap it with first element.

We repeat this for remaining n-1 elements. Thus '1' swap for each iteration. Total, we have n-1 iterations. Hence, total no. of swaps would be n-1. Hence, O(n). [2M]

Q2 (B) $\boxed{T(n) = 2T(n/2) + \log n}$ — [1M]

```
if(n>0)
{ for (i=1; i<=n ; i=i*2)   → one for loop from 1 to n but i is doubled after every iteration.
      printf("PDEU");
  return (f(n/2) + f(n/2))   → Two recursive calls of size n/2
}.
```

[2M]

Thus, for loop runs for i iterations.

$1, 2, 4, \ldots 2^i \Rightarrow 2^i = n \qquad i = \log n$

$T(n) = 2T(n/2) + \log n$

Q2 (C) $\boxed{\Theta(n^2)}$ — [1M]

$T(n) = 9T(n/3) + n.$

$a = 9 \quad b = 3 \quad f(n) = n$

$n^{\log_b 9} = n^{\log_3 9} = n^2$

Thus, $n \le c_1 \cdot n^{2-\epsilon}$

This is true for $\epsilon = [0, 1]$.

Case 1 applied and $\boxed{T(n) = \Theta(n^2)}$

[2M]

Q2 (D) $\boxed{221}$ — [1M]

	A	B	C	D	E
Elements	10	20	15	30	25

To merge two arrays of n_1 and n_2 elements, no. of comparisons required in worst case is $n_1 + n_2 - 1$ if arrays are sorted.

Optimal algo will always start merging the arrays from smaller size.

A C B E D

10 15 20 25 30

A-C merged, no. of comparison = $10+15-1 = \underline{24}$

size = 25

AC B E D

25 20 25 30

AC - B merged, no. of comparison = $25+20-1 = \underline{44}$

size = 45

E D ACB

25 30 45

E - D merged, no. of comparison = $\underline{54}$

size = 55

AcB ED

45 55

final merged no. of comparison = 99

[2M]

Q3

Given operations

[2M] for all 4 correct / Else [1M]

PUSH(S, x) —— O(1)

POP(S) —— O(1)

MULTIPUSH(S, k, x) —— O(min(k, n−s))

MULTIPOP(S, k) —— O(min(k, s))

's' elements already in stack.

'n' is size of stack

In worst case, we may have sequence of multipush and multipop operations alternately on an empty stack

for ex

MULTIPUSH(S, k, x) — O(k)
MULTIPOP(S, k) —— O(k)
MULTIPUSH(S, k, x) —— O(k)
MULTIPOP(S, k) —— O(k)
⋮
MULTIPUSH(S, k, x) —— O(k)
MULTIPOP(S, k). —— O(k)

[2M]

Assuming $k < n$.

[2M]

Thus $\boxed{T(N) = \underbrace{O(k) + O(k) + \cdots + O(k)}_{\text{'N' times}}}$

where 'N' is no. of operation.

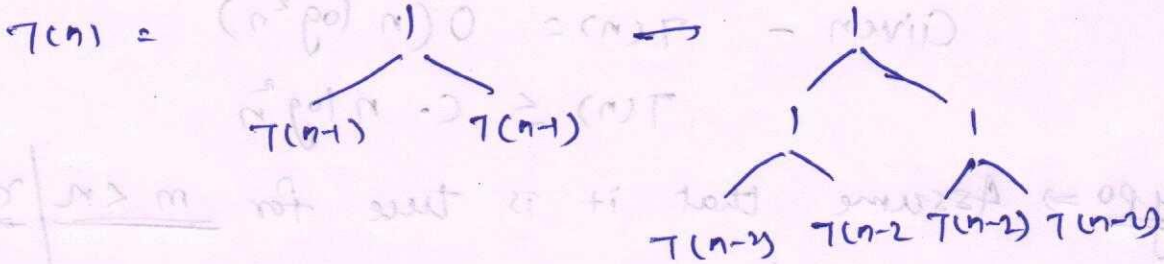
$\therefore \boxed{T(N) = N \cdot O(k)}$

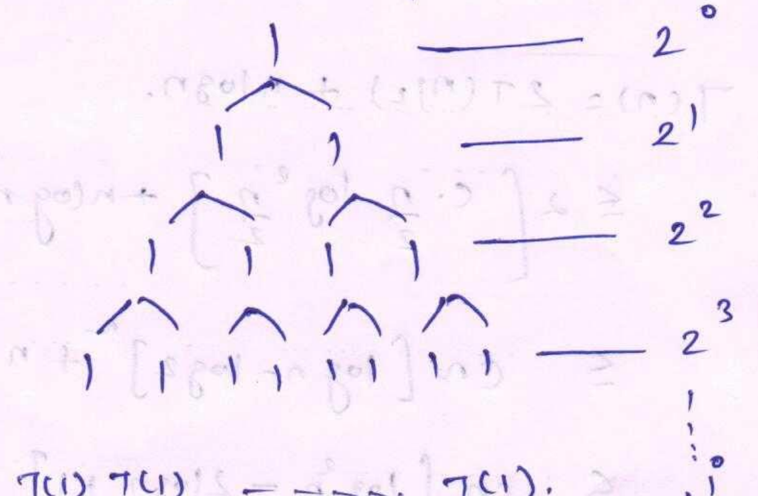
if $k = n$

$\boxed{T(N) = O(n \cdot N)}$

Avarge Amortized cost $= \dfrac{T(N)}{N} = \dfrac{O(n \cdot N)}{N} = O(n)$

Q4 (A) $T(n) = 2T(n-1) + 1$





[2M] for correct tree upto leaf node

2^i at i^{th} level.

$i = 0$ to $n-1$

$$T(n) = \sum_{i=0}^{n-2} 2^i + 2^{n-1} \cdot T(1). \quad \text{Assume } T(1) = 1$$

$$= \sum_{i=0}^{n-2} 2^i + 2^{n-1}$$

$$= \underbrace{2^0 + 2^1 + 2^2 + \dots + 2^{n-2}}_{GP} + 2^{n-1}$$

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

[2M] for correct calculation with separate leaf node

[1M] → $\boxed{T(n) = O(2^n)}$ ⟶ Guess.

Qu (B) $T(n) = 2T(n/2) + n\log n$

Given - $T(n) = O(n \log^2 n)$

$T(n) \le c \cdot n \log^2 n$

Hypo ⇒ Assume that it is true for $m < n$ / $m = n/2$

Thus, $T(n/2) \le c \cdot \frac{n}{2} \log^2 \frac{n}{2}$ — [1M]

Induction ⇒

[2M]

$$T(n) = 2T(n/2) + n\log n.$$

$$\le 2\left[c \cdot \frac{n}{2} \log^2 \frac{n}{2}\right] + n\log n$$

$$\le cn\left[\log n - \log 2\right]^2 + n\log n$$

$$\le cn\left[\log^2 n - 2\log n + 1\right] + n\log n$$

$$\le \underline{cn\log^2 n} - 2cn\log n + cn + n\log n$$

$$\le cn\log^2 n - [\underbrace{2cn\log n - cn - n\log n}_{\text{positive.}}]$$

Thus,

$$2cn\log n - n\log n - cn \ge 0$$

$$2cn\log n - n\log n \ge cn.$$

$$(2c-1)\cancel{n}\log n \ge c\cancel{n}$$

$$2c\log n - \log n \ge c$$

$$2c\log n - c \ge \log n$$

$$c[2\log n - 1] \ge \log n$$

$$c \ge \frac{\log n}{2\log n - 1} \quad n \ge 2$$

[2M].

$$\boxed{c \ge \frac{1}{2 - 1/\log n}} \qquad \boxed{c \ge 1} \quad n=2$$

Q4 OR

$T(n) = \sqrt{n}\, T(\sqrt{n}) + n.$

[1M] $\rightarrow$ Let $n = 2^m \longleftrightarrow m = \log n$

$T(2^m) = 2^{m/2}\, T(2^{m/2}) + 2^m$

[2M]

$S(m) = 2^{m/2}\, S(m/2) + 2^m$

$= 2^{m/2}\left[2^{m/4} \cdot S(m/4) + 2^{m/2}\right] + 2^m$

$= 2^{m/2} \cdot 2^{m/4} \cdot S(m/4) + 2^m + 2^m$

$= 2^{m/2}\, 2^{m/4}\left[2^{m/8}\, S(m/8) + 2^{m/4}\right] + 2^m + 2^m$

$= \underbrace{2^{m/2}\, 2^{m/4} \cdot 2^{m/8}}_{\mathrm{I}}\, S(m/8) + \underbrace{2^m + 2^m + 2^m}_{\mathrm{II}}$

Total levels $\frac{m}{2^i} = 1$ $\quad i = \log m$

[2M]

(I) $\Rightarrow$ $2^{m/2} \cdot 2^{m/4} \cdot 2^{m/8} \cdot \ldots \cdot 2^{1} = 2^{\frac{m}{2} + \frac{m}{4} + \frac{m}{8} + \cdots + 1}$

$= 2^{m\left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots\right)} = 2^{m}.$ $\quad \left[\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots = 1\right]$

(II) $\Rightarrow$ $2^m + 2^m + 2^m + \cdots + 2^m = 2^m \log m.$

Thus $S(m) = 2^m\, S(1) + 2^m \log m$ Assume $S(1) = $ constant

$S(m) = O(2^m \log m.)$

$T(n) = 2^{\log n} \cdot \log \log n$

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

Proof by induction

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

$$T(n) = O(n \log\log n)$$
$$\leq c \cdot n \log\log n$$

Assume true for some $m < n$

$m = n^{1/2} \rightarrow n = m^2$ [1 M]

Hypothesis

$\hookrightarrow T(\sqrt{n}) \leq c \cdot n^{1/2} \log\log n^{1/2}$

Induction

$\hookrightarrow T(n) = \sqrt{n}\, T(\sqrt{n}) + n$

$$= \sqrt{n}\left[c \cdot \sqrt{n} \log\log n^{1/2}\right] + n$$

$$= c \cdot n \cdot \log \frac{\log n}{2} + n$$

(2M)

$$= cn\left[\log\log n - \log 2\right] + n$$

$$= \underline{cn \log\log n} - cn + n$$

$$= cn \log\log n - (\underbrace{cn - n}_{\text{positive}})$$

[2 M]

$$c\cancel{n} - \cancel{n} \geq 0$$

$$c - 1 \geq 0$$

$$\boxed{c \geq 1} \longrightarrow \text{Proved.}$$

Q5 (A)

Given Algo -

for one man - find a woman whose height difference is minimum

Repeat for all men

a) This is Greedy Approach. ← [1 M]

b) No, it is not optimal. — [1 M] for Yes / No

Counter Example -

$m_1 = 20$ $m_2 = 10$
$w_1 = 11$ $w_2 = 30$ } — [1 M] for counter example

Our algo will make pair of m_1 & w_1.

since $20 - 11 = 9$, < $20 - 30 = 10$

Second pair will be then m_2 & w_2

Thus average diff = $20 - 11 = 9$; $30 - 10 = 20$ } $\frac{20+9}{2} = 14.5$

But if we make pairs like - $m_1 w_2$ & $m_2 w_1$

Then average diff = $30 - 20 = 10$; $11 - 10 = 1$ } $\frac{10+1}{2} = 5.5$

Hence not optimal.

c) $O(n^2)$ ← [1 M] for only $O(n^2)$

for 1^{st} man — 'n' comparison
for 2^{nd} man — 'n-1' comparison } [1 M] for how n^2

Thus $T(n) = 1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2} = O(n^2)$

Q5 (B) Knapsack

	A	B	C	D	E	F
w	10	20	30	40	50	60
P	60	100	120	150	200	220
P/w	6	5	4	3.75	4	3.66

Sort →

A	B	C	E	D	F
10	20	30	50	40	60
60	100	120	200	150	220

[1 M]

Item	Knapsack capacity	Profit
A	$100 - 10 = 90$	60
B	$90 - 20 = 70$	100
C	$70 - 30 = 40$	120
D	$40 - 50 \times \frac{40}{50} = 0$	$200 \times \frac{40}{50} = 160$

[3M]

Total profit $= 60 + 100 + 120 + 160$

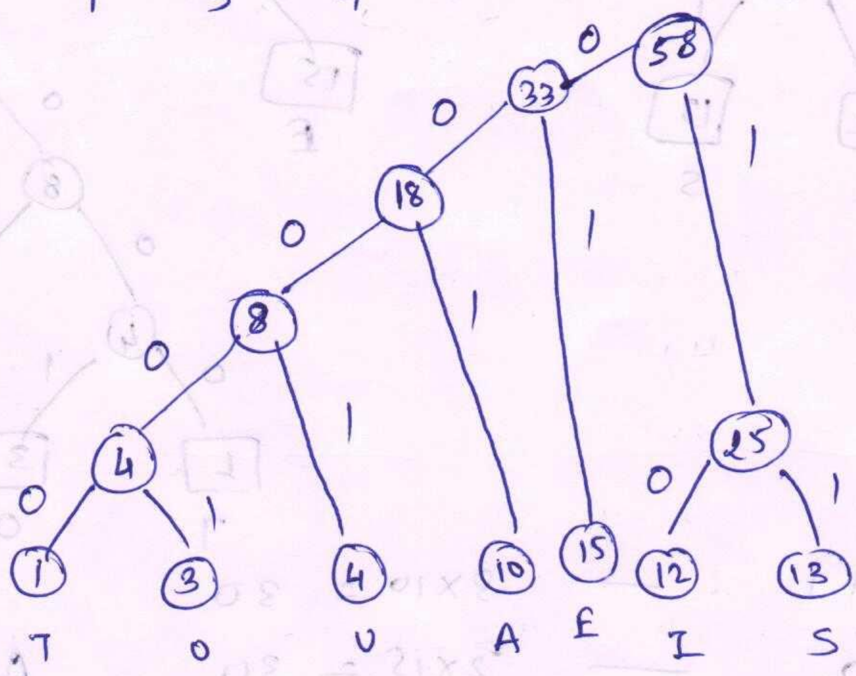
$= \underline{440}$

[1M]

[1M]	A	E	I	O	U	S	T	Actual Requirement
	10	15	12	3	4	13	1	$58 \times 8 = 464$ bits

Sort acc. to frequency -

T	O	U	A	I	S	E
1	3	4	10	12	13	15

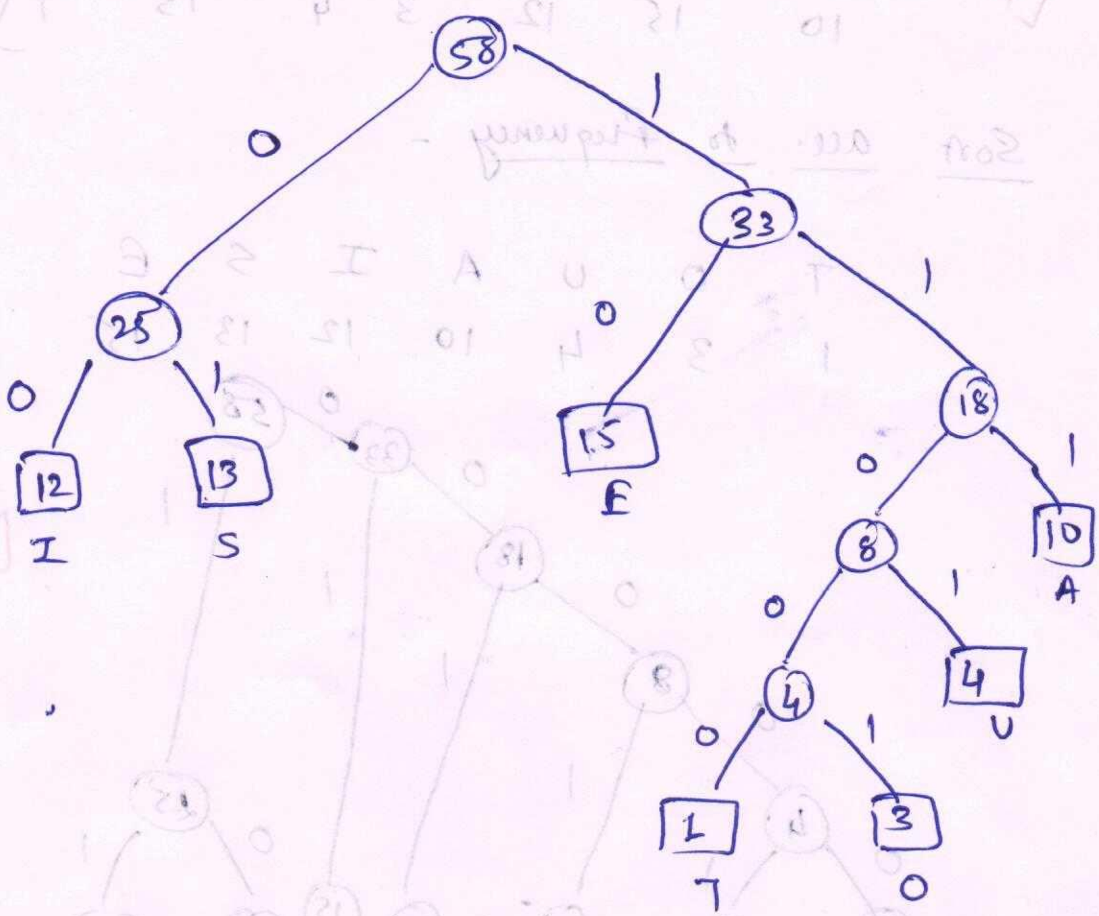


[2M] for correct tree

A	001	$3 \times 10 = 30$	
E	01	$2 \times 15 = 30$	
I	10	$2 \times 12 = 24$	
O	00001	$5 \times 3 = 15$	[1M]
U	0001	$4 \times 4 = 16$	
S	11	$2 \times 13 = 26$	
T	00000	$5 \times 1 = 5$	
$7 \times 8 = 56$	23	146	Total = 225 bits.

$\boxed{\text{Savings} = 51.5\%}$ ← [1M]

Alternate Possible Tree



A	111	—	3×10 = 30	
E	10	—	2×15 = 30	Actual Requirement
I	00	—	2× 12 = 24	
O	11001	—	5× 3 = 15	= 58×8
U	1101	—	4×4 = 16	= 464
S	01	—	2×13 = 26	
T	11000	—	5×1 = 5	

$7\times8=\underline{\underline{56}}$ + $\underline{\underline{23}}$ + $\underline{\underline{146}}$ = $\boxed{225 \text{ bits}}$

$$\text{Saving} = \frac{464-225}{464}\times 100 = \underline{\underline{51.5\%}}$$