Name - Sradha Kedia
Roll no. - 20234757053

4 (b) $n = 4$, $w_1 = 1$, $w_2 = 3$, $w_3 = 3$, $w_4 = 5$ and
$W = 9$

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 0 | 1 | 1 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 3 | 0 | 1 | 1 | 3 | 4 | 4 | 6 | 7 | 7 | 7 |
| 2 | 0 | 1 | 1 | 3 | 4 | 4 | 4 | 4 | 4 | 4 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

```
M[0][W] = 0;
for i = 1 to 4
 {
   for w = 0 to 9
    {
      if (w < wi)
          opt(i,w) = opt(i-1, w)
      else
          opt(i,w) = max( opt(i-1,w), wi + opt(i-1, w-wi))
    }
 }
```

I - $i = 1$, $w_1 = 1$
$w = 0$, if $\rightarrow$ $0 < 1$ true; $opt(1,0) = opt(0,0)$
$w = 1$, if $\rightarrow$ $1 < 1$ false; $opt(1,1) =$ ~~opt~~ $\max(opt(0,1), 1 + opt(0,0))$
$= \max(0, 1) = 1$
$w = 2$, if $\rightarrow$ $2 < 1$ false, $opt(1,2) = \max(opt(0,2), 1 + opt(0,1))$
$= \max(0, 1+0) = 1$

$w = 3$, if $\to 3 < 1$, false $\quad opt(1, 3) = max(opt(0,3), 1 + opt(0,2))$

$= max(0, 1+0) = 1$

$w = 4$, if $4 < 1$, false $\quad opt(1,4) = max(opt(0,4), 1 + opt(0,3))$

$= max(0, 1) = 1$

false for all $w = 5$ to $9$; too

$w = 5$, $\quad opt(1,5) = max(opt(0,5), 1 + opt(0,4))$

$= max(0, 1) = 1$

for $w = 6$, $w = 7$, $w = 8$, $w = 9$, similarly $opt(1, w) = 1$.

II. $i = 2$, $w_2 = 3$

$w = 0$, if $(0 < 3)$ true, $\quad opt(2, 0) = opt(1, 0) = 0$

$w = 1$, " $(1 < 3)$ ", $\quad opt(2, 1) = opt(1, 1) = 1$

$w = 2$, " $(2 < 3)$ ", $\quad opt(2, 2) = opt(1, 2) = 1$

$w = 3$, " $(3 < 3)$ false, $\quad opt(2, 3) = max(opt(1,3), 3 + opt(1, 0))$

$= max(1, 3 + 0) = 3$

$w = 4$, to $9$, all false, $\therefore$ $opt(2, 4) = max(opt(1,4), 3 + opt(1, 1))$

$= max(1, 3 + 1)$

$= 4$

$opt(2, 5) = max(opt(1,5), 3 + opt(1, 2))$

$= max(1, 3 + 1) = 4$

$opt(2, 6) = max(opt(1, 6), 3 + opt(1, 3))$

$= max(1, 3 + 1) = 4$

$opt(2, 7) = max(opt(1, 7), 3 + opt(1, 4)) = max(1, 3+1) = 4$

$opt(2, 8) = max(opt(1, 8), 3 + opt(1, 5)) = max(1, 4) = 4$

$opt(2, 9) = max(opt(1, 9), 3 + opt(1, 6)) = max(1, 4)$

$= 4$

III $i=3, \ w_3=3$

$w=0$, if $(0<3)$ true, $opt(3,0) = opt(2,0) = 0$

$w=1$, if $(1<3)$ true, $opt(3,1) = opt(2,1) = 1$

$w=2$ true, $opt(3,2) = opt(2,2) = 1$

$w=3$ to $w=9$ false, $opt(3,3) = max(opt(2,3), 3+opt(2,$
$= max(3, 3+0)$
$= 3$

$opt(3,4) = max(opt(2,4), 3+opt(2,1))$
$= max(4, 3+1) = 4$

$opt(3,5) = max(opt(2,5), 3+opt(2,2))$
$= max(4, 3+1) = 4$

$opt(3,6) = max(opt(2,6), 3+opt(2,3))$
$= max(4, 3+3) = 6$

$\boxed{opt(3,6) = 6}$ (Ans)

4

$opt(3,7) = max(opt(2,7), 3+opt(2,4))$
$= max(4, 3+4) = 7$

$opt(3,8) = max(opt(2,8), 3+opt(2,5))$
$= max(4, 3+4) = 7$

$opt(3,9) = max(opt(2,9), 3+opt(2,6))$
$= max(4 \cancel{4}, 3+4) = 7$

IV, $i=4, \ w_4=5$

$w=0$, if $(0<5)$ $opt(4,0) = opt(3,0) = 0$

$\vdots$ $\vdots$ $opt(4,1) = opt(3,1) = 1$

$w=4$ if $(4<5)$ $opt(4,2) = opt(3,2) = 1$

$opt(4,3) = opt(3,3) = 3$

$opt(4,4) = opt(3,4) = 4$

$w=5$ if $(5<5)$ false, $opt(4,5) = max(opt(3,5), 5+opt(3,0))$
$= max(4, 5+0) = 5$

$w=6$, if $(6\leq 5)$ false ; $opt(4,6) = max(opt(3,6), 5+opt(3,1))$
$w=7,8,$
$w=9$ false
$= max(6, 5+1) = 6$

$opt(4,7) = max(opt(3,7), 5+opt(3,2))$
$= max(7, 5+1) = 7$

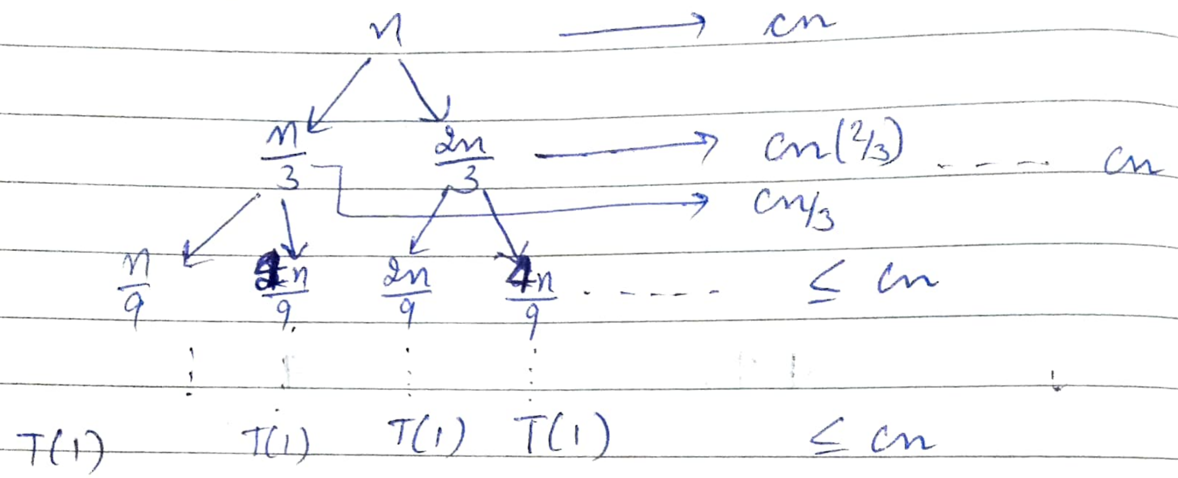
$opt(4,8) = max(opt(3,8), 5+opt(3,3))$
$= max(7, 5+3) = 8$

$opt(4,9) = max(opt(3,9), 5+opt(3,4))$
$= max(7, 5+4) = 9$

4 (a) We assume that activities are sorted in increasing order of their finishing time. Now, the Recursive activity selector is not that efficient from Greedy activity selector.
Greedy activity selector is an iterative approach which is greedy in nature giving O(n) complexity.
But in recursive Activity selector, the algorithm can be O(n logn) in worst case.
Thus, they are different algorithms where GAS is improved version (greedy version) of RAS.

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

Recursive tree :




to remember, $\frac{n}{3}$ left subtree will be smaller than $\frac{2n}{3}$ and the height of the tree is determined [by] rightmost part of the tree.

After $k^{th}$ iteration,

$$T\left(\left(\frac{2}{3}\right)^k . n\right) = 1$$

$$\left(\frac{2}{3}\right)^k . n = 1$$

$$n = \left(\frac{3}{2}\right)^k \Rightarrow \log n = k \log(3/2)$$

$$\therefore k = \log_{\frac{3}{2}} n$$

total time $\leq cn . \left(\log_{3/2} n\right)$

$$T(n) = O\left(n \log_{\frac{3}{2}} n\right) \text{ (Ans)}$$

(b) The suitable algorithm that goes with the given assumption is bucket sorting.

Best case when array is uniformly distributed and buckets have only one elements in them thus converting it to constant time thus overall all linear sorting.

eg - 0.11, 0.23, 0.31, 0.46, 0.51, 0.64, 0.71, 0.89, 0.92

here, bucket's sizes are 1.

| | |
|---|---|
| 0 | → ~~0.~~ |
| 1 | → 0.11 |
| 2 | → 0.23 |
| 3 | → 0.31 |
| 4 | → 0.46 |
| 5 | → 0.51 |
| 6 | → 0.64 |
| 7 | → 0.71 |
| 8 | → 0.89 |
| 9 | → 0.92 |

Worst case, when not uniformly distributed

eg - 0.19, 0.18, 0.16, 0.14, 0.13, 0.12...

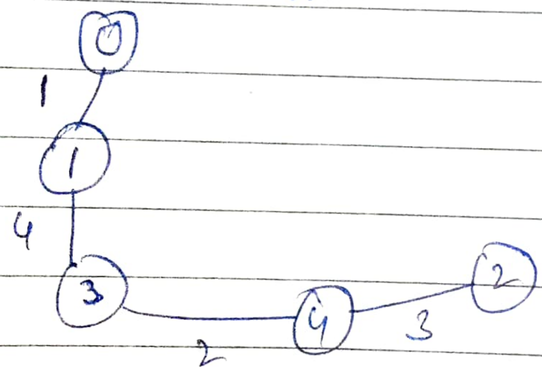
here all the elements goes in one bucket then here bucket's sorting will also contribute to overall time complexity. i.e. $O(n^2)$.

3 (a)

$$
W = \begin{array}{c|ccccc|}
 & 0 & 1 & 2 & 3 & 4 \\
0 & 0 & 1^{\checkmark} & 8 & 1 & 4 \\
1 & 1^{*} & 0 & 12 & 4^{\checkmark} & 9 \\
2 & 8 & 12 & 0 & 7 & 3 \\
3 & 1^{*} & 4^{*} & 7 & 0 & 2^{\checkmark} \\
4 & 4^{*} & 9^{*} & 3 & 2^{*} & 0
\end{array}
$$

root is 0

we start from 0, & see 1 is minimum, we choose 1 node.




It will not generate unique MST, as we choose second node as 1.

(b) $a = 786$
$b = 120$

for $a$ and $b$ length be the power of 2
$a = 0786$
$b = 0120$

$n = 4$

$a = a_1 * 10^{n/2} + a_2$ $\quad$ $I \rightarrow a \times b = \underset{A}{a_1 b_1} 10^n + (\underset{B}{a_1 b_2} + \underset{C}{a_2 b_1}) 10^{n/2} + \underset{D}{a_2 b_2}$

$b = b_1 * 10^{n/2} + b_2$

Step 1: $a = 07 \times 10^{4/2} + 86$
$b = 01 \times 10^2 + 20$
$a_1 = 07$, $b_1 = 01$, $a_2 = 86$, $b_2 = 20$

$A =$ Multiply $(a_1, b_1) = 7$ — (i)
$B =$ Multiply $(a_1, b_2) = 7 \times 20 = 140$ — (ii)
$C =$ Multiply $(a_2, b_1) = 86$ — (iii)
$D =$ Multiply $(a_2, b_2) = 1720$ — (iv)

So, $786 \times 120 = A \times 10^n + (B + C) 10^{n/2} + D$
Divide again (i), (ii), (iii) & (iv).
for $a_1, b_1 = 07 \times 01$
(i) $\rightarrow$ $A_1 =$ Multiply $(0, 0) = 0$
$B =$ " $(0, 1) = 0$
$C =$ " $(7, 0) = 0$
$D =$ " $(7, 1) = 7$
from the above, we get multiply$(a_1, b_1) \Rightarrow$ (i)
$= 7$.

(iii) → $86 \times 01$ $(a_2, b_1)$

$A = \text{Multiply}(8 \times 0) = 0$

$B = \text{"} \quad (8, 1) = 8$

$C = \text{"} \quad (6, 0) = 0$

$D = \text{"} \quad (6, 1) = 6$

So, 86 for $\boxed{\text{(iii) B}}$

0
0
86
---
86

(apply same formula as I)

(ii) → $07 \times 20$ $(a_1, b_2)$

$A = \text{Multiply}(0 \times 2) = 0$

$B = \text{Multiply}(0, 0) = 0$

$C = \text{"} \quad (7, 2) = 14$

$D = \text{"} \quad (7, 0) = 0$

So, 140

for B of (ii) is 140

(iv) $86 \times 20$

$A = \text{multiply}(8, 2) = 16$

$B = \text{"} \quad (8, 0) = 0$

$C = \text{"} \quad (6, 2) = 12$

$D = \text{"} \quad (6, 0) = 0$

So, 172

D for (iv)

now, $7 \times 10^4 + (86 + 140)\, 10^2 + 1720$

$= 7000 + 22600 + 1720$

$= 94320$ (Ans)

Date
Page

0 1 2 3 4 5 6 7 8 9
5 (a) Array 20, 47, 15, 8, 9, 4, 40, 30, 12, 17

step 1: pivot = a [ mid ]
= a[4]
= 9

step2: partition the subarray
20 47 15 08 9 4 40 30 12 17
↑ ↑

→ swap 20 and 4
4 47 15 08 9 20 40 30 12 17
↑ ↑

step3: swap 47, 08
4, 8, 15, 47, 9, 20, 40, 30, 12, 7

moving left and right bound now will cross them so, our bound is at 2 and all elements left from this are smaller than pivot and on right all elements are larger ~~than~~ now move pivot to its final partition we get

[ [4,8], 9 [47, 17, 20, 40, 30, 12, 15] ]

step 4) now, we call quicksort on left sublist [4, 8], pivot will be 4.
move pivot to the end we get,
8, 4
our bound is at 0 in the sublist & left and right bound crossed, now we will move pivot to its

original location, we get 4, 8.

Step 6.) we again call QS on left sublist [8]
so, as only one element is there, we ~~got~~ will
return from here.

4, 8, 9, 47, 17, 20, 40, 30, 12, 15

L'is sorted and now we will similarly continue
for right subarray & get the answers.

5. (b) Fibonacci series 0, 1, 1, 2, 3, 5, .... is a
sorted array so if any element is
repeating more than $\frac{n}{2}$ times, then for
sure it will be present at anyone
of neighbourhood indices $i-1$, $i+1$.

so, now once we get the middle element
we will count its occurence in $(\log n)$
complexity by BS (binary search)

and the recurrence relation is $T(n) = 2T(n/2) + 1$
for fibonacci's searching (BS algo)

Date
Page

1. a) the first loop has $O(n)$ complexity.

```
for ( p=1; p<n; ++p) — O(n)
{
  s=0
  for(q=n; q>1; q=q/2) — O(logn)
    ++s;
  for (r=1; r<n; r=r*2) — O(logn)
    ++t;
}
```

$2n\log n$

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

1. b) [10, 20, 30, 40, 50]

C1 : INS = 4
MS = 7

C2 : INS = 9
MS = 17