Part A

1. The algorithms like merge sort, quick sort and binary search are based on
   a. Greedy algorithm
   b. Divide and conquer algorithm
   c. Dynamic programming approach
   d. Hash table

   Answer: Divide and Conquer algorithm

2. The steps in the divide and conquer process that takes a recursive approach is said to be
   a. Sort
   b. Conquer
   c. Divide
   d. Both b and c

   Answer: Divide

3. The algorithm which has time complexity of O(n log n) for best, worst and average cases is
   a. Merge sort
   b. Quick sort
   c. Insertion sort
   d. Selection sort

   Answer: Merge sort

4. The recurrence relation for finding maximum and minimum elements from an array using divide and conquer technique is
   a. 2T(n/2) + n
   b. 4T(n/2) + n^2
   c. 2T(n/2) + 2
   d. 3T(n/2) + 1

   Answer: 2T(n/2) +2

5. Which approach is based on computing the distance between each pair of distinct points and finding a pair with the smallest distance?
   a. Brute force
   b. Greedy approach
   c. Divide and conquer
   d. Branch and bound

   Answer: Brute force

6. How many bits are needed for standard encoding if the size of the character set is X?
   a. log X
   b. X+1
   c. 2X
   d. X^2

   Answer: log X

7. Which of the following is a variable length coding method?
   a. ASCII code

b. EBCDIC code
c. Grey code
d. Huffman code

Answer: Huffman code

8. Consider the two matrices P and Q which are 10 x 20 and 20 x 30 matrices respectively. What is the number of multiplications required to multiply the two matrices?
a) 10*20
b) 20*30
c) 10*30
d) 10*20*30
Answer: 10*20*30
9. If an optimal solution can be created for a problem by constructing optimal solutions for its subproblems, the problem possesses ________ property.
a. Overlapping subproblems
b. Optimal substructure
c. Memoization
d. Greedy

Answer: Optimal substructure

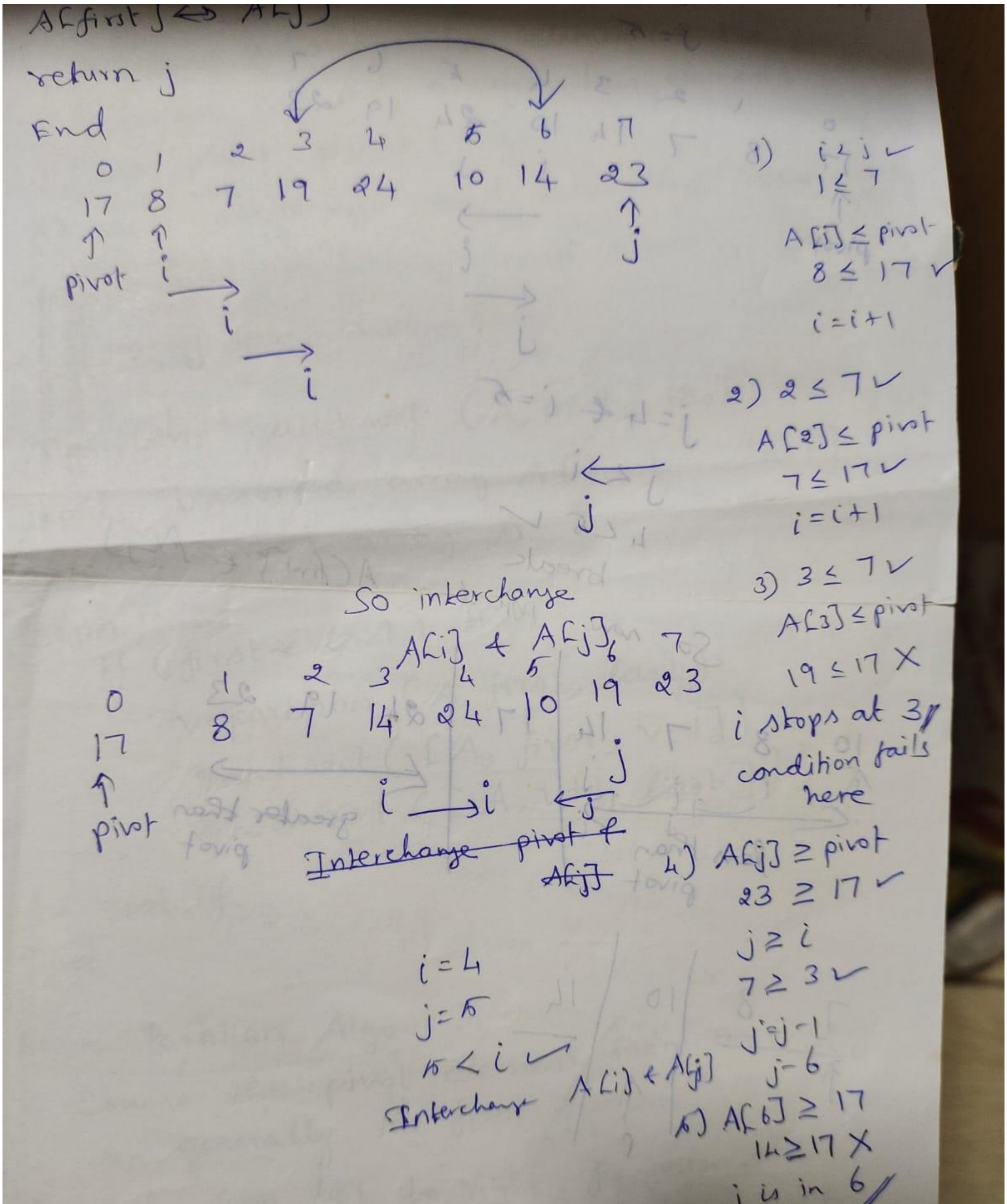
10. Prim's algorithm is a ______
a) Divide and conquer algorithm
b) Greedy algorithm
c) Dynamic Programming
d) Approximation algorithm

Answer: Greedy algorithm

Part B

Answer any four

1. Apply quick sort on the following sequence 17, 8, 7, 19, 24, 10, 14, 23 and also analyse the time complexity. Perform the dry run for the given example.
Sorting the elements + Dry run - 7 marks
Time complexity – 3 marks



A[first] ⟺ A[j]

return j

End

0	1	2	3	4	5	6	7
17	8	7	19	24	10	14	23
↑ pivot	↑ i						↑ j

i → i → i

j ←

1) i < j ✓
1 ≤ 7

A[i] ≤ pivot
8 ≤ 17 ✓

i = i + 1

2) 2 ≤ 7 ✓
A[2] ≤ pivot
7 ≤ 17 ✓
i = i + 1

3) 3 ≤ 7 ✓
A[3] ≤ pivot
19 ≤ 17 ✗

i stops at 3 //
condition fails here

So interchange
A[i] & A[j]

0	1	2	3	4	5	6	7
17	8	7	14	24	10	19	23
↑ pivot			i	→ i	← j	j	

~~Interchange pivot & A[j]~~

4) A[j] ≥ pivot
23 ≥ 17 ✓

j ≥ i
7 ≥ 3 ✓

j = j − 1
j = 6

5) A[6] ≥ 17
14 ≥ 17 ✗
j is in 6 //

j < i 6 < 3 ✗
NO

i = 4
j = 5
5 < i ✓

Interchange A[i] & A[j]

0 1 2 3 4 5 6

17 8 7 14 10 24 19 23

i j

↑ pivot

$i = 4$

$j = 5$

0 1 2 3 4 5 6 7

17 8 7 14 10 24 19 23

i j

↑ pivot

→ i

← j

$j = 4$ & $i = 5$

$j < i$

$4 < 5$ ✓

break

interchange A[first] & A[j]

So now

10 8 7 14 | 17 | 24 19 23

i i j

less than pivot

greater than pivot

7 8 | 10 | 14

↑

r

$$T(n) = \begin{cases} c & \text{if } n = 1 \\ 8T\left(\frac{n}{2}\right) + an^2 & \text{if } n \geq 2 . \end{cases}$$

Quick sort time complexity analysis

The best care occurs when the pivot element is placed exactly in the middle of the array and partitions the list evenly.

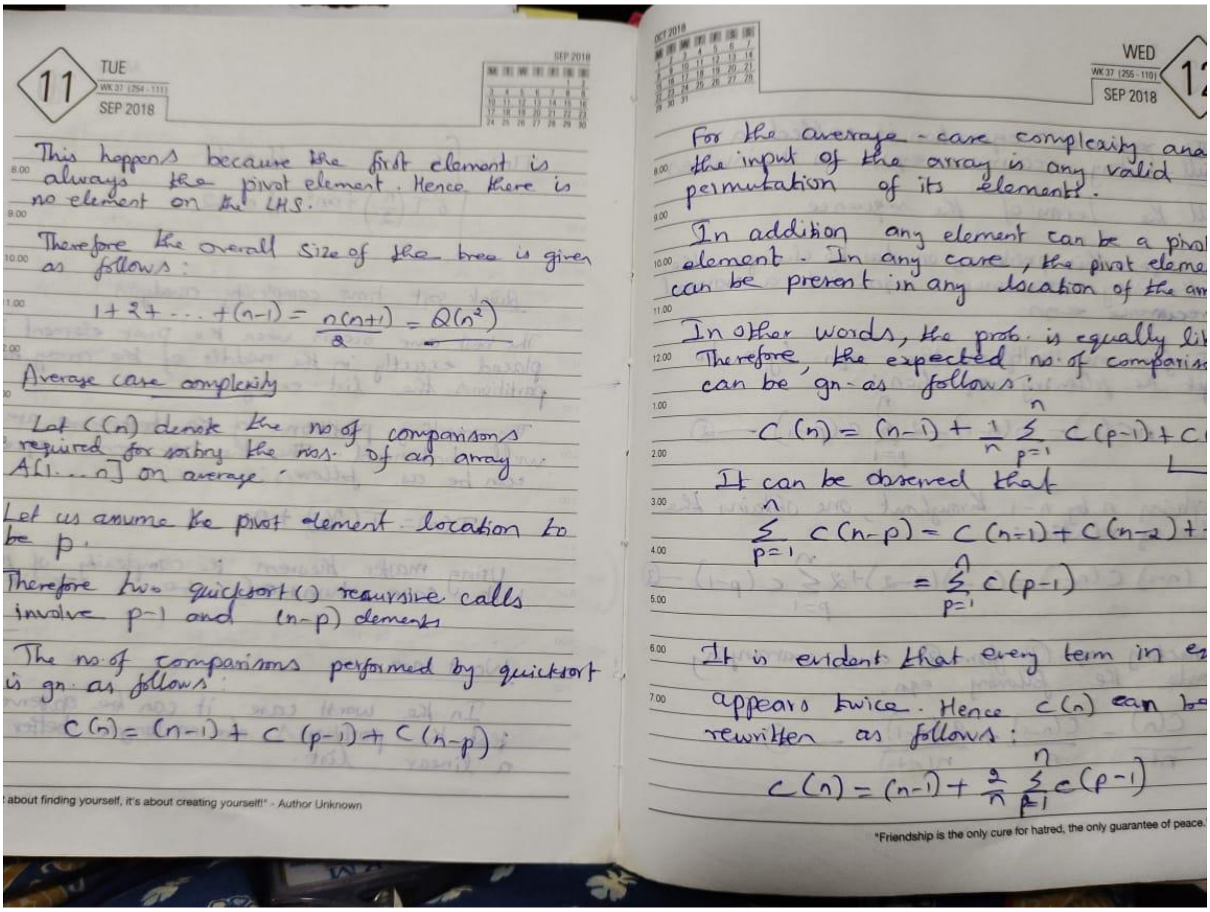
The resulting partitions of the best care are well balanced. Thus the recurrence equation can be as follows:

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

Using master theorem the complexity of the best care turns out to be $T(n) \in \Theta(nl$ $n)$

Worst care

In the worst care, it can be observed that the partitions are no longer better than a linear list.



11 TUE
WK 37 (254 - 111)
SEP 2018

SEP 2018

M	T	W	T	F	S	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

8.00 This happens because the first element is always the pivot element. Hence there is no element on the LHS.

9.00

10.00 Therefore the overall size of the tree is given as follows:

11.00

$$1+2+\dots+(n-1) = \frac{n(n+1)}{2} = \underline{Q(n^2)}$$

2.00

Average case complexity

Let $C(n)$ denote the no of comparisons required for sorting the nos. of an array $A[1\dots n]$ on average.

Let us assume the pivot element location to be p.

Therefore two quicksort() recursive calls involve p-1 and (n-p) elements

The no. of comparisons performed by quicksort is gn. as follows:

$$C(n) = (n-1) + C(p-1) + C(n-p);$$

about finding yourself, it's about creating yourself!" - Author Unknown

OCT 2018

M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

WED
WK 37 (255 - 110)
SEP 2018
12

For the average - case complexity ana
8.00 the input of the array is any valid permutation of its elements.

9.00

In addition any element can be a pivo
10.00 element. In any case, the pivot eleme
can be present in any location of the arr

11.00

In other words, the prob. is equally lik
12.00 Therefore, the expected no. of comparis
can be gn. as follows:

1.00

$$C(n) = (n-1) + \frac{1}{n}\sum_{p=1}^{n} C(p-1) + C$$

2.00

It can be observed that

3.00

$$\sum_{p=1}^{n} C(n-p) = C(n-1) + C(n-2) + $$

4.00

$$= \sum_{p=1}^{n} C(p-1)$$

5.00

6.00 It is evident that every term in e

7.00 appears twice. Hence $C(n)$ can be rewritten as follows:

$$C(n) = (n-1) + \frac{2}{n}\sum_{p=1}^{n} C(p-1)$$

"Friendship is the only cure for hatred, the only guarantee of peace.

This recurrence equation is called a
8.00 full-history recurrence eqn. as it involves
9.00 all the terms of the sequence.

10.00 To solve this problem, one has to change the full-history recurrence eqn. into a normal
11.00 recurrence eqn.

12.00 Therefore one multiplies the eqn. by n to get the following equation:
1.00

$$nC(n) = n(n-1) + 2\sum_{p=1}^{n} C(p-1) \quad — ②$$

2.00

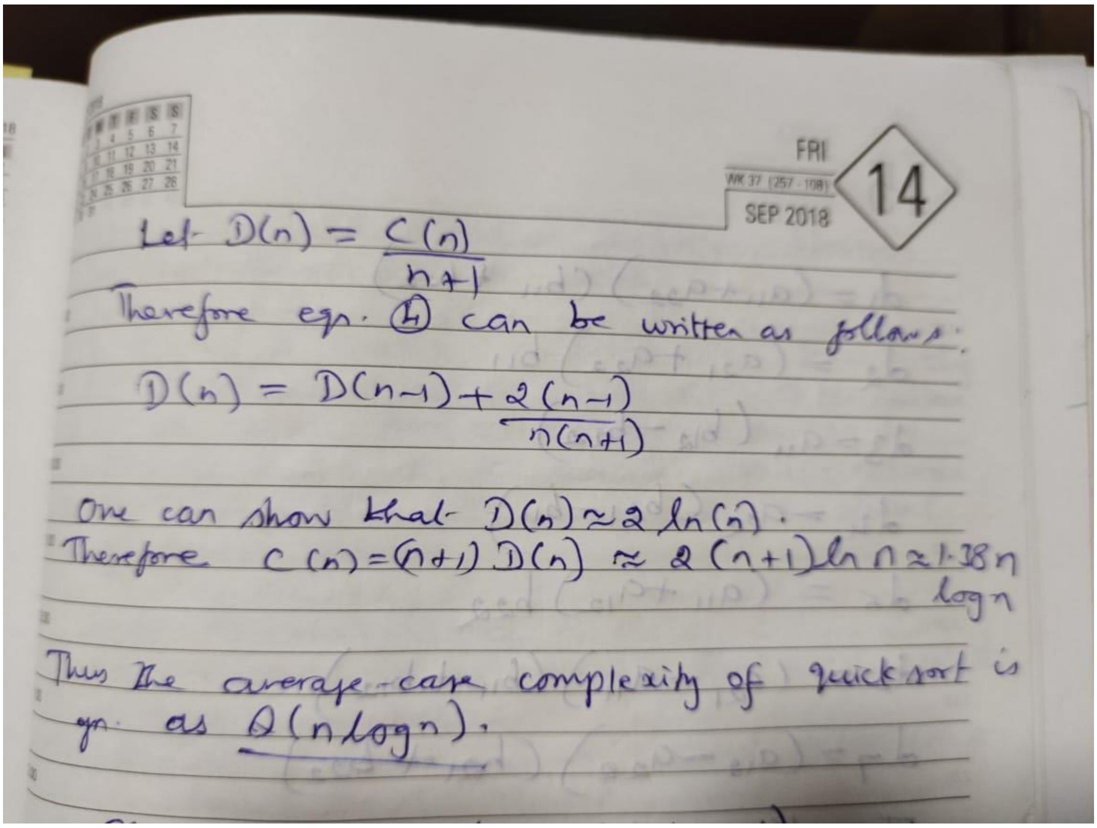
3.00 Replacing n by n-1 throughout, one obtains the following relation:
.00

$$(n-1)\,C(n-1) = (n-1)(n-2) + 2\sum_{p=1}^{n} c(p-1) \quad — ③$$

00

00 Subtracting eqn. ③ from ② and rearranging provide the following eqn.:
0

$$\frac{C(n)}{n+1} = \frac{C(n-1)}{n} + \frac{2(n-1)}{n(n+1)} \quad — ④$$



FRI

WK 37 (257-108)

SEP 2018

14

Let $D(n) = \frac{C(n)}{n+1}$

Therefore eqn. ④ can be written as follows:

$$D(n) = D(n-1) + \frac{2(n-1)}{n(n+1)}$$

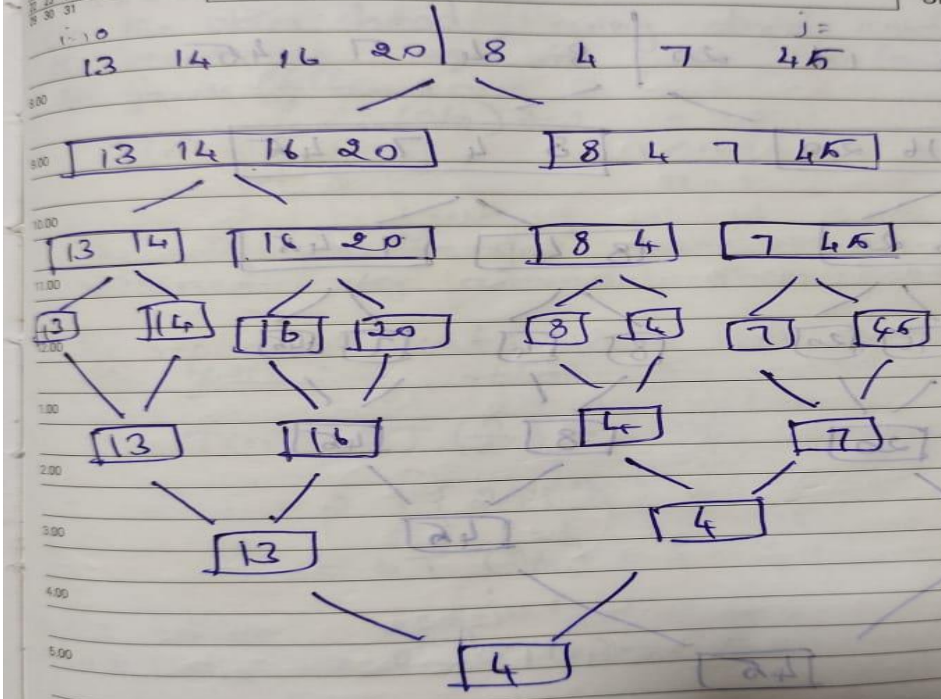
One can show that $D(n) \approx 2\ln(n)$.

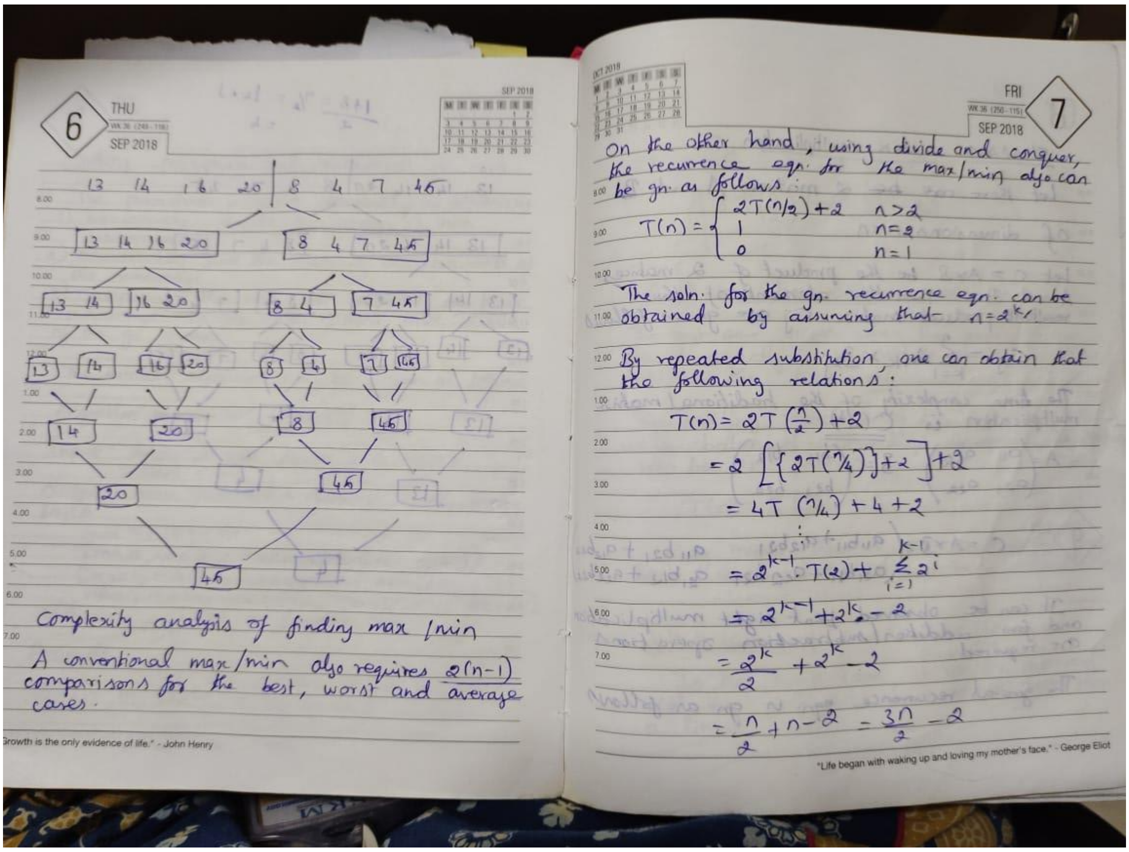
Therefore $C(n) = (n+1)D(n) \approx 2(n+1)\ln n \approx 1.38 n \log n$

Thus the average-case complexity of quick sort is gn. as $\underline{\Theta(n\log n)}$.

2. Find the maximum and minimum element from the given array A = {13, 14, 16, 20, 8, 4, 7, 45} using divide and conquer technique and analyse the time complexity also. Perform the dry run for the given example.
   Finding maximum and minimum + Dry run – 7 marks
   Time complexity – 3 marks

$\frac{1+8}{2} = 9/2 = \lfloor 4.5 \rfloor$
$= 4$




6
THU
SEP 2018
13 14 16 20 8 4 7 45
13 14 16 20
8 4 7 45
13 14
16 20
8 4
7 45
13
14
16
20
8
4
7
45
14
20
8
45
20
45
45
"Growth is the only evidence of life." - John Henry
FRI
7
SEP 2018
"Life began with waking up and loving my mother's face." - George Eliot

Complexity analysis of finding max /min

A conventional max/min algo requires $\underline{2(n-1)}$ comparisons for the best, worst and average cases.

On the other hand, using divide and conquer, the recurrence eqn. for the max/min algo can be gn. as follows

$$T(n) = \begin{cases} 2T(n/2)+2 & n>2 \\ 1 & n=2 \\ 0 & n=1 \end{cases}$$

The soln. for the gn. recurrence eqn. can be obtained by assuming that $n=2^k$.

By repeated substitution, one can obtain that the following relations:

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

$$= 2\left[\{2T(n/4)\}+2\right]+2$$

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

$$\vdots$$

$$= 2^{k-1}\cdot T(2) + \sum_{i=1}^{k-1} 2^i$$

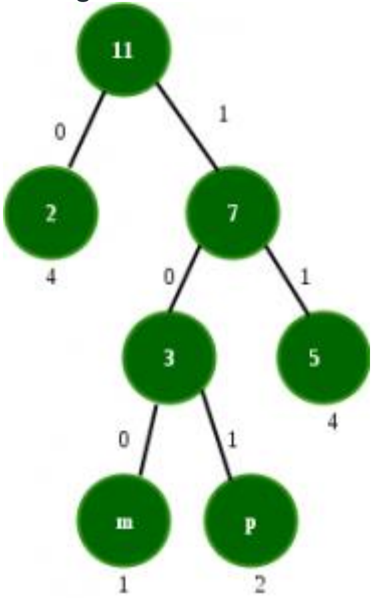
$$= 2^{k-1} + 2^k - 2$$

$$= \frac{2^k}{2} + 2^k - 2$$

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

3. Compute the frequency table for the characters "mississippi" and then find the codes for each character by constructing Huffman tree by applying greedy technique.
Frequency table – 3 marks
Huffman tree + codes for each character – 7 marks

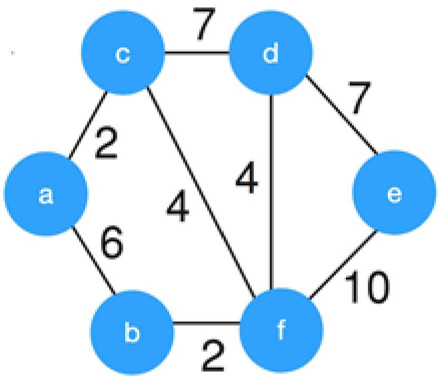
The generated Huffman tree is:



Following are the codes:

Character	Frequency	Code	Code Length
m	1	100	3
p	2	101	3
s	4	11	2
i	4	0	1

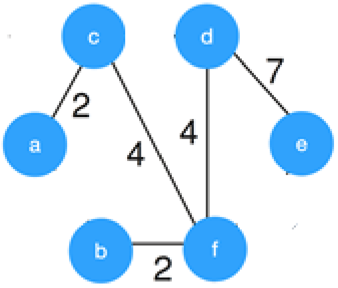
4. Consider the given graph.



Construct minimum spanning tree using Kruskals algorithm.

Step by step construction of the MST – 10 marks

Explanation: Kruskal's algorithm constructs the minimum spanning tree by constructing by adding the edges to spanning tree one-one by one. The MST for the given graph is,



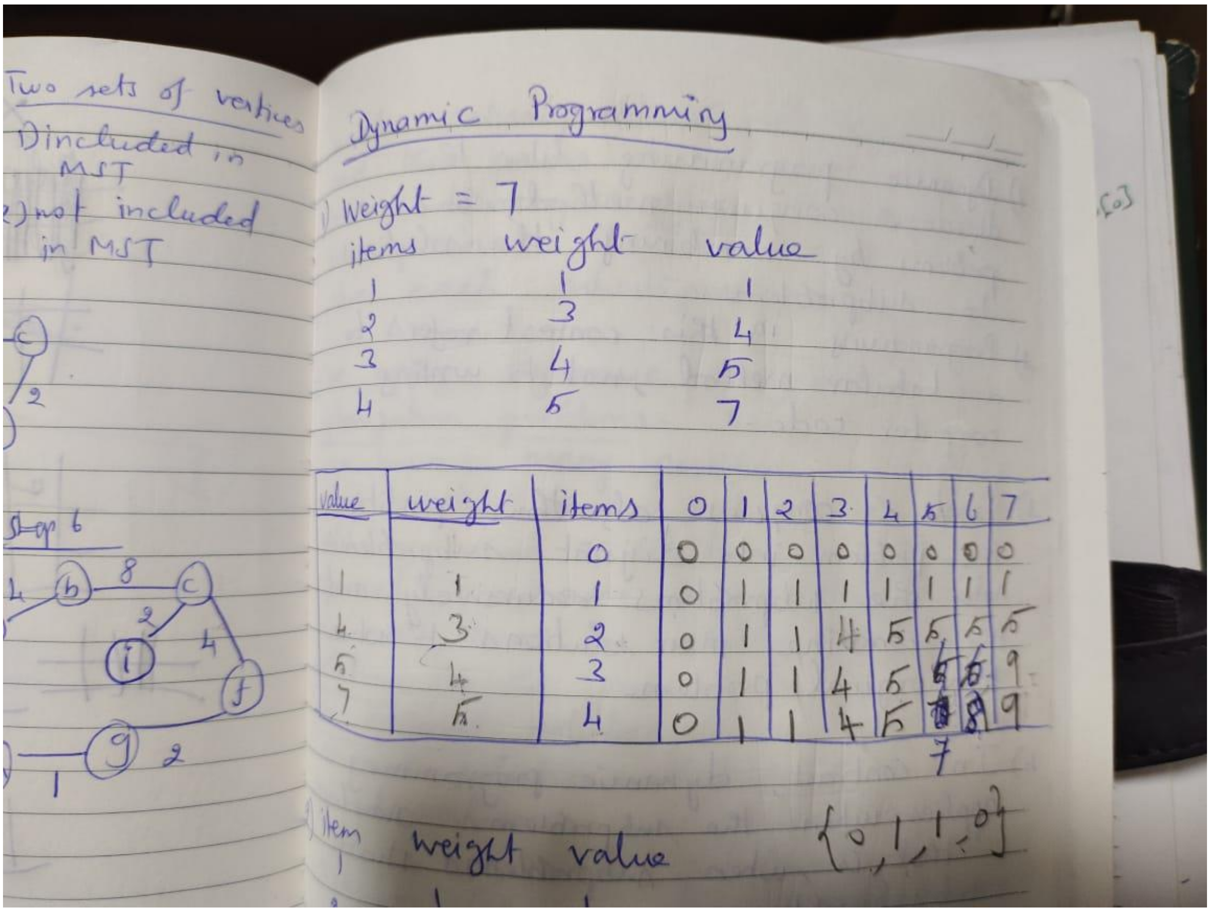
So, the weight of the MST is 19

5. Let there be a Knapsack with capacity W=7 Kg. Let there be 4 items with whose profit and weight are given in the table. Find the optimal order for loading the items in the given Knapsack using dynamic programming approach.

Items	Weight (Kg)	Profit
1	1	1
2	3	4
3	4	5
4	5	7

Table – 8 marks

solution set – 2 marks



Two sets of vertices

1) included in MST

2) not included in MST

Step 6

c
2
4
b
8
c
2
i
4
f
g
2
1

Dynamic Programming

1) Weight = 7

items	weight	value
1	1	1
2	3	4
3	4	5
4	5	7

value	weight	items	0	1	2	3	4	5	6	7
		0	0	0	0	0	0	0	0	0
1	1	1	0	1	1	1	1	1	1	1
4	3	2	0	1	1	4	5	5	5	5
5	4	3	0	1	1	4	5	6	6	9
7	5	4	0	1	1	4	5	[illegible]	8	9

7

{0, 1, 1, 0}

2) item weight value

1