Name : Harishchandrasing Sahabsing Kushwah
Roll No : 246245
Sub : DAC Assignment

Q.1
=>

n	$n^2 \log(n)$	$n \log(n)$	$10.n^2$	$\log(n)$	2^n	$n^3 \log(n)$
2	4	2	40	1	4	8
4	32	8	160	2	16	128
8	192	24	640	3	256	1536
16	1024	64	2560	4	65536	16384
32	5120	160	10240	5	4.2949×10^9	163840
2048	46137344	22528	4194304	11	undefind	9.44×10^{10}

From above table we can observe that growth of the functions in ascending order

$\log(n) < n\log(n) < 10n^2 < n^2\log(n) < n^3\log(n) < 2^n$

Q. 2

ROLL NO = 246245

x = ROLL-NO % 4

x = 246245 % 4 = 1

A = 1, 2, 4, 1, 2, 1, 1

k = 4 --- maximum element

A	1	2	4	1	2	1	1
	1	2	3	4	5	6	7

--- n = 7

i	4	2	0	1
	1	2	3	4

---- counting array

i	A[i]	C[A[i]]
1	1	0+1 = 1
2	2	0+1 = 1
3	4	0+1 = 1
4	1	1+1 = 2
5	2	1+1 = 2
6	1	2+1 = 3
7	1	3+1 = 4

Pree sum of counting array

C	4	6	6	7

B	1	1	1	1	2	2	4
	1	2	3	4	5	6	7

... external array to store sorted elements

i	A[i]	C[A[i]]	B[C[A[i]]]
7	1	4	B[4] = 1
6	1	3	B[3] = 1
5	2	6	B[6] = 2
4	1	2	B[2] = 1
3	4	7	B[7] = 4
2	2	5	B[5] = 2
1	1	1	B[1] = 1

Q.3

$\Rightarrow 2n^2+3n+1 = O(n^2)$

n	$f(n) = 2n^2+3n+1$	$c \cdot O(n^2) = 3 \cdot n^2$
0	1	0
1	6	3
2	15	12
3	28	27
→ 4	45	48
5	66	75

$2n^2+3n+1 = O(n^2) \quad \forall \quad n \geq 4 \quad \& \quad c = 3$

Q 4

=>

A :	1	2	1	0
	0	1	2	3

pivot = 1 low = 0 high = 3 is = high = 3

i = 3

is = 3 if (arr[3] > pivot) { → false

0 > 1

}

i = 2

is = 3 if (arr[2] > pivot) { → false

1 > 1

}

i = 1

is = 3

if (arr[1] > pivot) { → True

2 > 1

swap (arr[1], arr[is])

is--

}

A :	1	0	1	2
	0	1	2	3

i = 0

is = 2 if (arr[0] > pivot) { → false

1 > 1

}

swap(arr[low], arr[ik])

A :

1	0	1	2
0	1	2	3

New pivot index = ik for next iteration
ik = 2 ... from previous iteration

A' :

1	0
0	1

A'' :

2
3

low = 0 high = 1
here low < high
∴ further algorithm work.

low = 3 high = 3
low < high → False

pivot = 1
ik = high = 1

i = 1
ik = 1 if(arr[i] > pivot) { → false
0 > 1
}

j = 0
ik = 1 if(arr[j] > pivot) { → false
1 > 1
}

swap(arr[low], arr[ik])

A' :

0	1
0	1

New pivot index = k = 1

A''' =	0		
	0		

low = 0 high = 0

low < high → false

Now final array after sorting

A :	0	1	1	2

In above array the position of similar elements remains same therefore quick sort is stable sort algorithm.

2.5

=> n = 3

m = 13

IN = (9, 6, 5)

P = (15, 18, 20)

To find an optimal solution find the profit is to weight (P/w) ratio and then apply the greedy approach

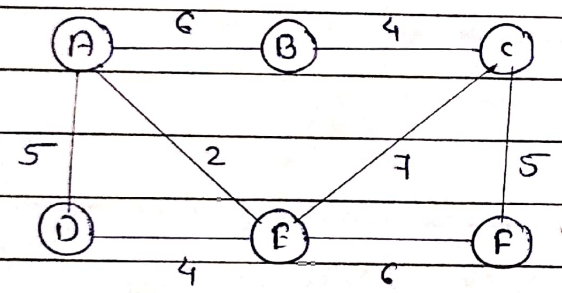
P/w = (15/9 , 18/6 , 20/5)

= (1.66 , 3 , 4)

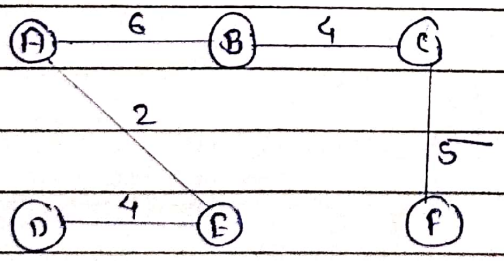
select maximum P/W

item	profit	weight
I_3	20	13-5 = 8
I_2	6	8-6 = 2
I_1	(2/6) × 15 = 3.33	2-2 = 0
Total profit	29.33	

2. c)



a) Kruskal's algorithm

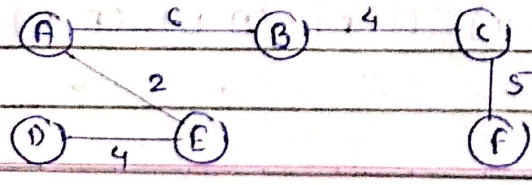


(A,E)	(D,F)	(B,C)	(A,D)	(C,F)	(A,B)	(E,F)	(E,C)
2	4	4	5	5	6	6	7

u, v	j, k	i	A	B	C	D	E	F	min cost
		0	A	B	C	D	E	F	
(A, E)	(A, E)	1	A	B	C	D	E	E	0
(D, E)	(D, A)	2	A	B	C	A	E	E	0+2 = 2
(B, C)	(B, C)	3	A	B	C	A	A	E	2+4 = 6
(A, D)	(A, A)	3	A	B	B	A	A	E	6+4 = 10
(C, F)	(B, F)	4	A	B	B	A	A	E	10+0 = 10
(A, B)	(A, B)	5	A	A	A	A	A	B	10+5 = 15
(E, F)	(A, A)	6	A	A	A	A	A	A	15+6 = 21
(E, C)	(A, A)	7	A	A	A	A	A	A	21
			A	A	A	A	A	A	21
							min cost		21

b) prims algorithm

(k, u)	min cost	A	B	C	D	E	F	Newly inserted
(A, E)	2	0	A	E	E	0	E	
			6	7	4		6	(D, E)
Select minimum edge	2+4 = 6	0	A	E	0	0	E	
			6	7			6	(A, B)
	6+6 = 12	0	0	B	0	0	E	
				4			6	(B, C)
	12+4 = 16	0	0	0	0	0	C	(E, C)
							5	
	16+5 = 21	0	0	0	0	0	0	



min cost = 21

A	D	A	B	F
E	E	B	C	C

Q.7

=) ROLL_NO = 246245

$x = (ROLL\_NO + 50) \% 10$

$x = (246245 + 50) \% 10$

$x = 5$

Frequencies of message 3, 15, 9, 19, 11, 25, 5

Sort the frequencies in ascending order

3, 5, 9, 11, 15, 19, 25

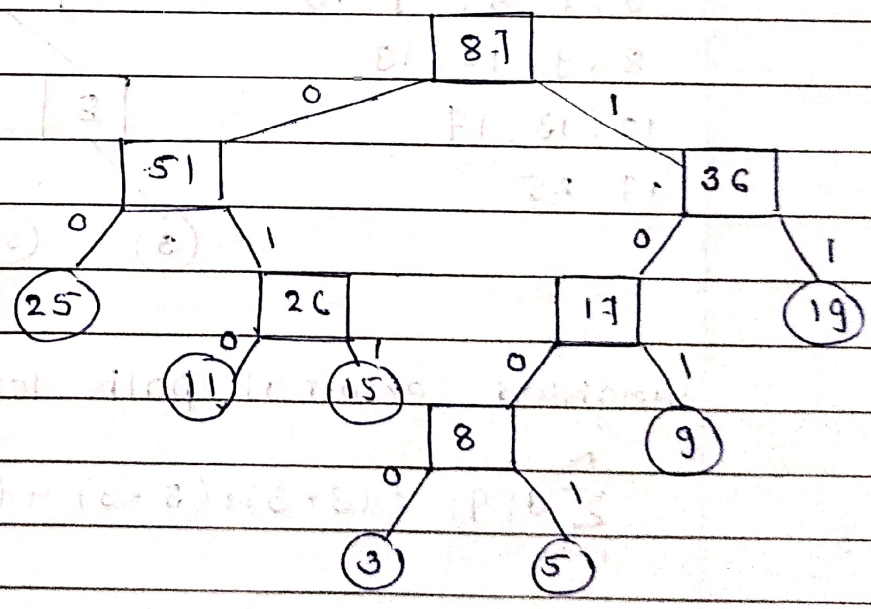
8, 9, 11, 15, 19, 25

11, 15, 17, 19, 25

17, 19, 25, 26

25, 26, 36

36, 51



Huffman code

Frequence

3 → 1000

5 → 1001

9 → 101

19 → 11

15 → 011

11 → 010

25 → 00

Q. 8

=> ROLL-NO : 246245

x = (ROLL-NO + 50) % 10

x = (246245 + 50) % 10

x = 5

Files = (3, 15, 9, 19, 11, 25, 8

Files : (3, 7, 5, 9, 5, 13)

Sort in ascending order

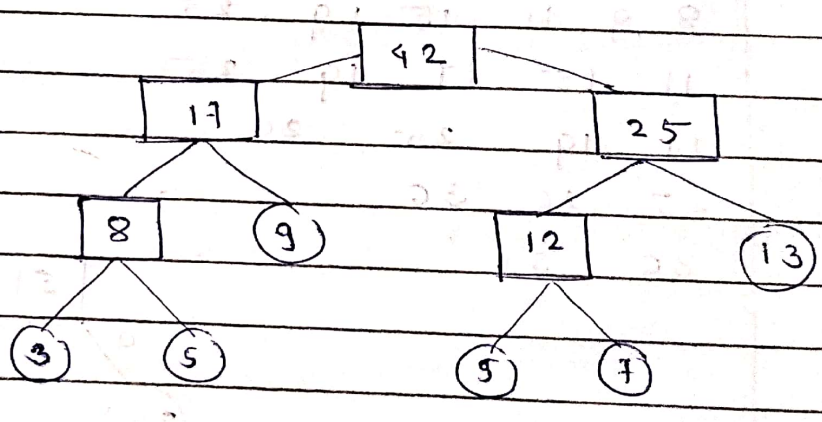
3, 5, 5, 7, 9, 13

5, 7, 8, 9, 13

8, 9, 12, 13

12, 13, 17

17, 25



Weighted external path length

$$\sum_{i=1}^{n} d_i q_i = (3*3)+(5*3)+(9*2)+(5*3)+(7*3)+(13*2)$$

$$= 9 + 15 + 18 + 15 + 21 + 26$$

$$= 104$$

Page No.
Date

Q. 9]

=> for i := 1 to n do ---- (n)

for j := n/2 to 1 do ---- n*(n/2)

write(i) ---- n*(n/2)

$= n + n*(\frac{n}{2}) + n*(\frac{n}{2})$

$= n + \frac{2*n*n}{2}$

$= n + n^2$

$= n^2 + n$

In asymptotic notation $f(n) = n^2 + n = O(n^2)$

Q. 10]

=> Algorithm Exponent (a, n)

{

p = 1; ---- (1)

for i := 2 to n do ---- (n-2+1+1) = n

p = a*p; ---- (n-1)

return p; ---- (1)

}

$= 1 + n + n - 1 + 1$

$= 2n + 1$

Page No.
Date

Q-11

Sr No	Algorithm	Time complex (Best)	Time complexit (Average)	Time complexi (worst)	Space complex	Stable	In-place	Best case input	Worst case input
1.	Bubble sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$	Yes	Yes	Already sorted	Reverse
2.	Selection sort	$\Omega(n^2)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$	No	Yes	-	Any unsorted input
3.	Insertion sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$	Yes	Yes	Already sorted	Reverse
4.	Merge sort	$\Omega(n \log n)$	$\Theta(n \log n)$	$O(n \log n)$	$O(n)$	Yes	No	-	Any unsorted input
5.	Quick sort	$\Omega(n \log n)$	$\Theta(n \log n)$	$O(n^2)$	$O(\log n)$	No	Yes	Randomized data	Already sorted
6.	Heap sort	$\Omega(n \log n)$	$\Theta(n \log n)$	$O(n \log n)$	$O(1)$	No	Yes	-	Any unsorted input
7.	Counting sort	$\Omega(n+k)$	$\Theta(n+k)$	$O(n+k)$	$O(k)$	Yes	No	small range of k	large range of k
8.	Radix sort	$\Omega(nk)$	$\Theta(nk)$	$O(nk)$	$O(n+k)$	Yes	No	small digit size	large digit size
9.	Bucket sort	$\Omega(n+k)$	$\Theta(n+k)$	$O(n^2)$	$O(n)$	Yes	No	uniform data	Highly skewed data