Data Structures and Algorithms Refresher Module - 2024

Final Exam - 6 August, 2024

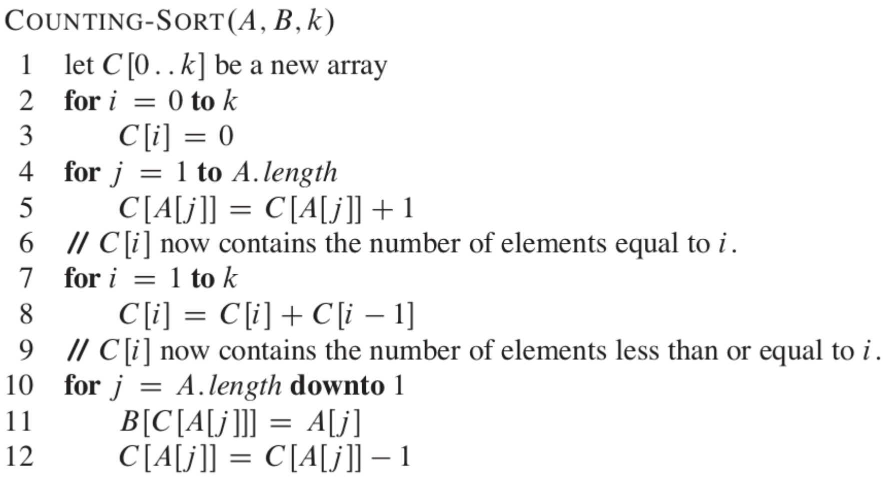
Maximum Marks: 70 **Duration: 120 mins**

Use of mobile devices/ computers/ tablets is prohibited. Using books/ notes or any other reference material is not permitted.

Attempt all questions; Show all the working steps wherever necessary.

All the best !!

1. **[i] (6 marks)** The counting sort algorithm is given as below:



```
COUNTING-SORT$(A, B, k)$
let $C[0..k]$ be a new array
for $i = 0$ to $k$
    $C[i] = 0$
for $j = 1$ to $A.length$
    $C[A[j]] = C[A[j]] + 1$
// $C[i]$ now contains the number of elements equal to $i$.
for $i = 1$ to $k$
    $C[i] = C[i] + C[i-1]$
// $C[i]$ now contains the number of elements less than or equal to $i$.
for $j = A.length$ downto 1
    $B[C[A[j]]] = A[j]$
    $C[A[j]] = C[A[j]] - 1$
```

Illustrate the working of the above algorithm with an example where *k = 5* and *A.length >= 6*. The sample array should have at least one element that has more than one occurrence.

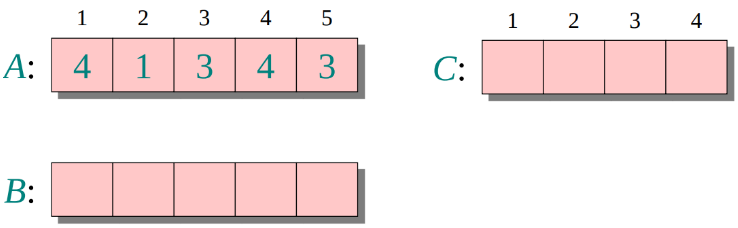
Answer:

An example array of size >=6 with values ranging from 0 to 5 (value of k given) to be taken.

At least one number to be repeated twice or more in this array.

(0.5 mark)

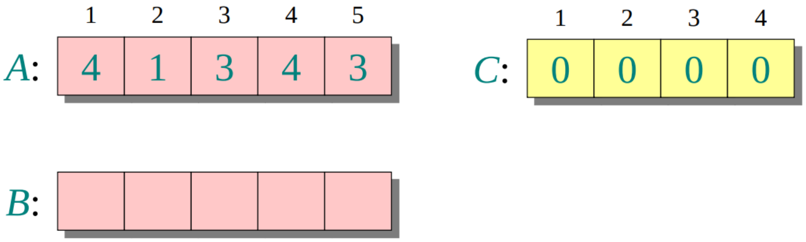
Example with *n=5* elements and *k=4*. Students should take a similar example of *n=6* elements and *k=5*.



	1	2	3	4	5
A:	4	1	3	4	3
B:					

	1	2	3	4
C:				

After the execution of the first "for" loop:



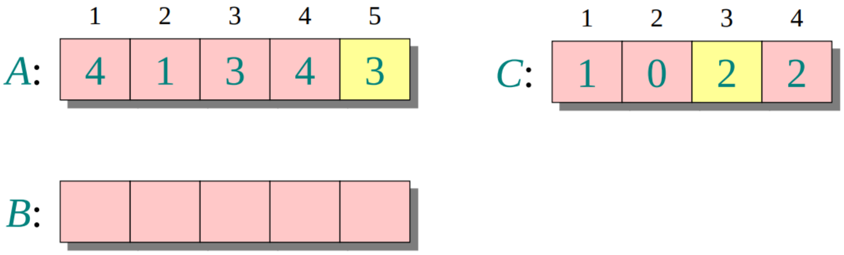
	1	2	3	4	5
A:	4	1	3	4	3
B:					

	1	2	3	4
C:	0	0	0	0

```
for $i \leftarrow 1$ to $k$
    do $C[i] \leftarrow 0$
```

(1 mark)

After the execution of the second "for" loop:



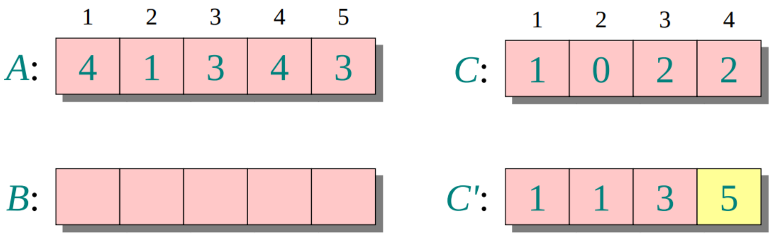
	1	2	3	4	5
A:	4	1	3	4	3
B:					

	1	2	3	4
C:	1	0	2	2

```
for $j \leftarrow 1$ to $n$
    do $C[A[j]] \leftarrow C[A[j]] + 1$    ⊳ $C[i] = |\{\text{key} = i$
```

(1.5 marks)

After the execution of the third "for" loop:



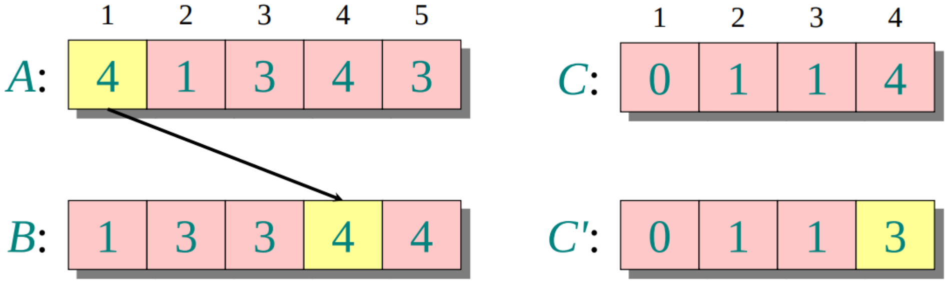
	1	2	3	4	5
A:	4	1	3	4	3
B:					

	1	2	3	4
C:	1	0	2	2
C':	1	1	3	5

```
for $i \leftarrow 2$ to $k$
    do $C[i] \leftarrow C[i] + C[i–1]$        ⊳ $C[i] = |\{\text{key} \le i\}|$
```

(1.5 marks)

After the execution of the fourth "for" loop:



	1	2	3	4	5
A:	4	1	3	4	3
B:	1	3	3	4	4

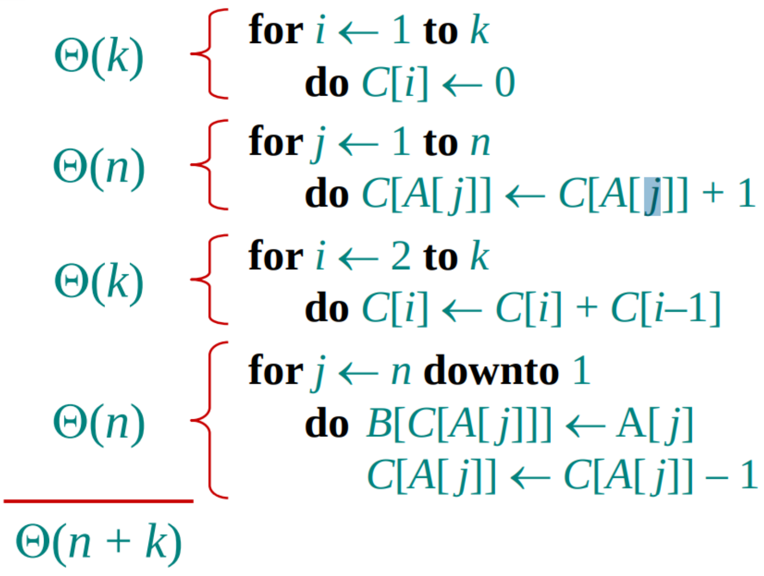
	1	2	3	4
C:	0	1	1	4
C':	0	1	1	3

```
for $j \leftarrow n$ downto 1
    do $B[C[A[j]]] \leftarrow A[j]$
        $C[A[j]] \leftarrow C[A[j]] – 1$
```

(1.5 marks)

[ii] (2 marks) Discuss the time complexity of the above algorithm.

Answer:



```
$\Theta(k)$    for $i \leftarrow 1$ to $k$
                   do $C[i] \leftarrow 0$
$\Theta(n)$    for $j \leftarrow 1$ to $n$
                   do $C[A[j]] \leftarrow C[A[j]] + 1$
$\Theta(k)$    for $i \leftarrow 2$ to $k$
                   do $C[i] \leftarrow C[i] + C[i-1]$
$\Theta(n)$    for $j \leftarrow n$ downto $1$
                   do $B[C[A[j]]] \leftarrow A[j]$
                      $C[A[j]] \leftarrow C[A[j]] - 1$
$\Theta(n + k)$
```

(0.5 mark each for deriving the TC for each loop.)

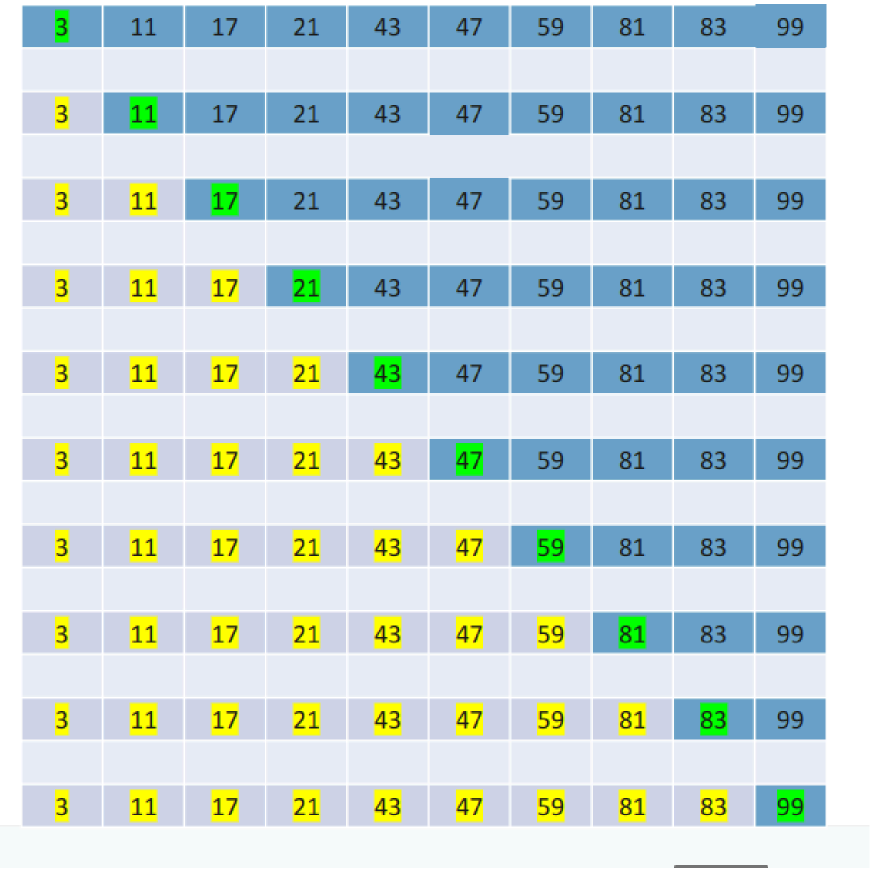
The TC can be expressed in Big-Oh or Big-Theta in the answer.
Full marks can be given even if the final answer is expressed only in terms of n, like O(n)

2. **[i] (8 marks)** Discuss the worst case of Quicksort and illustrate with a suitable example. Choose an array with array size as $n >= 5$.

Answer:
In the worst case, the partition algorithm generates two arrays of size zero and (n-1) for further sorting.
This happens when the array is already sorted and the first element is chosen as the pivot in each step.

(2 marks)

Example:
Array of size n>=6 to be chosen



3	11	17	21	43	47	59	81	83	99
3	11	17	21	43	47	59	81	83	99
3	11	17	21	43	47	59	81	83	99
3	11	17	21	43	47	59	81	83	99
3	11	17	21	43	47	59	81	83	99
3	11	17	21	43	47	59	81	83	99
3	11	17	21	43	47	59	81	83	99
3	11	17	21	43	47	59	81	83	99
3	11	17	21	43	47	59	81	83	99
3	11	17	21	43	47	59	81	83	99

(4 marks)

Recurrence:

$$\boxed{\begin{aligned} T(n) &= T(n-1) + T(0) + \Theta(n) \\ &= T(n-1) + \Theta(n)\,. \end{aligned}}$$

Worst Case Time Complexity : O(n^2)

(2 marks)

3. **[i] (4 marks)** State Master Theorem.

Answer:

A recurrence relation of the form

$$T(n) = aT\left(\frac{n}{b}\right) + cn^k, where\ a \geq 1, b \geq 2, c \geq 0, k \geq 0$$

has the following solution

$$T(n) \ = \ \begin{array}{cc} O(n^{\log_b a}) & if\ a > b^k \\ O(n^k logn) & if\ a = b^k \\ O(n^k) & if\ a < b^k \end{array}$$

[ii] (4 marks) An algorithm solves a problem correctly in 10 steps for input size of *n <= 3*. For all higher input sizes, it divides the input instance of size *n* into 3 different subproblems, each of size *floor(n/3)*. It then recursively solves each of these subproblems and combines the solutions in *$9n^2$ + 4n* time. Write the recurrence relation corresponding to the running time of this algorithm. Solve this using the Master theorem.

Answer:

Recurrence relation:

Base case: T(n) = O(1), for n<=3

Recursive step: T(n) <= 3T(n/3) + ($9n^2$ + 4n), for n>3

<= 3T(n/3) + f(n^2)

= O($\log_3$ n) + O(n^2)

Applying Master Theorem, a = 3, b=3, k=2; hence the third case where a < b^k is applicable

Therefore, T(n) = O(n^2)

4. **[i] (4 marks)** A scheduling application uses a *singly linked list* to implement a job queue. The names and priority keys of the jobs are maintained in the queue. Two variables `head` and `tail` are used to track the front and rear end respectively.

   Assume that the *Enqueue* operation ensures that the *Dequeue* operation is done in O(1) time. Write the pseudocode of a function to *Dequeue* a job whenever the processor becomes free to take up the next job.

   Use the `struct node` definition given below to declare the variables `head, tail` and a temporary variable called `current` to hold the element that gets dequeued.

```
struct node {
    char jobname[64];
```

```
        int p_key;
        struct node *next;
    }
```

Answer:

For *Dequeuing* in this scenario, the linked list operation to be performed is *Deletion of the front node.*

`Deque_job(struct node *head){`
 	
`struct node *current;`
	(1 mark)

`if (head==NULL){`
`     printf("The job queue is empty\n");`
`     return -1;`
`}`
	(1 mark)

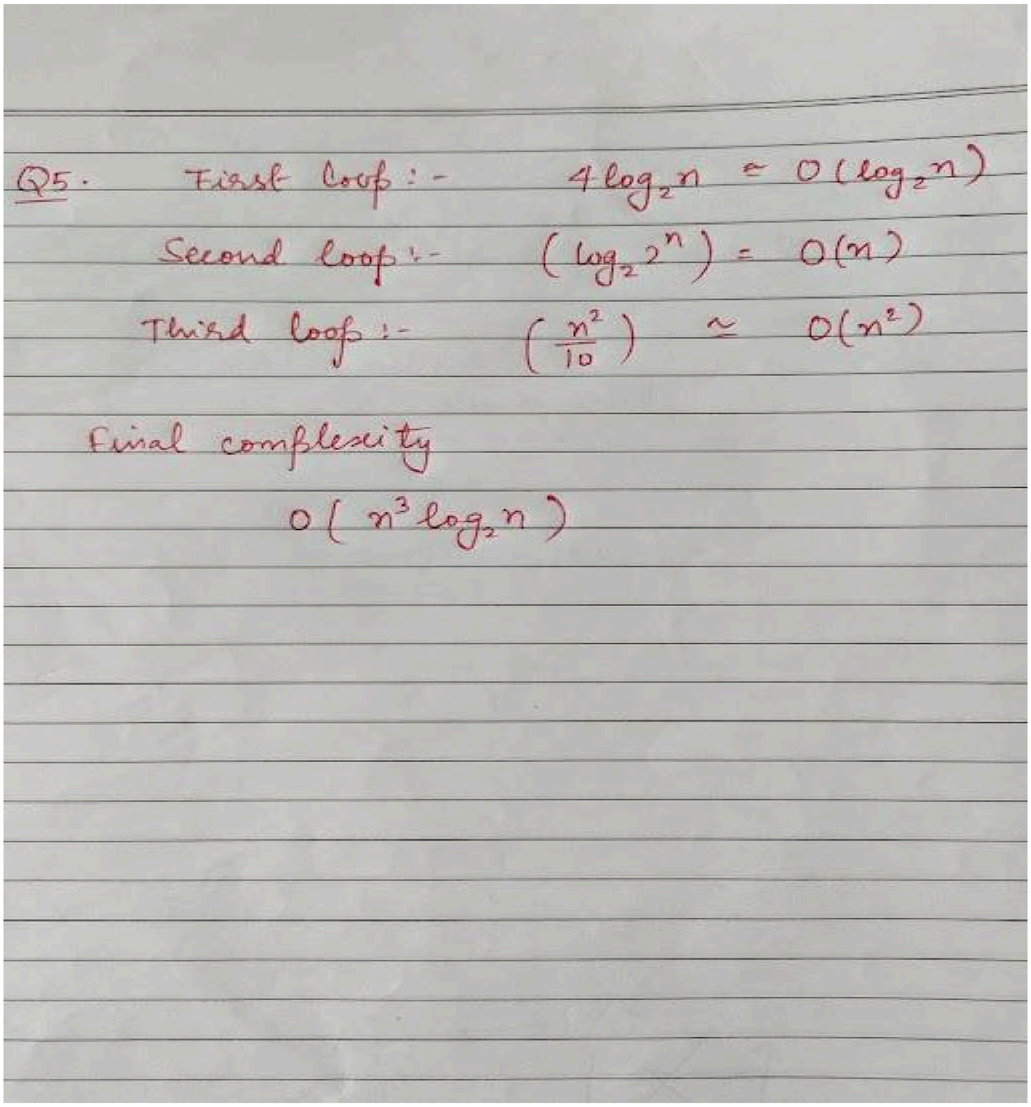
`current = head;`
`head = head ->next;`
`current->next = NULL;`
	(1 mark)

`return(current);`
`}`
	(1 mark)

5. **(4 marks)** Find time complexity of the following segment of code: Express your answer in Big-Oh notation in terms of the variable *n*.

```
x=0;
for(i=n^4;i>=1;i= i/2)
     for(j=2;j<=2^n;j=j^2)
          for(k=n^2;k>=1;k=k-10)
               x=x+1;
```

Answer:



Q5. First Loop :- $4\log_2 n \simeq O(\log_2 n)$

Second loop :- $(\log_2 2^n) = O(n)$

Third loop :- $\left(\frac{n^2}{10}\right) \simeq O(n^2)$

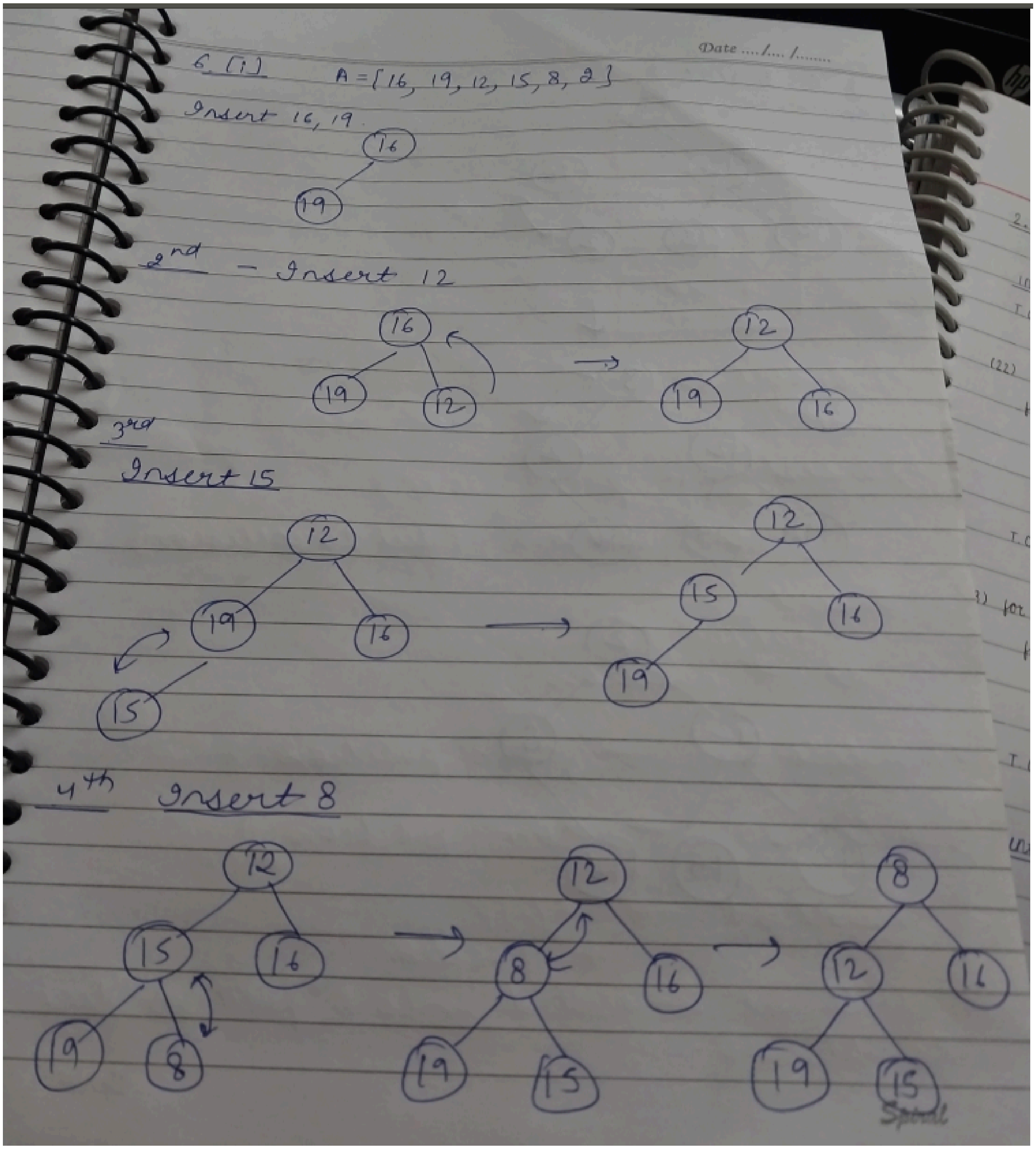
Final complexity

$O(n^3 \log_2 n)$

(1 mark each for each step + 1 mark for the final answer)

6. **[i] (8 marks)** Illustrate the operation of the standard heap algorithm *Build-Min-Heap* on the array A = {16, 19, 12, 15, 8, 2}, showing the order of the elements of the array and the heap diagram at each stage.

Answer: (Please refer to the pics given below)
(1 mark each for each insertion + 0.5 mark extra for steps involving exchanges)



Date ..../..../........

6. (i) A = {16, 19, 12, 15, 8, 2}

Insert 16, 19.

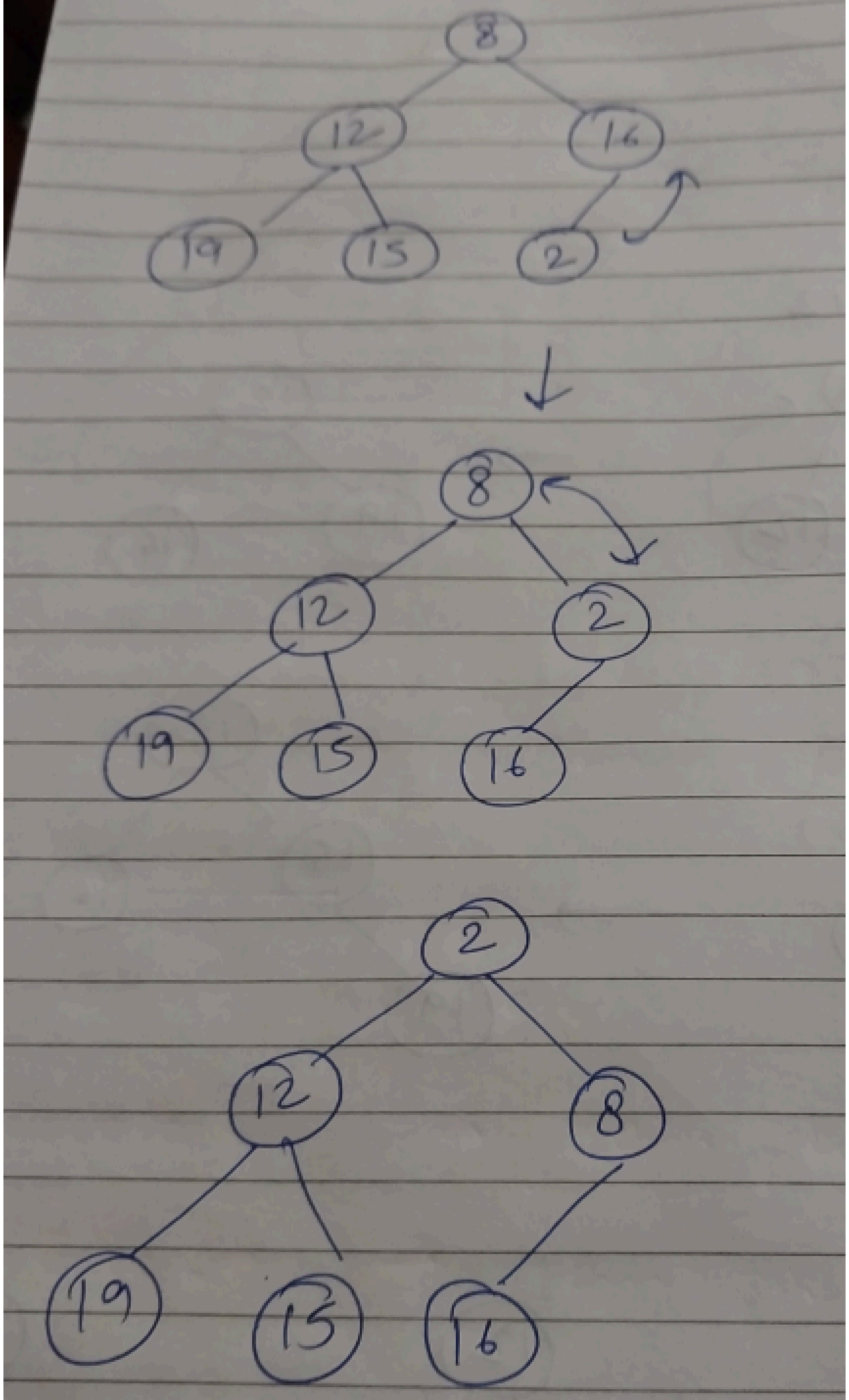
2nd - Insert 12

3rd

Insert 15

4th Insert 8

5th Insert 2



[ii] (6 marks) Derive the time complexity of the algorithm to find *kth* smallest in a Min-heap of *n* distinct elements. Assume the use of a suitable standard heap algorithm discussed in class. Explain your analysis.

Answer:

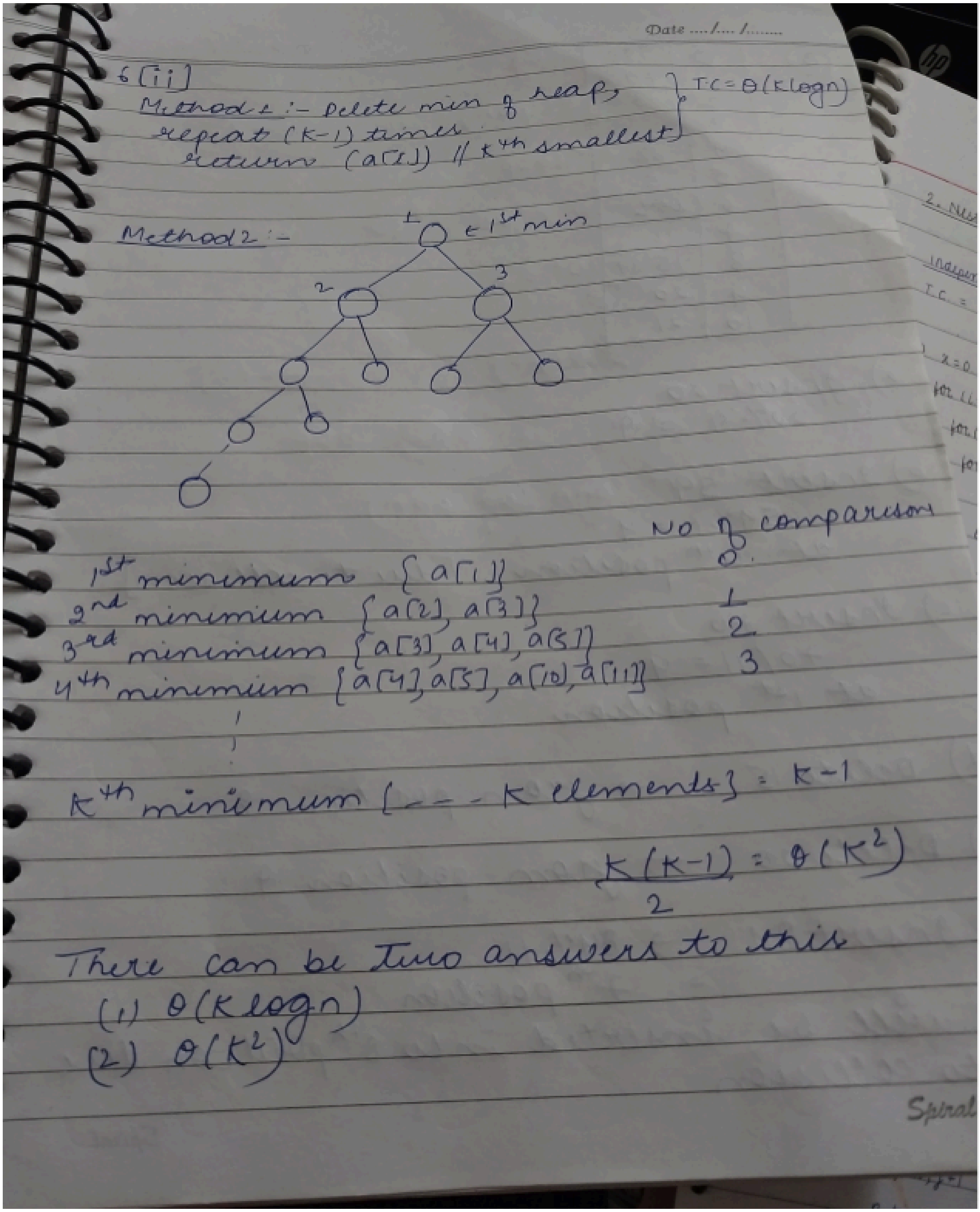
The standard heap algorithm Min-Heap-Extract-Min can be used for *k* number of iterations.

Time complexity of one call to Min-Heap-Extract-Min is O(log n), hence for k iterations it is O(k *logn).

(3 marks)

Illustration with an example /Explaining how TC of Min-Heap-Extract-Min is O(log n) (3 marks)

(If they use Method 2 given in the answer, they have to mention which standard heap algorithm is being used)



Date ..../..../........

6 [ii]

Method 1 :- Delete min of heap, repeat (K-1) times return (a[1]) // K^{th} smallest } T.C = $\theta(K \log n)$

Method 2 :-

		No of comparison
1st minimum	{a[1]}	0
2nd minimum	{a[2], a[3]}	1
3rd minimum	{a[3], a[4], a[5]}	2
4th minimum	{a[4], a[5], a[10], a[11]}	3

K^{th} minimum {- - - K elements} : K-1

$$\frac{K(K-1)}{2} = \theta(K^2)$$

There can be Two answers to this

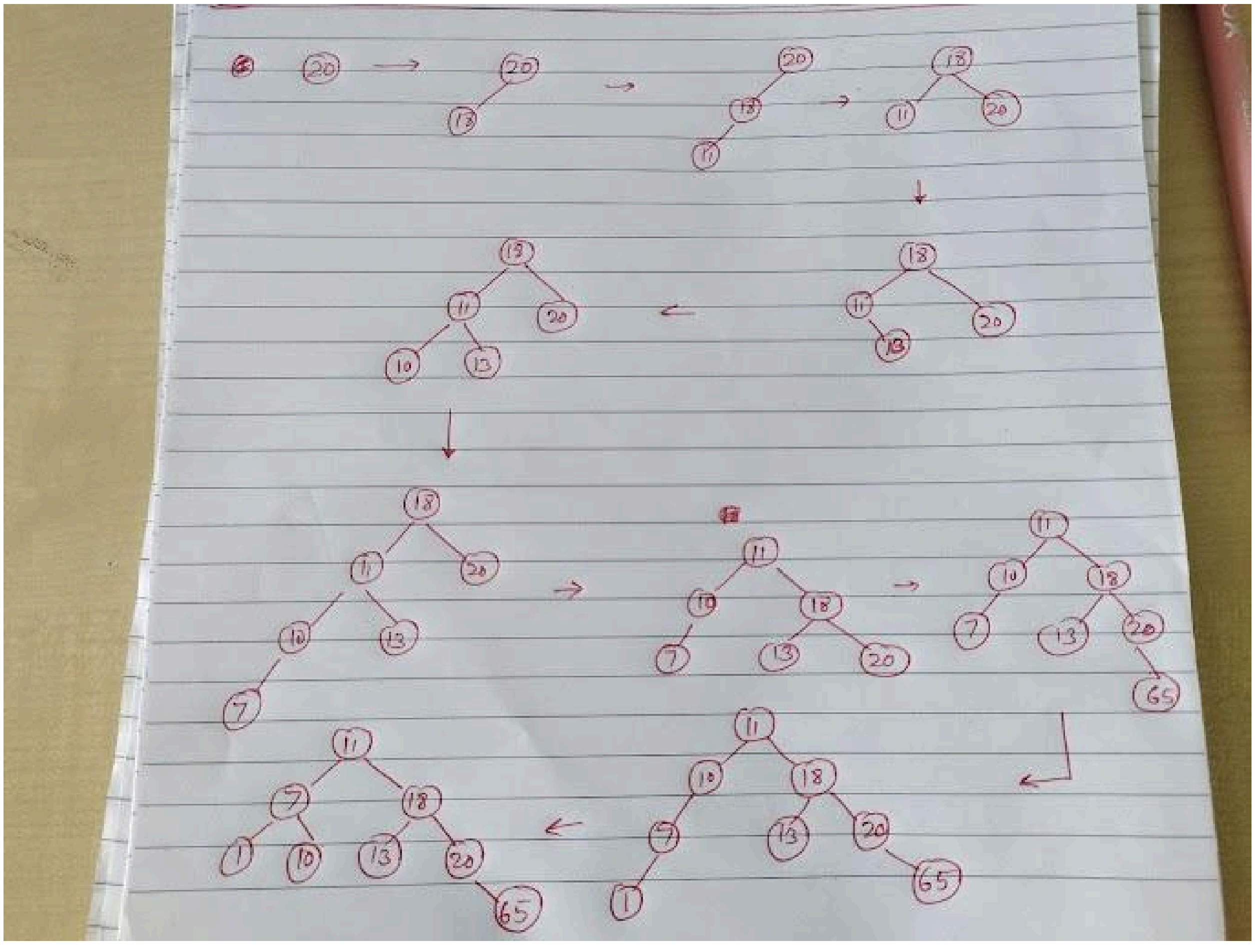
(1) $\theta(K \log n)$

(2) $\theta(K^2)$

7. **(8 marks)** Perform the steps and rotations to build an AVL tree for the below sequence of numbers:

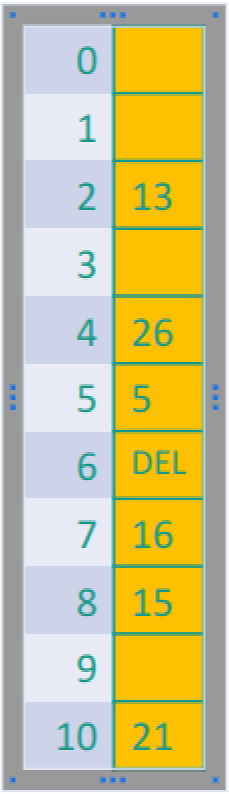
   20, 18, 11, 13, 10, 7, 65, 1

Answer:



(1.5 marks each for each rotation step + 0.5 mark each for steps without rotation)

8. **(6 marks)** Consider the following Hash Table that uses the method of Linear Probing to resolve collisions.



0	
1	
2	13
3	
4	26
5	5
6	DEL
7	16
8	15
9	
10	21

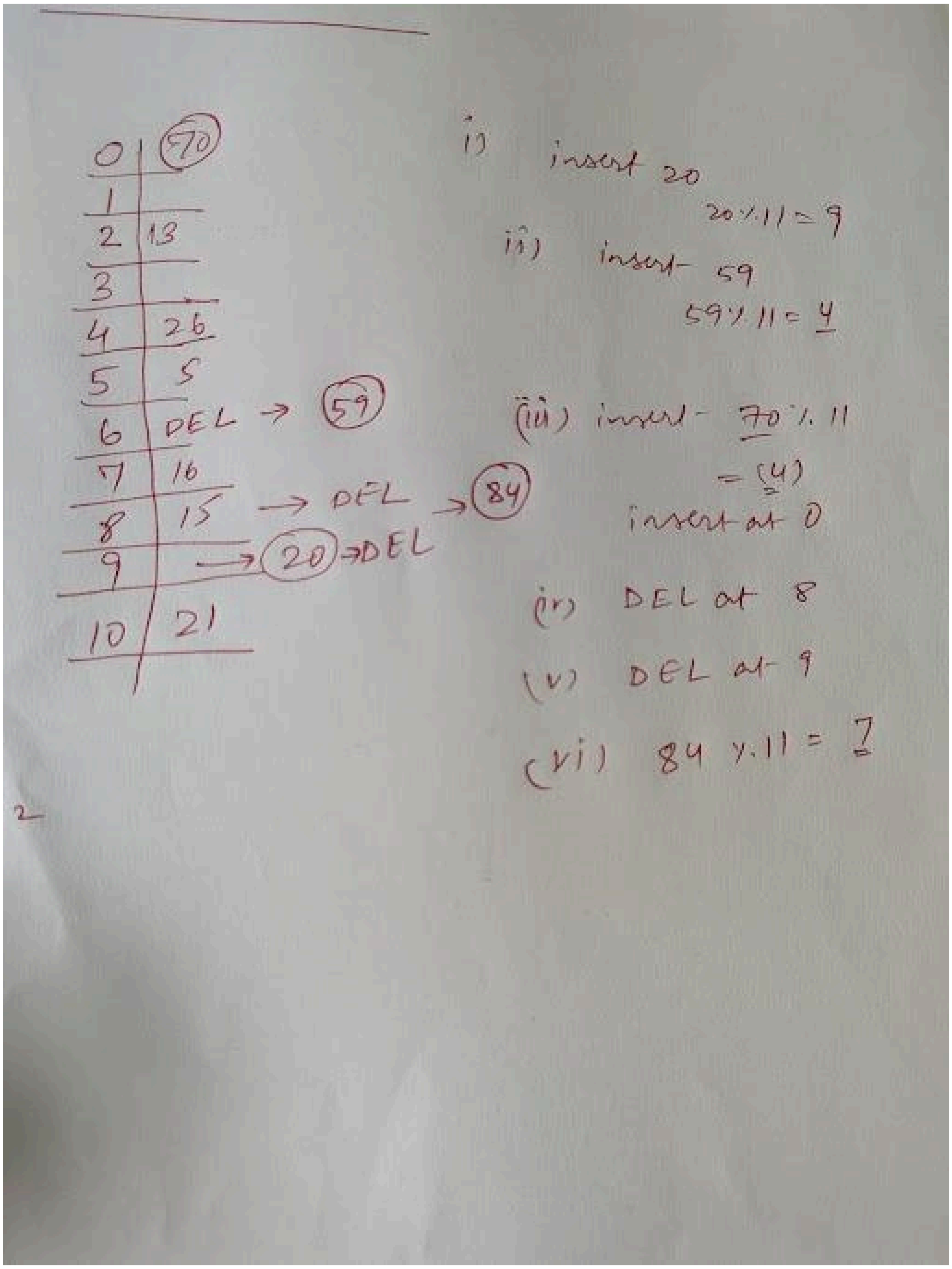
The hash function being used is:

```
int HashFun(int key) {
    return key % 11;
}
```

Perform the following operations in the sequence given. Show the hash table contents after each step with a brief explanation:

a) Insert 20
b) Insert 59
c) Insert 70
d) Delete 15
e) Delete 20
f) Insert 84

Answer:



0	70
1	
2	13
3	
4	26
5	5
6	DEL → 59
7	16
8	15 → DEL → 84
9	→ 20 → DEL
10	21

i) insert 20

20%11=9

ii) insert 59

59%11 = 4

(iii) insert 70 % 11

= (4)

insert at 0

(iv) DEL at 8

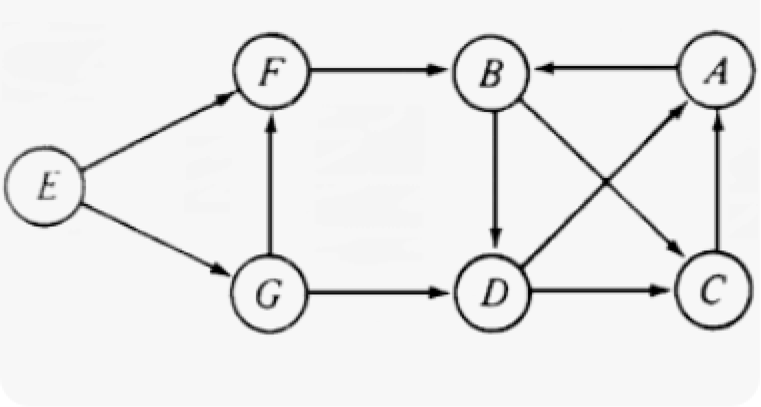
(v) DEL at 9

(vi) 84 %11 = 7

2

(1 mark each for each operation. Ensure that in operation 2 DEL is replaced and subsequently operation 3 places 70 in 0th position. 0 marks for these steps if this is not correct)

9. Consider the following Graph G.



Graph *G = (V, E)*

[i] (2 marks) Show the adjacency list of G, assuming that each list is ordered alphabetically

Answer (Please refer to the pic below 9[ii])

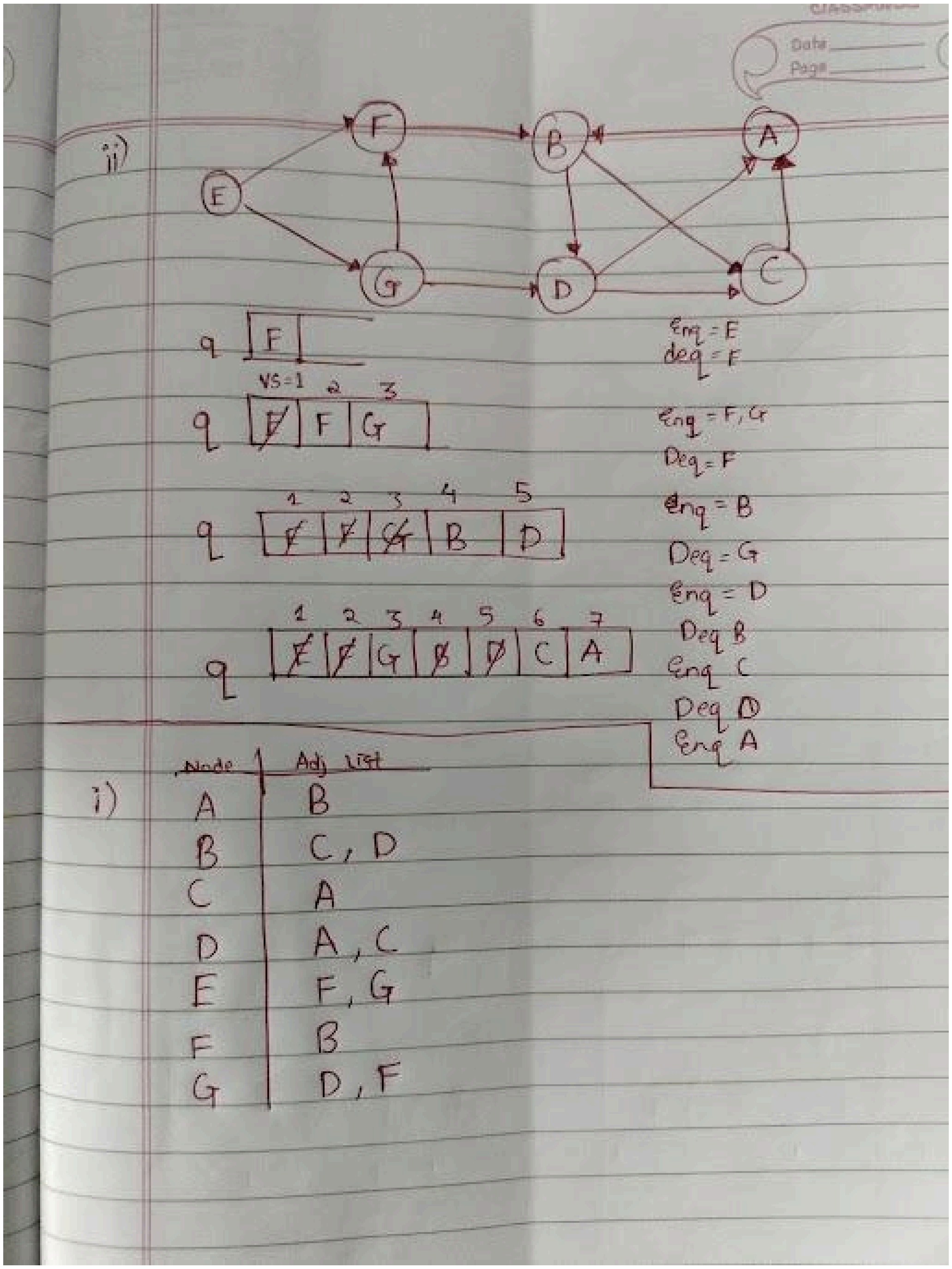
Reduce 0.5 marks if the adjacency lists are not in alphabetical order.

[ii] (8 marks) Show how breadth-first search operates on the graph G, starting from the vertex E. Show the visit time of the vertices and also the enqueuing and dequeuing of vertices as applicable. Assume that the vertices are visited in alphabetical order, when there are more than one vertices in an adjacency list.

Answer (please refer to the pic below)

5 marks for enqueue-dequeue steps - 1 mark each for each enqueue-dequeue

- **+ 1 mark for choosing the vertices in the correct order**
- **+ 2 marks for the final BFS tree.**



Date
Page
E
F
G
B
D
A
C

ii)

q

F	

VS=1	2	3
~~E~~	F	G

q

1	2	3	4	5
~~F~~	~~F~~	~~G~~	B	D

q

1	2	3	4	5	6	7
~~E~~	~~F~~	G	~~B~~	~~D~~	C	A

q

Enq = E
deq = F

Enq = F, G
Deq = F
Enq = B
Deq = G
Enq = D
Deq B
Enq C
Deq D
Enq A

i)

Node	Adj list
A	B
B	C, D
C	A
D	A, C
E	F, G
F	B
G	D, F

—-------------------------------- xxx —--------------------------------