DS
Answers

① Selection sort for these set of numbers 14,22,80,18,67,26,45,56,9.

L = {14,22,80,18,67,26,45,56,9}

Pass	List L(Process)
1	14,22,80,18,67,26,45,56,9
	Here minimum element is 9 so compare with first indexed element & swap
	9,22,80,18,67,26,45,56,14
2	Here next minimum element is 14 so compare with second indexed element & swap
	9,14,80,18,67,26,45,56,22
3	Here next minimum element is 18 so compare with third indexed element & swap
	9,14,18,80,67,26,45,56,22
4.	Here next minimum element is 22 is compared with next indexed element & swap
	9,14,18,22,67,26,45,56,80
5	Here next minimum element is 26 is compared with next indexed element & swap
	9,14,18,22,26,67,45,56,80
6	Here next minimum element is 45 is compared with next indexed element & swap
	9,14,18,22,26,45,67,56,80
7	Here next minimum element is 56 is compared with next indexed element & swap
	9,14,18,22,26,45,56,67,80
8.	Here next minimum element is 67 is compared with i=8 indexed element
	9,14,18,22,26,45,56,67,80

Hence L= {9,14,18,22,26,45,56,67,80} is the obtained sorted list using selection sort by (n-1) number of passes.

(b) Algorithm for Bubble Sort :-

* In the process of bubble sort, pairs of elements are checked
* The pairs that are out of order are interchanged until the ~~the~~ whole list is ordered.
* So, if there are 'n' no. of elements in the list then (n-1) passes will be resulted finally.
* At each pass the next largest element of list called 'bubbles' are sent to appropriate positions of the list

Algorithm:-

Procedure bubble sort (L, n)
  for i=1 to n-1 do
  for j=1 to n-i do
  if (L[j]>L[j+1])
    swap[L[j], L[j+1]];
  end
  end
end Bubble sort

2(a) Quick sort for the following 45, 31, 55, 77, 63, 99, 22.

LOC	1	2	3	4	5	6	7
	45	31	55	77	63	99	22

↑left

if (1<9) True

so position (A, 1, 7).

Now apply partition

  loc = left = 1 [i.e position]

while [1<9] True so

while [45 ≤ 22] and (1<9)
False

if (45 > 22) True then

swap (45, 22)

loc = right = 7

left = left+1
  = 2

22 31 55 77 63 99 45 ← loc

left (7>2) right

while $(31 \le 45)$ & ~~(2<7)~~ True

~~right = righ-1 = 6~~ left = left+1

22 31 55 77 63 99 45 loc

left right

22 31 55 77 63 99 45

if [A [loc] < A[left]] ⇒ 45 < 55 true

~~loc = left = 5~~

~~right = right −1 = 7-1=6~~

45 < 55

swap 45, 55

loc = left = 5

right = right −1 = 7-1=6

22 31 45 77 63 99 55

loc, left, right

$(45 \le 99)$ $(3 < 6)$

True

22 31 45 77 63 99 55

loc, left, right

$(45 \le 63)$ $(3 < 5)$

True

right = right −1

22 31 45 77 63 99 55

loc, left, right

$(45 \le 77)$ $(3 < 4)$

loc right = right −1

$\underbrace{22 \quad 31}_{I}$ $\boxed{45}$ 77 $\underbrace{63 \quad 99 \quad 55}_{II}$

right left

45 is pivot element

The first partition is in order. So consider 2nd partition

	loc 1	2	3	4
consider	77	63	99	55
	left			right

if (A(loc) > A(right))

77 > 55

swap (77 & 55)

loc = right = 4

left = left+1 = 2

			loc
55	63	99	77
	left		right

while (A(loc) ≥ A(left)] and (loc > left) do

left = left+1

(77 ≥ 63) & (4>2) True

left = 3

			loc
55	63	99	77
		left	right

if [A(loc) < A(left)]

77 < 99

swap (99, 77)

loc = left

right = right - 1 = 3

		loc	
55	63	77	99
		right	left

77 is pivot element

Finally combine all partition and pivots in order obtained we get

22 31 45 55 63 77 99.

Scanned with CamScanner

2(B) Merge Sort Algorithm

```
Algorithm merge sort (low, high)
{
 if (low < high) then
 {
  mid = [(low+high)/2];
  merge sort (low, mid);
  merge sort (mid+1, high);
  merge (low, mid, high);
 }
}
Algorithm merge (low, mid, high)
{
 h = low ; i = high ; j = mid+1;
 while ((h <= mid) and (j <= high)) do
 {
  if (a[h] ≤ a[i]) then
  {
   b[i] == a[h];
   h = h+1;
  }
  else
  {
   b[i] == a[j];
   j = j+1;
  }
  i = i+1;
 }
 if (h > mid) then
 for (k = j to high do)
 {
  b[i] == a[k];
  i = i+1;
 }
 else
 for k = h to mid do
 {
  b[i] == a[k];
  i == i+1;
 for K = low to high do
  a[k] == b[k];
 }
```

(3)(a) Collision resolution ? Various open addressing methods to collision resulsion with an example

Hash functions are there to map different keys to unique locations and any hash function which is able to do so is known as the perfect has function. Since the size of the hash table is very less comparitively to range of keys, The Perfect has function is practically impossible. If more than one keys map to the same location and this is known as Collision.

The way of handling collisions when two or more items should be kept in the same location especially in a hashtable is known as collision resolution.

Various Open Addressing Methods are :-

(1) Linear Probing technique

(2) Quadratic Probing technique

Linear Probing Techniques :-

Ex:- 11, 10, 20, 23, 25, 22

size = 10

H(11) = 11 % 10 = 1

H(10) = 10 % 10 = 0

H(20) = 20 % 10 = 0

H(23) = 23 % 10 = 3

H(25) = 25 % 10 = 5

H(22) = 22 % 10 = 2

0	10
1	11
2	20
3	23
4	22
5	25
6	
7	
8	
9	

'0' is filled and next higher address will be check. 1 is also filled and higher will be checked i.e 2 ∴ 2 is empty 20 will be stored at position 2

Quadratic Probing Technique :-

In this technique the key elements are stored by the process of probing

(1) Assign J=0

(2) Get the hashvalues by $(k+j^2)$ mod size

$$H(k) = (k+j^2) \% \text{ size}$$

Ex:- Insert the element 76, 40, 48, 5, 20 with maximum size as 7.

$H(76) = (76+0)\%7$
$= 6$

$H(40) = (40+0)\%7$
$= 5$

$H(48) = (48+0)\%7$
$= 6$

0	48
1	
2	5
3	~~20~~
4	20
5	40
6	76

~~H(48)=~~

As 6 is already filled now j = 1

$H(48) = (48+1)\%7$
$= 0$

$H(5) = (5+4)\%7$
$= 2$

$5 < 7$ it is false
$(5+1^2) = 6 < 7$ is also ~~does~~ false

$H(20) = 20\%7$
$= 6 \neq$
$= 21\%7$
$= 0$

$(20+4)\%7$
$= 24\%7$
$= 3$

3(b) Binary Search for the set of numbers 11, 22, 33, 44, 55, 66, 77.

$L = \{ \overset{1}{11}, \overset{2}{22}, \overset{3}{33}, \overset{4}{44}, \overset{5}{55}, \overset{6}{66}, \overset{7}{77} \}$

As they haven't specified any key element to search so we can consider any of the key to search.

Let $K = 22$

step-1:- low high mid
1 7 $\left[\frac{1+7}{2}\right] = [8/2] = 4$

(i) $K = 22$ mid = 4
$22 = L[4]$
$22 = 44$ [False]

(ii) ~~22 <~~ (ii) $22 < L[4]$
$22 < 44$ [True] so
[L, low, mid-1, K]
[L, 1, 3, 22]

step - 2 :- low high mid

1 3 $\left[\frac{1+3}{2}\right] = \frac{4}{2} = 2$

K = 22 mid = 2

22 = L[2]

22 = 22 True

"key found"

and return L[2].

<u>case - ii</u> Let K = 20

<u>step - i</u> :- low high mid

1 7 $\frac{1+7}{2} = 8/2 = 4$

(i) K = 20 mid = 4

20 = L[4]

20 = 44 [False]

(ii) 20 < L[4]

20 < 44 [True] so (L, low, mid-1, key]

<u>step - ii</u> :- low high mid

1 3 4/2 = 2

(i) K = 20 L[2] = 22 mid = 2

20 = L[2]

20 = 22 False

(ii) 20 < L[2]

20 < 22 [True] so [L, low, mid-1, key]

[L, 1, 1, 20]

<u>step - iii</u> :- low high mid

1 1 $\frac{1+1}{2} = 1$

(i) K = 20 L[1] = [11]

20 = L[1]

20 = 11 (False)

(ii) 20 < L[1]

20 < 11

False

(iii) Return "key is not found."