Name - Rohit Prakash.
Branch - B.Tech CCSE
Sec - A.

Date 23/04/20

Q.1. Explain the ~~following~~ working of Binary search in the following array.
$a[8] = \{5, 3, 9, 2, 7, 8, 6, 1\}$

∵ for a binary search it is compulsary that the target array should be sorted. lets take an array 5,3,9,2,7,8,6,1 our sorted array is:

1	2	3	5	6	7	8	9

lets assume we have to search for location of 6 using the binary search.

	0	1	2	3	4	5	6	7	
b →	1	2	3	5	6	7	8	9	→ h

we will point b at 0 index and h at n-1 index i.e, 7th index.

$$\because Mid = \frac{b+h}{2} = \frac{0+7}{2} = 3.$$

so, we take Mid = 3

Comparing the mid value to the stored search element <6>

$a[3] = 5 \neq 6$

$\because a[mid] = a[5] < 6$

∴ target element value must be in the right side of the array.

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

Now ∵ a[Mid] < search element

$\therefore b = Mid + 1.$

$= 3 + 1 = 4$

$$Now, mid = \frac{b+h}{2} = \frac{4+7}{2} = \frac{11}{2} = 5$$

we get, h = 4

$$mid = \frac{b+h}{2} = \frac{4+4}{2} = \frac{8}{2} = 4.$$

Now, on compairing the value of a[4] to the searching element. we get

a[4] = 6 = searching element

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

we conclude that the correct location of our search element is 6 at position 4.

Explain the working of bubble sort in following array with algorithm.

6, 3, 5, 2, 9, 7, 1, 10, 4, 8.

i → 0	1	3	4	5	6	7	8	9
6	3	5	2	9	7			

j

i = 0	1	2	3	4	5	6	7	8	9
6	3	5	2	9	7	1	10	4	8

Step 1:- we will compare the value of index a[j] with index of a[j+i]

if a[j] > {a[j+1] then

a[j] ⇔ a[j+1].

at first j=0 , a[0]=6

and 6>3 , 6⇔3 and j → j+1.

3	6	5	2	9	7	1	10	4	8

again a[1]=6

and 6>5 ; 6⇔5 and j → 2.

3	5	6	2	9	7	1	10	4	8

again 6>2 , 6⇔2 and j → 3.

3	5	2	6	9	7	1	10	4	8

Now, 6 ≯ 9 ; j<10-i-j ; j++

initially i=0, then j<10-0-1 ⇒ j=4

Now a[4] > a[4+1]

i.e. 9>7, Comparision occurs

3	5	2	6	7	9	1	10	4	8

Now 9>1, j → 5, 9⇔1.

3	5	2	6	7	1	9	10	4	8

∵ 9≯10, j → 6, but Condition fails so

j → 7

3	5	2	6	7	1	9	10	4	8
0	1	2	3	4	5	6	7↑ j	8	9

Now, 10>4. Condition is true.

the 10⇔4, j → 8

3	5	2	6	7	1	9	4	10	8
0	1	2	3	4	5	6	7	8	9

Now, 10>8. Condition is true.

3	5	2	6	7	1	9	4	8	10

10⇔8

J = J+1.

∴ For loop <inner> will run until j → 9

∴ Now, i → i+1, i → 1 and j will run

until j < 10-i-1 ⇒ j<8

J → 4 , 7 > 1 Condition true. J → 5

3	2	5	6	1	7	9	4	8	10

a[5] = 7

7 ≯ 9, Condition fails J → 6.

3	2	5	6	1	7	9	4	8	10
	↑ i					↑ J			

Now 9 > 4 , Condition true. j → 7.

3	2	5	6	1	7	4	9	8	10

Now 9>8 , Condition true. J → 8

3	2	5	6	1	7	4	8	~~9~~	~~10~~

Now 9 ≯ 10 inner for loop ends.

Now step 1 and 2 will run.
until i → 9 or i < 10
after that outer for loop will end. and we get sorted array.

i.e.

~~3~~	~~5~~	~~2~~	~~6~~	~~7~~	

O/p.

1	2	3	4	5	6	7	8	9	10

→ Algorithm:-

Repeat Steps (ii) and (iii) until i < n and j < n-i-1. After that element in array will be executed as sorted array in output.

logical sector..

```
for (i=0; i<n; i++)
  {
    if (a[j] > a[j+1])
     {
       t = a[j];
       a[j]=a[j+1];
       a[j+1] = t;
```

Explain working of insection sort in
9, 3, 5, 2, 7, 8, 6, 4.

a[8] = {9, 3, 5, 2, 7, 8, 6, 4}.

```
i=0  1→j  2  3  4  5  6  7
 9   3    5  2  7  8  6  4
     → key
```

use another variable key which is pointing to a[1] = a[1]=3, Initially key is 3.

(i) For (j ← 1 to n-1)

(ii) key = a[j]

(iii) i = j-1, Initially j = 1, ∴ i = 1-1 = 0.

while ((i >= -1) ~~and~~ && (A[i] > key))

1st Condition 2nd Condition.

if these Condition ~~a~~ true, then swap.
a[i] with a[i+1] and decrement of i ← i-1
a[i+1] ← key // Correct position of key.

|3|9|5| 1|7|8|6|4|

after decrementing, i will point to -1.

∵ -1 ≯ -1 ∴ while loop won't be executed

program will Come to a[i+1] ← key.

a[-1+1] ← Key

a[0] ← Key

3 ← Key.

- Now J = 2 (i) (j)

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

∵ key = a[J] and i become j-1 = 2-1 = 1.

∴ key = a[2] = 5

∵ 1 > -1 and a[i] > 5

9 > 5 = true.

∴ Swap [a[i] with a[i+1].

|3|5|9|2|7|8|6|4|

decrement of i = i becomes 0.

0 > -1, Condition is true, and a[i] = a[0] = 3.

∴ a[0+1] ← key

3 ≯ 5 Condition true.

a[1] = ← key.

5 ← key.

3- Now j = 3.

∴

3	5	2	9	7	8	6	4

again $i = j - 1 = 1$.

Comparing key = 2 to i[1] = 5

5 > 2 and 1 > -1, Condition is true.

3	2	5	9	7	8	6	4

again $i = i - 1 = 0$

again Comparing key=2 to i[0] = 3

5 > 3 and 0 > -1, Condition is true.

Step-4

2	3	5	7	9	8	6	4

Now $i = i - 1 = 2$

Comparing i[2] ⊁ 7 or 5 ⊁ 7

∴ a[i] = a[i+1] = a[2+1] = 3.

Step-5 Repeate Process in step 3 and Step 4 until the array gets sorted. Completely. and finally we get

2	3	4	5	6	7	8	9

Algorithm for insection sort.

1) R initialise. i=0 and j = 1

2) set key = a[j] and i = j-1

3) set Condition ~~to~~ 1st → i > -1 and Condition.
   2nd → a[i] ≥ key.

4) if both Condition become true then swap a[i] with a[i+1] and decrement of i by 1.

5) if Condition become false, then a[i] = a[i+1]. and a[i+1] ← key.

6. Repeate Step 1, 2, 3, 4, 5 until i < n and step 2, 3, 4 until j < n-1.

4, 3, 5, 2, 9, 7, 1, 10, 8, 6.

a[10] = {4, 3, 5, 2, 9, 7, 1, 10, 8, 6}.

	0	1	2	3	4	5	6	7	8	9
J →	4	3	5	2	9	7	1	10	8	6

In this sorting method we would take an element as pivot element and would arrange such that the no. before or after that pivot element. get swapped by it until there comes an element greater than pivot element or less that pivot element from right to left.

1. take pivot = 6 and second pointer will be at 4.
here p = starting element, r = ending element.

0	1	2	3	4	5	6	7	8	9
4	3	5	2	9	7	1	10	8	6
↑p ↑j									↑r ↑x

if A[j] ≤ x ; then i = i + 1.
4 < 6.

then i = i + 1 and exchange a[i] ↔ a[j].

initially i = -1

Now $J = 2$ again $5 \not< 6$

then, $i = i+1 = 1+1 = 2$

$a[2] = a[2] = 5$

Now again.

$J = 3$, $2 < 6$

$i = 3$, $a[i] = a[3] = 2$

→ $J = J+1 = 4$, $9 \not< 6$

Now increment of j by 1 ⇒ $J = 5$

$i = 4$

0	1	2	3	4	5	6	7	8	~~9~~ 9
4	3	5	2	9	7	1	10	8	6
				↑ i	↑ J				

Now $J = 5$, $7 \not< 6$, Condition wrong.

no need to increment of i

$J = 6$, $1 < 6$ Condition is true

swap a[i] by a[J].

0	1	2	3	4	5	6	7	8	9
4	3	5	2	1	7	9	10	8	6
				↑ i		↑ J			

Now repeate the whole process for left and riqut sub array.

riqut sub array

9	10	8	7.

left subarray

4	3	5	2	1.

<Algorithm>.

```
pactitions <a, p, r>
  x = a[r]
   i = p-1
   for j ← p to r-1.
   {
      if <a[j] ≤ (x)
        { i = i+1.
      Exchange a[i] ⇔ a[i].
   }
   }(a [i+1] = a[r]
```

```
return i+1
Quick sort (a, p, r)
if (p < r)
 { q = pactition (a, p, r)
  quick sort (a, p, q-1)
 Quick sort (a, q+1, r).
 }
```

Explain the working of selection sort in following array with algorithm.

3, 5, 2, 9, 7, 1.

for. Selection sort we will take 3 variable. i, j, k respectively.

a[6] = {3, 5, 2, 9, 7, 1}.

i k	k j				
0	1	2	3	4	5
3	5	2	9	7	1.

initialising i by 0 and setting i=k.
  we get k=0
  j = i+1 <initialisation of j>

if a[J] < a[k], then k := j

∴

3	5	2	9	7	1

5 ≮ 3, Condition fails, J → J + 1 → 2

3	5	2	9	7	1
0	1	2	3	4	5
i, k		J			

Now a[J] < a[k] or 2 < 3 Condition true.

then. put k = j = 2.

Now, k ≠ i, then swap a[i] with a[k]

2	5	3	9	7	1
i		k	J		

Now, J → 3 and 9 ≮ 3 Condition fails.

then J → 4.

2	5	3	9	7	1
0	1	2	3	4	5
i		k		J	

Now, a[J] ≮ a[k], 7 ≮ 3 Condition fail, J → 5

Now, 1 < 3, Condition true k = 5

swap 1 with 2.

1	5	3	9	7	2
0	1	2	3	4	5

3. i = 1, J = 2, k = 1.

Now, for loop terminates here <inner>.

3. Repeat Step 2 and 3 until $i < 5$ After that we will get sorted list of array i.e.

1	2	3	5	7	9

<Algorithm>

Start

initialise $i = 0$ and $j = i + 1$.

take another variable $k = i$

if $a[j] < a[k]$ then store the value of j into k and then k will not be equal to, i. Swap $a[i]$ with $a[k]$.

if condition becomes false then only increment the value of j.

j will run until $j < n$.

repeat the steps (iv), (v) and (vi) until $i < n-1$.

after that we will get sorted array

logic.

```
        for (i=0 ; i<n-1; i++)
        {
```