UNIT - I

2/11/21 Tuesday

→ Data structure means organizing the data in the computer memory in efficient manner

(or)

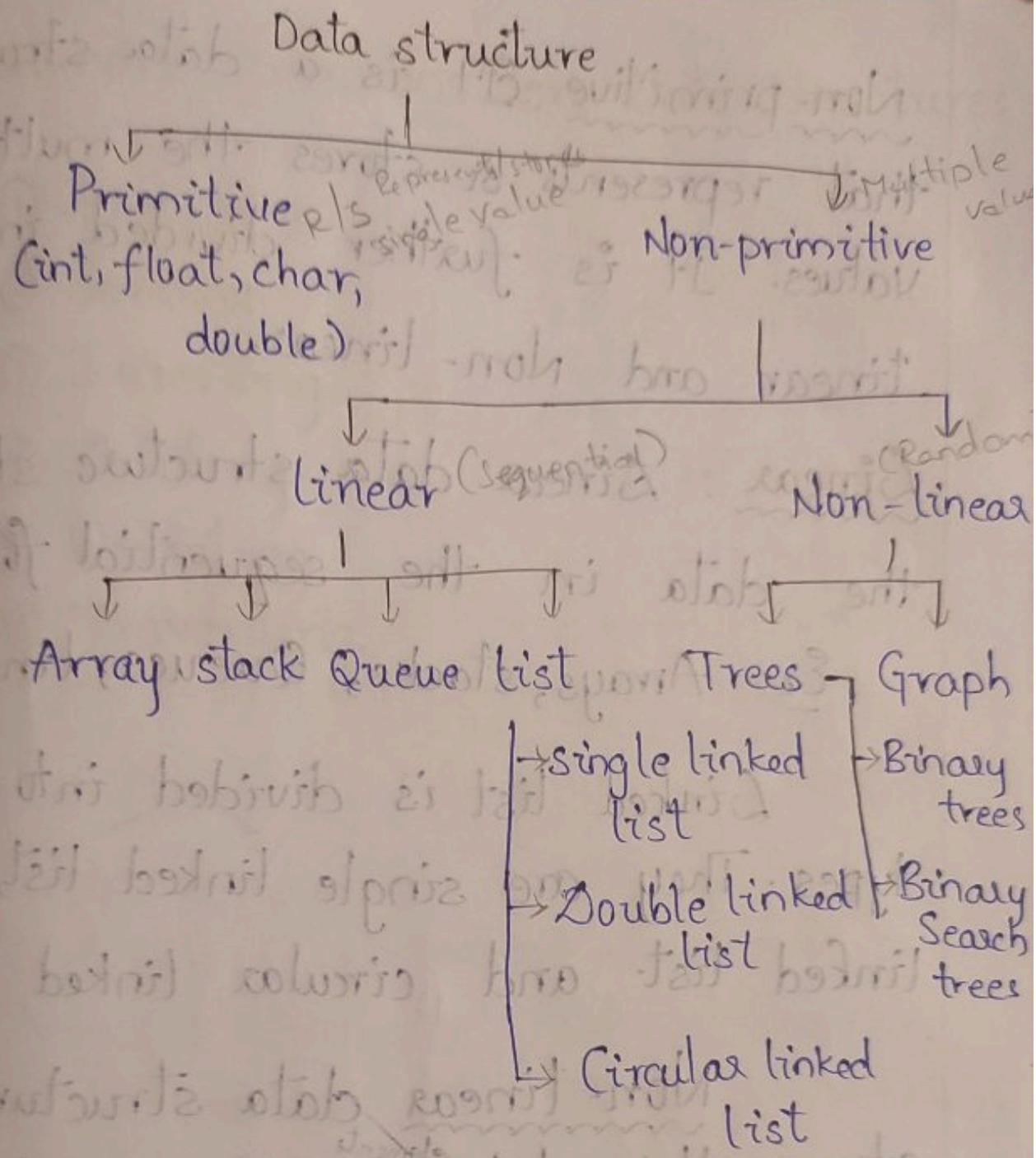
→ Data structure refers to storing the data in an organized way.

Ex: Stacks, queues, trees, graphs, lists

→ If the data is not organised properly then the operations are performed on that data structure are not effective.

** Types of Data structures

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA



Data structures are divided into two types ① Primitive ② Non-primitive

Primitive :- Primitive data structure represents or stores single value.

Ex: int, float, char, double.

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

Non-primitive: It is a data structure that represents & stores the multiple values. It is further divided into linear and Non-linear.

Linear: Linear data structure stores the data in the sequential format

Ex: Arrays, stacks, Queues, linked list.

Linked list is divided into three types. They are single linked list, double linked list and circular linked list.

Non-linear data structures stores the data elements in the random order in the computer memory. Non-linear data structure is divided into two categories. They are trees, Graphs.

Trees are further classified into Binary tree and Binary Searching tree

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

Common operations on Data Structures

1. Searching
2. Sorting
3. Inserting
4. Deleting
5. Updating
6. Traversing.

MI NOTE 9 PRO MAX
UAD CAMERA

Linear	Non-Linear
① In linear, the elements are stored in sequential order.	① In Non-linear, the elements are stored in random order.
② In this, linear, single level is involved.	② In Non-linear, multiple levels are involved.
③ Here, memory is not utilized efficiently.	③ Here, the memory is utilized efficiently.
④ Easy to implement	④ Difficult to implement
⑤ Time complexity of linear data structure increases with increase in Input size.	⑤ Time complexity of Non-linear data structure remains same with increase in Input size.
⑥ Mainly used for developing the softwares	⑥ Mainly used in artificial Intelligence and Image processing.
⑦ Example : Stack, Queue, Array, list.	⑦ Examples : Trees, Graphs.

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

03

Performance Analysis

The performance of any algorithm or program is measured based on the two factors.

1. Space Complexity
2. Time Complexity.

Space Complexity: It is the total amount of Computer memory required by algorithm to complete its execution.

Ex:

```
int square(int a)
{
    return a*a;
}
```

→ The above code requires 2 bytes of memory to store "a" and another 2 bytes for return value (if it is a 32 bit machine)

Time complexity: It is the total ~~time~~ amount of time taken by algorithm to complete its execution.

Ex:

```
int square(int a)
{
    return a*a;
}
```

EDMI NOTE 9 PRO MAX
QUAD CAMERA

In the above code it requires 1 unit of time for "a" and another one unit of time for return value (Totally it takes 2 units of time)

Analysis of Algorithm

There are 3 cases to analyse an algorithm.

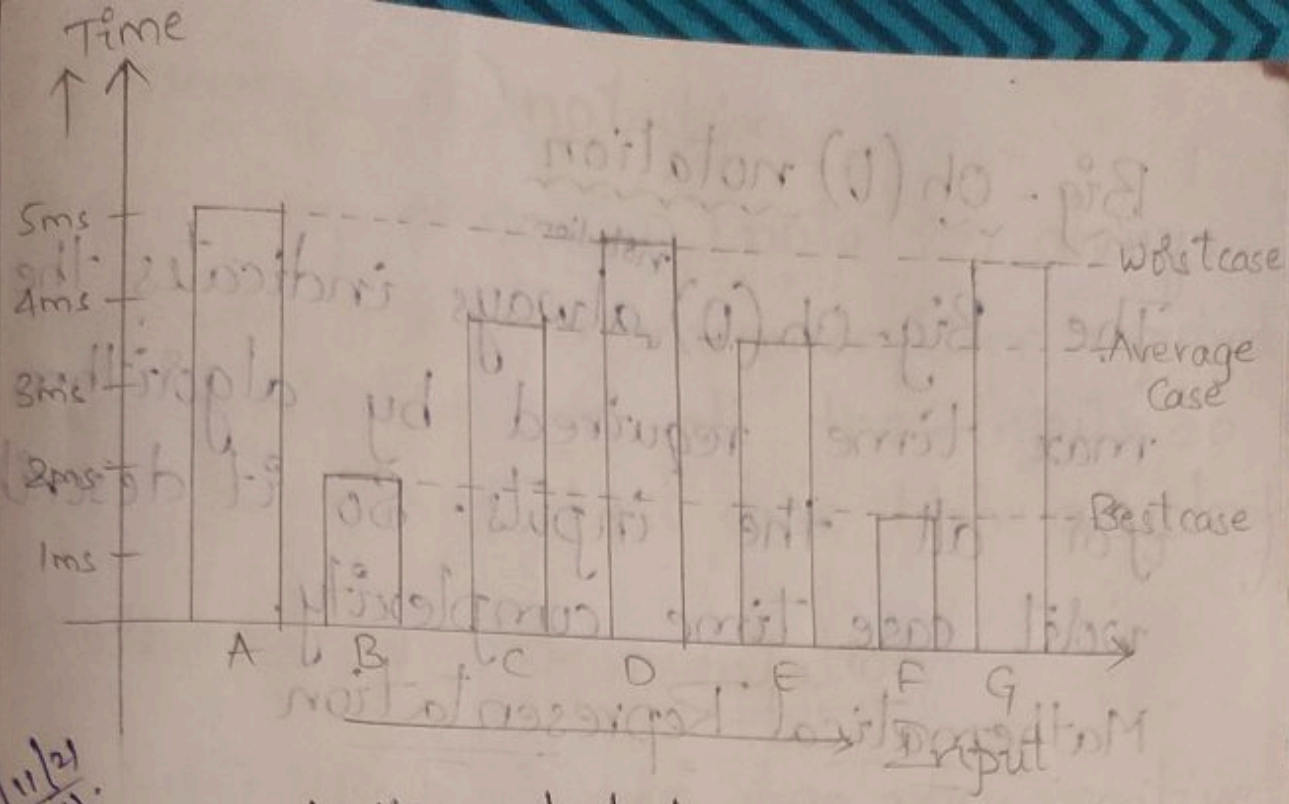
1) Best Case
2) Average Case
3) Worst Case

Best Case : The minimum time required by the algorithm to complete its execution (run fastest among all the inputs)

Average time : The average time required by the algorithm to complete its execution.

Worst Case : The maximum time required by the algorithm to complete its execution (run slowest among all the inputs)

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA



05/11/21 Friday.

Asymptotic Notations

Asymptotic Notations are Notations which are used to describe the behaviour of time & space complexity.

(or)

Asymptotic Notations are of an algorithm is a mathematical representation of its complexity.

→ Big - Oh(O) notation

→ Big - omega(Ω) notation

→ Big - Theta(Θ) notation

→ Big - Oh(o) notation

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

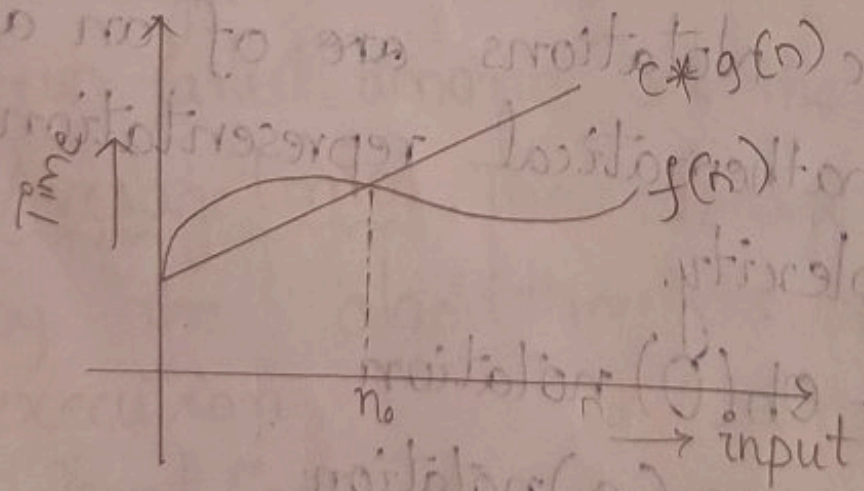
Big-Oh (O) notation

The Big-Oh (O) notation always indicates the max time required by algorithm for all the inputs. So it describes worst case time complexity.

Mathematical Representation

let $f(n)$ & $g(n)$ are two non-negative functions then $f(n) = O(g(n))$, if there exists two constants c, n_0 such that $f(n) \leq c * g(n)$, $n > n_0$, $c > 0, n \geq 1$

Graphical Representation



REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

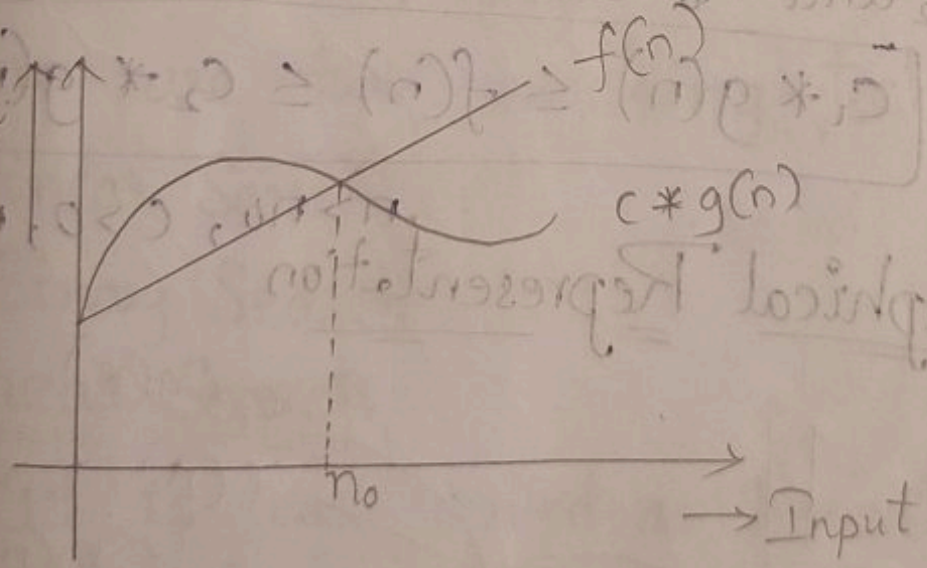
Big-omega (Ω) notation

The Big-omega (Ω) notation always indicates the minimum time required by algorithm for all the inputs. So it describes best case time complexity

Mathematical Representation

Let $f(n)$ & $g(n)$ are two non-negative functions then $f(n) = \Omega(g(n))$, if there exist two constants c, n_0 such that $f(n) \geq c * g(n)$, $n > n_0$, $c > 0$, $n \geq 1$

Graphical Representation



REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

Big - Theta(θ) notation

The Big-Theta(θ) notation always indicates the average time required by an algorithm for all the inputs. So it describes average time complexity.

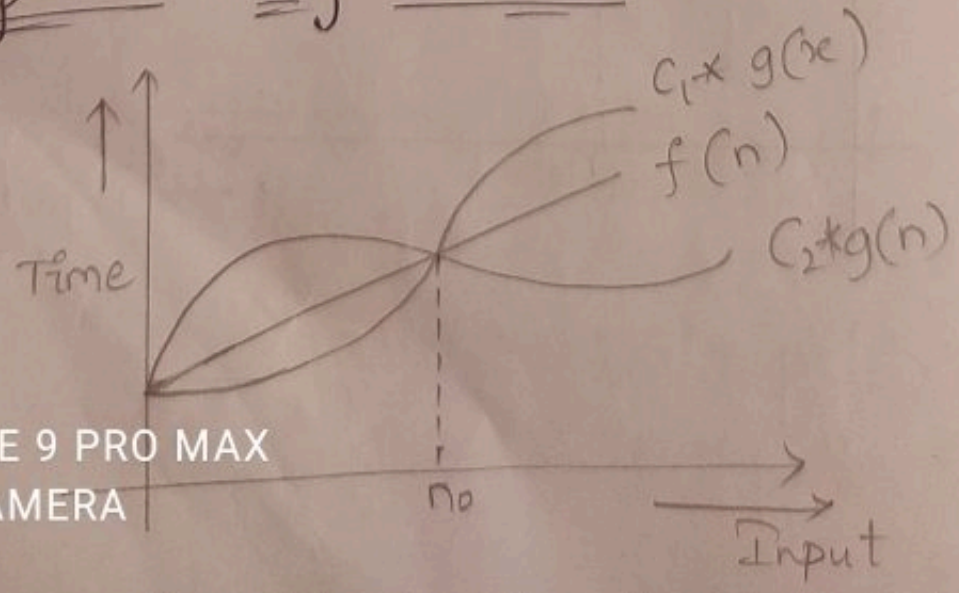
Mathematical Representation

Let $f(n)$, and $g(n)$ are two non-negative function then $f(n) = \theta(g(n))$ if there exist three +ve constants c_1, c_2 and n_0 such that

$$\boxed{c_1 * g(n) \le f(n) \le c_2 * g(n)}$$

$n > n_0,\ c > 0,\ n \ge 1$

Graphical Representation



REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

small-oh(o) notation

let $f(n)$ and $g(n)$ are two non-negative functions; then $f(n) = o(g(n))$ such that;

$$\boxed{\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0}$$

09/11/21 ~~Monday~~ Tuesday

Searching: Searching is the process of finding particular element from a group of element (or) from a given array

There are two searching techniques

1) linear Search
2) Binary Search

linear Search

→ It is also called as Sequential Search

→ For performing linear Search, takes as array as input along with a key which is to be Searched.

It works by comparing the "key value

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

with each and every element of array until a match is found.

→ If the match is found then it returns the position of the key element.

→ If the match is not found then we declare that key was not found.

NOTE:

In this search there is no seperate logic just we are comparing key value with each and every element.

Working

```
for(i=0; i<n; i++)
{
    if(key==a[i])
        printf(" key is found at %d location",i);
        break;
}
if(i==n)
printf(" key was not found");
```

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

From the above program

Ex:- a[i] = {9, 7, 15, 32, 5, 96, 109} Key = 96

for i = 0 ⇒ a[0] = 9

if 96 == 9 ×

for i = 1 ⇒ a[1] = 7

96 == 7 ×

for i = 2 ⇒ a[2] = 15

96 == 15 ×

for i = 3 ⇒ a[3] = 32

96 == 32 ×

for i = 4 ⇒ a[4] = 5

96 == 5 ×

for i = 5 ⇒ a[5] = 96

96 == 96 ✓

Key is found at 5 location

Advantages of linear search

⇒ Easy to implement & understand

⇒ Simple Implementation

Disadvantages

⇒ Slower algorithm

⇒ inefficient when compared with Binary Search

Binary Search

→ It is a faster algorithm

→ In this Searching we need to sort the elements before

→ Here the Searching is based on "mid" value

where $mid = \frac{low + high}{2}$

→ For performing binary search, take an sorted array along with a key value which

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

is to be searched.

* Here we are comparing the key value with the mid value.
* Here, there are three possible cases

1) if(key == A[mid])
   printf("key found")
2) else if (key < A[mid])
   High = mid-1
3) else (key > A[mid])
   low = mid+1

* If the match is found, then it returns the position of the element
* If the match is not found, then we declare that key was not found.

Ex: Perform Binary Search for following elements

{20, 50, 100, 200, 250, 320, 450, 590},
key = 450.

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

$mid = \frac{low + high}{2} = \frac{0+7}{2} = \frac{7}{2} = 3.5$

if (key == A[mid])

450 == 200 ×

if (key < A[mid])

key < 200 ×

450 < 200

if (key > A[mid])

450 > 200 ✓

low = mid + 1

low = 3 + 1 = 4

$mid = \frac{4+7}{2} = \frac{11}{2} = 5.5 = 5$

if (key == A[mid])

450 == 320

if (key < A[mid])

450 < 320 ×

if (key > A[mid])

450 > 320 ✓

low = mid + 1 => 5 + 1 = 6

$mid = \frac{6+7}{2} = 6.5 = 6$

if (key == A[mid])

450 = 450

found at 6th position.

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

Differences b/w linear & Binary Search

Linear	Binary
* In this search all the elements are need not to be sorted	* In this search all the elements should be sorted
* It is also called as Sequential search	* It is also called as logorithmic Search
* It is less efficient	* It is more efficient
* It is less complex	* It is more complex
* Takes more time	* Takes less time
* Best case is to find the key in 1st position	* Best case is to find the key in mid position
* Time complexity is O(N)	* Time complexity is O(logN)
* It is simple method for Searching.	* It is efficient method for Searching.

S C C

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

10/11
Wednesday

Fibonoci Search

→ It is a searching Technique that depends on the fibonoci numbers and based on the "Divide and Conquer" rule

→ It is a comparision basic technique uses fibonoci numbers to search an element in a sorted array. If the key is found then it returns its index otherwise it returns -1

Algorithm

Step-1 :- First we need to find F(k) which is the kth fibonoci number which is greater than or Equal to the size of the array.

Step-2 :- If F(k) = 0, then we need to stop and print the message Key was not found.

Step-3 :- Set variable Offset = -1

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

Step-4 :- Then we need to set

$i = \min(offset + F(k-2), n-1)$

Step-5 :- We will check

case-(i) :- if key == A[i] then return i and print key was found.

case-(ii) :- else if key > A[i] then k = k-1, offset = i and need to repeat steps 4 and 5

case-(iii) :- else if key < A[i] then k = k-2, and repeat the steps 4 and 5

Perform Fibonaci Search for following elements 50, 58, 64, 74, 80, 100, 120

key = 100

$i = \min(offset + F(k-2), n-1)$

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

F(K-2)	F(K-1)	F(K)	offset	i	A[i]	Action
3	5	8	-1	2	64	100>64, Fib one down
2	3	5	2	4	80	100>80, Fib one down
1	2	3	4	5	100	100=100 Key found.

F(K-2)	F(K-1)	F(K)	offset	i	A[i]	Action
3	5	8	-1	2	64	100>64 Fib one down
2	3	5	2	4	80	100>80 Fib one down
1	2	3	4	5	100	100=100 Key found

11/11/21

① As n is 7 then F(k) is greater than & equal to 7

$f(k) = 8$

$F(k-1) = 5$

$F(k-2) = 3$

offset = -1 initially

$i = \min(\text{offset} + F(k-2), n-1)$

$= \min(1+3, 6)$

$= \min(2, 6) = 2$

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

$A[2] = 64$

$key = 100$

$100 > 64$

(2) $k = k - 1$

$k = 5$

$k - 1 = 3$

$k - 2 = 2$

$offset = i$

$offset = 2$

$i = \min[offset + F(k-2), n-1]$

$i = \min[2 + 2, 6)$

$i = \min(4, 6)$

$i = 4$

$A[4] = 80$

(3) $K = k - 1$

$k = 3$

$k - 1 = 2$

$k - 2 = 1$

$i = 0$

$offset = i$

$offset = 4$

$i = \min[offset + F(k-2), n-1]$

$i = \min[4 + 1, 6)$

$i = \min[5, 6]$

$i = 5$

$A(5) = 100$

MI NOTE 9 PRO MAX
UAD CAMERA

②

0	1	2	3	4	5	6	7	8	9	10
10	20	30	40	45	50	70	80	85	90	100

$n = 11$; key = 85

F(k-2)	F(k-1)	F(k)	offset	i	A[i]	Action
5	8	13	-1	4	45	85>45 Fib one down
3	5	8	4	7	80	~~80>45~~ 85>80 Fib one down
2	3	5	7	9	90	85<90 Fib two down
1	1	2	7	8	85	85=85 key found

key was found at location 8.

① $F(k) \geq n$ [$n = 11$]

$F(k) = 13$

$F(k-1) = 8$

$F(k-2) = 5$

Initially

offset = -1

$i = \min[\text{offset} + F(k-2), n-1]$

$i = \min[-1+5, n-1]$

$i = 4$

② offset = i

$F(k) = k-1$

$F(k) = 8$

$F(k-1) = 5$

$F(k-2) = 3$

$i = \min[\text{offset} + F(k-2), n-1]$

$= \min[4+3, 10]$

$= \min[7, 10]$

$= 7$

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

③ $F(k) = k-1$

$F(k) = 5$

$F(k-1) = 3$

$F(k-2) = 2$

$i = \min[offset + F(k-2), n-1]$

$i = \min[7+2, 10]$

$i = 9$

④ offset = 7

$k = 2$

$F(k) = 2$

$F(k-1) = 1$

$F(k-2) = 1$

$i = \min[offset + F(k-2), n-1]$

$i = \min[7+1, 10]$

$i = 8$

$A(8) = 85$

15/11/21

Sorting: Arranging the elements in specific order

1) Bubble sort

Sorting: Arranging the elements in specific order (Ascending order)

→ There are 6 sorting techniques

1) Bubble sort
2) Selection sort
3) Insertion sort

DMI NOTE 9 PRO MAX
QUAD CAMERA

4) Radix sort
5) Merge sort
6) quick sort

1) Bubble sort :-

It is one of the sorting Techniques available in ~~sorting~~ data structures.

→ It is a simple method that sorts the array elements by repeatedly moving the largest element to the highest index position of the array. It is a comparision based algorithm in which each pair of adjacent elements are compared and these elements are swapped if they are not in order.

Working principle

→ Bubble sort will always starts by comparing the first element of array with second element of array. If first element is greater than the second element then swap first & Second elements

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

then compare second and third element and do the same process until all the elements are sorted.

→ It is known as Bubble sort because, with every complete iteration the largest element is bubble up towards the last place just like water bubbles raises up to the water surface.

Logic

```
int i, j;
for(i=0; i<n; i++)
{
    for(j=0; j<n-1; j++)
    {
        if(a[j] > a[j+1])
        {
            temp = a[j];
            a[j] = a[j+1];
            a[j+1] = temp;
        }
    }
}
```

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

Ex:

A[ ] = 9 27 35 12 6

P-1

pass-1 9 27 35 12 6

No

No need to swap

pass-2 9 27 35 12 6

No need to swap

pass-3 9 27 35 12 6

35 > 12

swapping

After swap

9 27 12 35 6

pass-4 9 27 12 35 6

35 > 6

After swap

9 27 12 6 35

P-2

9 27 12 6 35

pass-1

no need to swap

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

pass-2 9 27 12 6 35

swap

After swap

9 12 27 6 35

pass-3

After swap 9 12 6 27 $\boxed{35}$

pass-4

no need to swap

9 12 6 $\boxed{27}$ $\boxed{35}$

Iteration-3

pass-1 no need to swap

9 12 6 $\boxed{27}$ $\boxed{35}$

12>6

swap

After swap

9 6 $\boxed{12}$ $\boxed{27}$ $\boxed{35}$

Iteration 4 – 9 6 12 27 35

After swap 6 $\boxed{9}$ $\boxed{12}$ $\boxed{27}$ $\boxed{35}$

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

2) Selection sort : In Selection sort, the smallest value among unsorted elements of the array is selected in every iteration and placed to its appropriate position. In this algorithm the given array is divided into two parts

① Sorted part ② unsorted part.

Initially sorted part of the array is empty and unsorted part is the given array.

In Selection sort, the first ~~shorte~~ smallest element is selected and placed at the 1st position then Second smallest element is selected and placed at 2nd position. This process is continued untill the entire array is sorted.

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

Logic

```
for(i=0; i<n; i++)
{
    min = i;
    for(j=i+1; j<n; j++)
    {
        if(a[j] < a[min])
            min = j;
    }
    if(min != i)
    {
        temp = a[i]
        a[i] = a[min]
        a[min] = temp
    }
}
```

A[0] = 5 1 10 6 8

0 1 2 3 4

15

15

Iteration-1

i = 0

min = 0

j = 1

a[1] < a[0]

1 < 5 ✓

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

min = 1

j = 2

a[2] < a[1]

10 < 1 x

j = 3

a[3] < a[1]

6 < 1 x

j = 4

a[4] < a[1]

8 < 1 x

j = 5 loop breaks

temp = a[0]

temp = 5

a[0] = 1

a[1] = 5

0	1	2	3	4
1	5	10	6	8

Iteration 2

i = 1

min = 1

j = 2

a[2] < a[1]

10 < 5 x

j = 3

a[3] < a[1]

6 < 5 x

j = 4

a[4] < a[1]

8 < 5 x

j = 5 loop breaks.

temp = a[1]

temp = 5

a[1] = 5

a[1] = 5

1 5 10 6 8

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

0 1 2 3 4
15 10 6 8

Itération-3

i = 2 , min = 2

j = 3

a[3] < a[2]

6 < 10 ✓

min = 3

j = 4

a[4] < a[3]

8 < 6 ✗

j = 5 loop breaks

temp = a[2]

temp = 10

a[2] = 6

a[3] = 10

1 5 6 10 8

Iteration-4

i = 3, min = 3

j = 4

a[4] < a[3]

8 < 10 ✓

min = 4

j = 5 loop breaks

temp = 10

a[3] = a[4]

a[3] = 8

a[4] = 10

Sorted array is 1 5 6 8 10

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

18/... Thursday

Insertion sort : In this sort every iteration moves an element from unsorted portion to sorted portion untill all the elements are sorted in the list

working procedure

Step-1 : Assume that the first element in the list is in sorted portion & all the remaining elements are in unsorted portion.

Step-2 : Take first element from the unsorted portion and insert that element into the sorted portion in the order specified.

Step-3 : Repeat the above process untill all the elements from unsorted portion moved into sorted portion

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

logic

```
for(i=1;i<n;i++)
{
    temp = a[i]
    j = i-1;
    while((temp<a[j]) && (j>=0))
    {
        a[j+1] = a[j];
        j=-;
    }
    a[j+1] = temp;
}
```

Array

A[] = 67 3 59 21 8

Iteration-1

i = 1

temp = 3, j = 0;

(3<67) ✓ && (j>=0) ✓

a[1] = a[0]

a[1] = 67

j = -1 while loop breaks

a[0] = 3

3 67 59 21 8

NOTE 9 PRO MAX
D CAMERA

Iteration-2

i = 2

temp = 59

j = 1

while(59 < 67) && (j >= 0) ✓

a[2] = a[1]

a[2] = 67

j = 0

while (59 < 3) × loop breaks

a[1] = 59

3 59 67 27 8

Iteration-3

i = 3, temp = 27, j = 2

while (27 < 67) && (j >= 0) ✓

a[3] = a[2]

a[3] = 67

j = 1

while(27 < 59) && (j >= 0)

a[2] = a[1]

a[2] = 59 ⇒ j = 0

while (27 < 3) ×

a[1] = 27

3 27 59 67 8

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

Iteration-4

i = 4, temp = 8, j = 3

while (8 < 27) && (3 > 0)

a[4] = a[3]

a[4] = 67

j = 2

while (8 < 59) && (2 > 0) ✓

a[3] = a[2]

a[3] = 59

j = 1

while (8 < 27) && (1 > 0) ✓

a[2] = a[1]

a[2] = 27

j = 0

while (8 < 3) × loop breaks

a[1] = 8

Sorted array ⇒ 3 8 27 59 67

EDMI NOTE 9 PRO MAX
QUAD CAMERA

Merge Sort

It is one of the most popular and efficient sorting algorithm and it follows "divide and conquer" principle

divide : divide the given array into two sub arrays.

conquer : recursively sort these two sub arrays.

combine / Merge : Combine these two sub arrays into a single sorted array.

Working principle :-

First of all merge sort divides given array into two sub arrays based on the mid value. After that we can recursively sort these two sub arrays. Here these two sub-arrays are again divided into another two sub arrays untill the array cannot be divided further. After we can merge these two sorted sub-arrays into a single sorted array.

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

Algorithm :

```
mergesort(Array, low, high)
{
    if(low < high)
    {
        mid = ( (low + high) / 2 );
        mergesort(Array, low, mid);
        mergesort(Array, mid+1, high);
        merge(Array, low, mid, high);
    }
}
```

Ex:

Sort the given elements

25 15 9 36 77 92 64 by using merge sort

	0	1	2	3	4	5	6
A[] =	25	15	9	36	77	92	64

$n = 7$

low = 0

high = 6

$mid = \frac{low + high}{2} = 3$

A[3] = 36

low to mid ⇒ 25 15 9 36 ⇒

mid+1 to high ⇒ 77 92 64

OMI NOTE 9 PRO MAX
QUAD CAMERA

Algorithm :

```
mergesort(Array, low, high)
{
    if(low < high)
    {
        mid = ( (low + high) / 2 );
        mergesort(Array, low, mid);
        mergesort(Array, mid+1, high);
        merge(Array, low, mid, high);
    }
}
```

Ex:

Sort the given elements

25 15 9 36 77 92 64 by using merge sort

	0	1	2	3	4	5	6
A[] =	25	15	9	36	77	92	64

$n = 7$

low = 0

high = 6

$mid = \frac{low + high}{2} = 3$

A[3] = 36

low to mid ⇒ 25 15 9 36 ⇒

mid+1 to high ⇒ 77 92 64

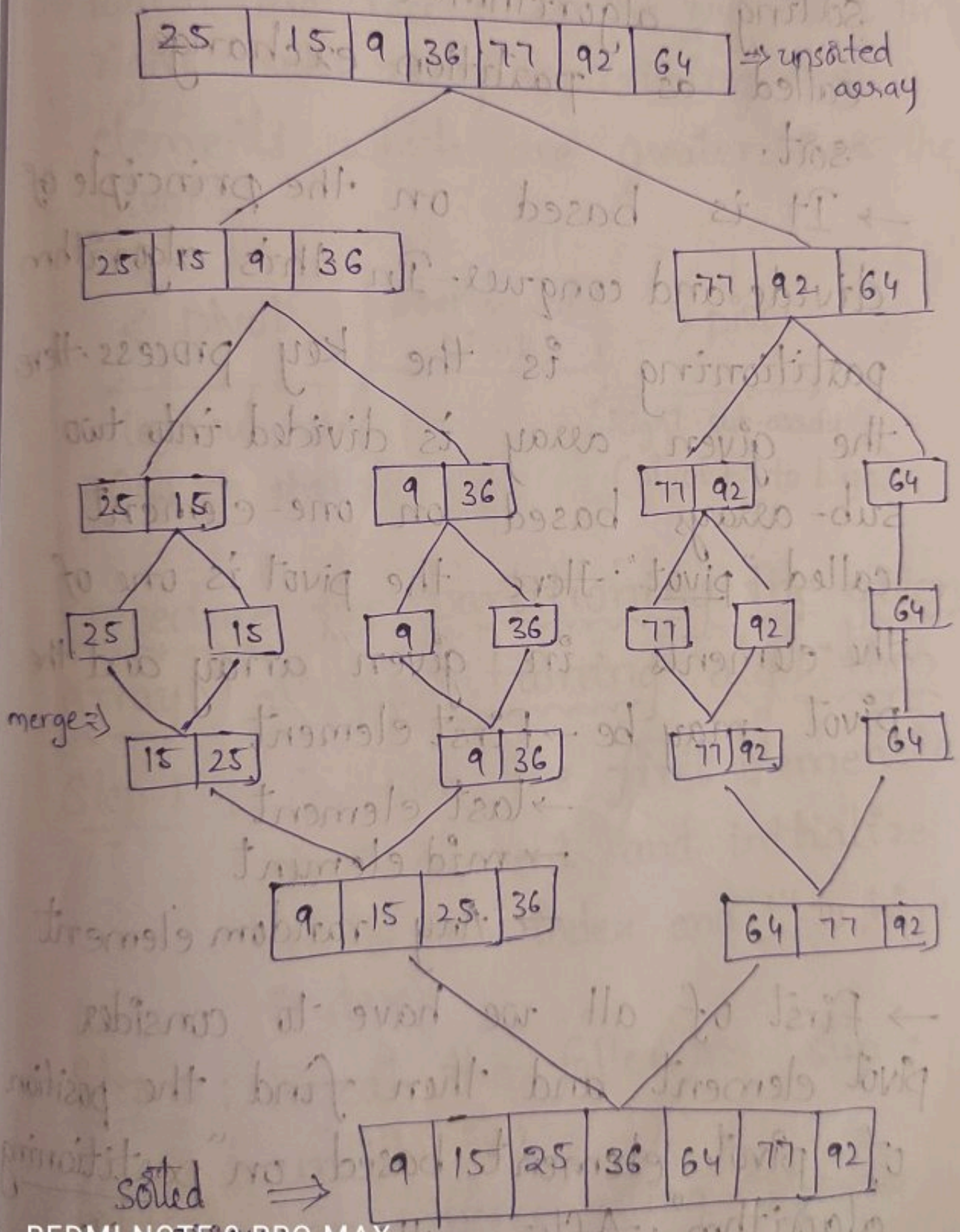
OMI NOTE 9 PRO MAX
QUAD CAMERA

25 15 9 36 ⟹ further divided [25 | 15] & [9 | 36]

low to mid mid+1 to high

77 92 64 ⟹ [77 | 92] & [64]

low to mid mid+1 to high



REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

19/11/

Quick Sort

→ It is one of the most popular sorting algorithm. and It is also called as partition exchange sort.

→ It is based on the principle of divide and conquer. In this algorithm partitioning is the key process. Here the given array is divided into two sub-arrays based on one-element called "pivot". Here the pivot is one of the elements in given array and the pivot may be → First element
→ last element
→ mid element
→ Any random element

→ First of all we have to consider pivot element and then find the position of pivot element based on "partitioning algorithm". After getting the position

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

of pivot. The given array is divided into two sub-arrays called as left sub-array and right sub-array. Here the left sub array contains the elements which are less than the pivot and the right sub array containing the elements which are greater than the pivot.

< pivot | position of pivot | > pivot

left sub array (low to pivot-1) | Right sub-array (pivot+1 to high)

Procedure for partitioning the given array or partitioning algorithm

Step-1 : Consider the first element as pivot element and initialize "i" to low index and "j" to high index

Step-2 : Repeat the following step until i<j

a) keep on incrementing "i" value while a[i] ≤ pivot

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

b) keep on decrementing "j" while a[j] > pivot

c) If i<j then swap(a[i], a[j])

Else

Step-3 :- ~~If i≥j then~~ swap(a[j], a[pivot])

Here "j" is the position of the pivot

After getting the position of the pivot there are two partitions

⟹ partition 1 → low to pivot - 1

⟹ partition 2 → pivot + 1 to high

Then apply same procedure for partition 1 and partition-2 untill all the array elements are sorted.

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

Algorithm

```
Quicksort (Array, low, high)
{
    if (low < high)
    {
        pivot = partition(Array, low, high)
        Quicksort (Array, low, pivot-1) - partition1
        Quicksort (Array, pivot+1, high) →
                                          partition-2
    }
}
```

Ex: Consider list of Elements

0	1	2	3	4	5	6	7	8
54	26	93	17	77	31	44	55	20

n=9

Step-1 : First of all consider 54 as pivot and assign i value with low index and j value with high index

0	1	2	3	4	5	6	7	8
54	26	93	17	77	31	44	55	20
pivot/i								j

check a[i] ≤ a[pivot]

54 ≤ 54 ✓

i++

i = 1

a[i] ≤ a[pivot]

26 < 54 ✓ ⇒ i++ ⇒ i=2

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

a[2] < pivot

93 < 54 ✗

a[j] ≥ a[pivot]

a[8] ≥ a[pivot]

20 ≥ 54 ✗

i = 2, j = 8

2 < 8 (i < j)

swap a[i] & a[j]

0	1	2	3	4	5	6	7	8
54	26	20	17	77	31	44	55	93

i = 2,

a[2] < a[pivot]

20 < 54 ✓

i++

i = 3

a[3] < a[pivot]

17 < 54 ✓

i = 4

a[4] < a[pivot]

77 < 54 ✗

a[j] ≥ a[pivot]

a[8] ≥ 54

93 ≥ 54

j = 7

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

$a[j] \geq a[pivot]$

$55 \geq 54$ ✓

$j = 6$

if $(i < j)$

$4 < 6$ ✓

swap $(a[i]$ and $a[j]$

0	1	2	3	4	5	6	7	8
54	26	20	17	44	31	77	55	93
				i		j		

$i = 4$, $j = 6$

$a[i] \leq a[pivot]$

$a[4] \leq a[pivot]$

$44 \leq 54$ ✓

$i = 5$

$a[5] \leq a[pivot]$

$31 \leq 54$ ✓

$i = 6$

$a[6] \leq a[pivot]$

$77 \leq 54$ ✗

$a[6] \geq a[pivot]$

$77 \geq 54$ ✓

$j = 5$

$a[5] \geq a[pivot]$

$31 \geq 54$ ✗

if $(i < j)$ $6 < 5$ ✗

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

swap (a[j], a[pivot])

31 26 20 17 44 $\boxed{54}$ 77 55 93

partition-1 partition-2

Consider partition-1

0	1	2	3	4
31	26	20	17	44
i/pivot				j

i = 0,

a[i] ≤ a[pivot]

31 ≤ 31 ✓

i = 1

a[1] ≤ a[pivot]

26 ≤ 31 ✓

i = 2

a[2] ≤ a[pivot]

20 ≤ 31 ✓

i = 3

a[3] ≤ a[pivot]

17 ≤ 31 ✓

i = 4

a[4] ≥ a[pivot]

44 ≥ 31 ✓

j = 3

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

$a[3] \geq a[pivot]$

$17 \geq 31$ ×

$i = 4, j = 3$

$i < j = \times$

swap(a[j], a[pivot])

17 26 20 $\boxed{31}$ 44

← partition 1 →

consider partition - 1

0	1	2
17	26	20
i/pivot		j

$i = 0$

$a[i] \leq a[pivot]$

$17 \leq 17$ ✓

$i = 1$

$a[i] \leq a[pivot]$

$26 \leq 17$ ×

$a[j] \geq a[pivot]$

$20 \geq 17$ ✓

$j = 1$

$a[i] \geq a[pivot]$

$26 \geq 17$ ✓

$j = 0$

$i < j$ swap(a[j], a[pivot])

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

$\boxed{17}$ 26 20

← partition-1 →

Consider partition-1

26 20

i/pivot j

$a[i] \le a[pivot]$

$26 \le 26$ ✓

$i = 1$

$a[j] \ge a[pivot]$

$20 \ge 26$ ×

$j = 1$

$(i < j)$ ×

swap (a[j], pivot)

20 26

17 20 26

17 20 26 31 44

Consider partion-2

77 55 93

i/pivot j

$a[i] \le a[pivot]$

$77 \le 77$ ✓

$i = 1$

$a[i] \le a[pivot]$

$55 < 77$ ✓

DMI NOTE 9 PRO MAX
QUAD CAMERA

Ex:

$i = 2$

$a[2] \le a[pivot]$

$93 \le 77$ ✗

$a[j] \ge a[pivot]$

$93 \ge 77$ ✓

$j = 1$

$a[j] \ge a[pivot]$

$55 \ge 77$ ✗

$i = 2, j = 1$

$(i < j)$ false

swap $(a[j], a[pivot])$

55 77 93

The sorted array is

17 20 26 31 44 55 77 93

Ex:

30 20 70 10 60 40 50

Step-1 : Consider First element 30 as pivot and assign i value with low index and j value with high index.

0	1	2	3	4	5	6
30	20	70	10	60	40	50
i/pivot						j

check $a[i] \le a[pivot]$

$30 \le 30$ ✓

$i = 1$

$a[1] \le a[pivot]$

$20 < 30$ ✓ $i = 2$

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

$a[2] \le a[pivot]$

$70 \le 30$ ✗

$a[j] \ge a[pivot]$

$50 \ge 30$ ✓

$j = 5$

$a[5] \ge a[pivot]$

$40 \ge 30$ ✓

$j = 4$

$a[4] \ge a[pivot]$

$60 \ge 30$ ✓

$j = 3 \Rightarrow a[3] \ge a[pivot] \Rightarrow 10 \ge 30$ ✗

$i = 2, j = 3$

$(i < j)$ ✓ true

swap $(a[i], a[j])$

0	1	2	3	4	5	6
30	20	10	70	60	40	50

~~pivot~~

$i = 2, j = 3$

$a[i] \le a[pivot]$

$10 \le 30$ ✓

$i = 3$

$a[3] \le a[pivot]$

$70 \le 30$ ✗

$a[j] \ge a[pivot]$

$70 \ge 30$ ✓

$j = 2$

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

$a[2] \geq a[pivot]$

$10 \geq 30$ ×

$i = 3, j = 2$

$i < j$ × swap(a[j], a[pivot])

10 20 [30] 70 60 40 50

← partition-1 → ← partition-2 →

partition-1 → sorted.

Consider partition-2

0	1	2	3
70	60	40	50
i/pivot			j

$a[i] \leq a[pivot]$

$70 \leq 70$ ✓

$i = 1$

$a[i] \leq a[pivot]$

$60 \leq 70$ ✓

$i = 2$

$a[2] \leq a[pivot]$

$40 \leq 70$ ✓

$i = 3$

$a[j] \geq a[pivot]$

$50 \geq 70$ ×

$j = 3$

$i < j$ × swap(a[j], a[pivot])

30 20 40 [70]

← partition-1 →

MI NOTE 9 PRO MAX
QUAD CAMERA

consider partition-1

0	1	2
50	60	40
i/pivot		j

$a[i] \le a[pivot]$

$50 \le 50$ ✓

$i = 1$

$a[i] \le a[pivot]$

$60 \le 50$ x

$a[j] \ge a[pivot]$

$40 \ge 50$ x

$i = 1, j = 2$

$i < j$ swap $(a[i]\ a[j])$

0	1	2
50	40	60

$a[i] \le a[pivot]$

$40 \le 50$ ✓

$i = 2$

$a[j] \ge a[pivot]$

$60 \ge 50$ ✓

$j = 1$

$a[j] \ge a[pivot]$

$40 \ge 50$ x

$i = 2, j = 1$

NOTE 9 PRO MAX
D CAMERA

$(i<j) \times$ false

swap a[j], a[pivot]

40 50 60

40 50 60 70

10 20 30 40 50 60 70

sorted array is 10 20 30 40 50 60 70

MI NOTE 9 PRO MAX
UAD CAMERA

23/11/21
Tuesday

Radix Sort : In Radix sort algorithm a list of integer numbers will be sorted based on the digits of individual numbers. Here the sorting is performed from "least significant Digit" to "most significant Digit"

→ It is also known as Bucket Sort. Here the no. of passes depends on the length of the number having maximum no. of digits.

Working procedure

Step-1:- Define 10 Queues each representing a bucket for each digit from 0 to 9.

Step-2: Consider (L.S.D) least significant Digit of each other number in list which is to be sorted.

Step-3:- Insert Each number into their respective queues based on the LSD

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

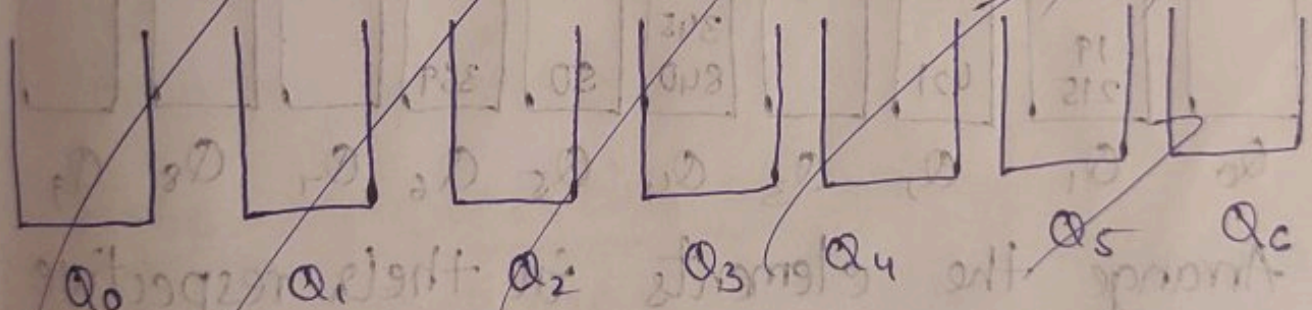
Step-4: Group all the numbers from Q_0 to Q_9 (Queue 0 to Queue 9) in the order they have inserted into their respective Queues.

Step-5: Repeat from step-3 based on next LSD

Step-6: The above process continues until all the numbers are sorted.

Elements

19 50 345 215 421 840 369

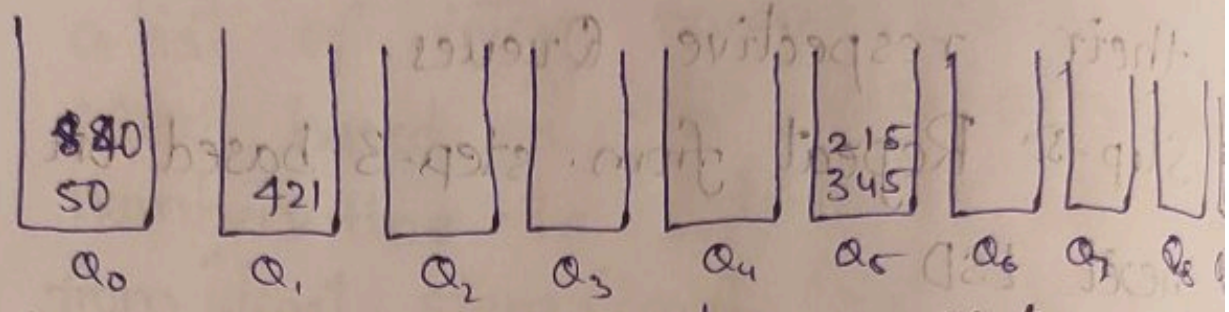


Arrange +

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA

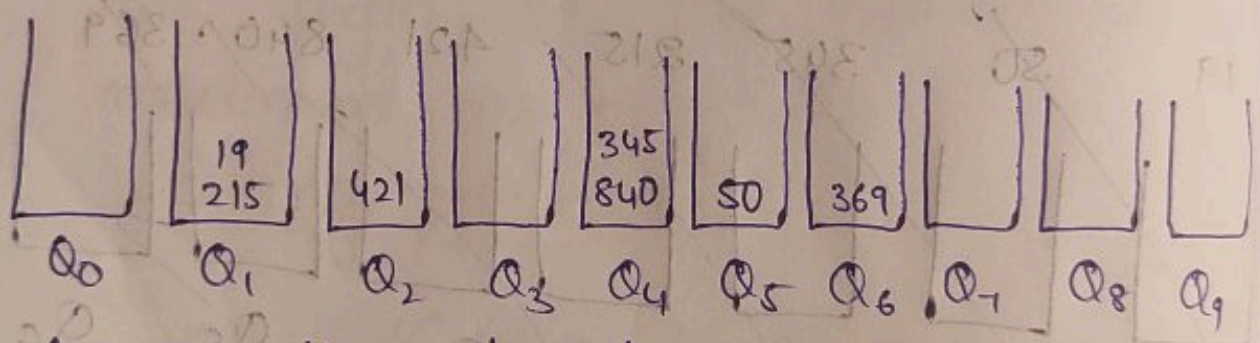
Elements

19 50 345 215 421 840 369



Arrange the elements in their respect queues based on first LSD.

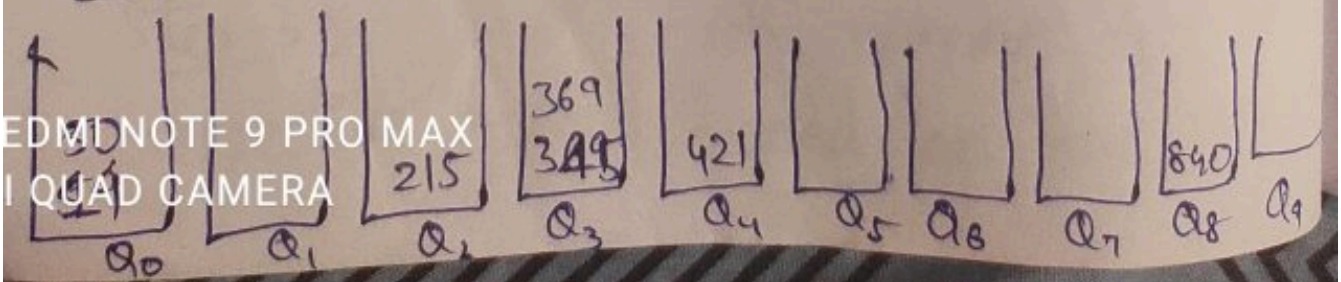
50 840 421 345 215 19 369



Arrange the elements in their respective queues based on next LSD

215 19 421 840 345 50 369

→ Now arrange the elements in their respective queues based on their next LSD



EDMI NOTE 9 PRO MAX
I QUAD CAMERA

The sorted elements

19 50 215 345 369 421 840

Applications of Data structures

Artificial Intelligence (AI)
Compiler Design (CD)
Machine learning (ML)
Data Base Management System (DBMS)
Block chain
Cryptography
Operating Systems Development.
Image processing

Time complexities of sorting

Sort	Best Case	Avg Case	Worst Case
Bubble	$O(n)$	$O(n^2)$	$O(n^2)$
Insertion	$O(n)$	$O(n^2)$	$O(n^2)$
Selection	$O(n^2)$	$O(n^2)$	$O(n^2)$
Quick sort	$O(n\log n)$	$O(\log n)$	$O(n^2)$
Merge sort	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$

REDMI NOTE 9 PRO MAX
AI QUAD CAMERA