	5
	teturn binary search (a, neg, mid-1, item);
	& start pinary search (9,109,100)
	3
10. The first of	10tons
	teturn -1;
	1
Olthut:	Enter 9
- albac.	Enter 1 tem which you want to search
	19
	Item found at 10 ration 2.
	Ganla A. O.
15	Sorting Algorithm:
1/	Bubble sort Algorithm:-
	surfine algorithm is a simplest
1.184-71	The state of the s
~~»	because moment of array elements is just lite
*	movement of an bubbles in the water.
	Bubbles in water rise up to the surface; similary
	the army elements in bubble sort move to
	It is not cuit
	The average and worst care completity
	of bubble sort is all a complexity
	of items.
	Bupple art & moone
,	· complexity does not matter
	simple and shortcode is preferred.
	preferred.

	Algorithm:
	begin Bubblesort (arr)
	for all array elements
	"if ar []] > atr [iti]
	swap (arr [i], arr [iti]]
	endif
	end for
	return an
	end bybbie sort.
ı	

Bubble SOFT complority:-

ьá.

the state of the s			
case	Time compresity	space compressity	
	destrict on College College		
Best case	o(n)	0(1).	
Average care	0(n2)		
5	<u>_</u>	-	
worst case	0(n²)		
Literature and		And and the state of the state	

Implementation of Bubble Sort:

C language Knownentation:

#include estation

void pint (Intal], int n)

int i:

for (i=0; i<n; i+t)

printf ("%d",a[i]);

?

```
void bubble (intall, intn)
  int is jotemp;
  for (i=o; ixn; itt)
    For (j=0+); j<n;j++)
    ? (a (j] x ali]
      temp = q[i];
      9[1] = 9[1];
      a[j] = temp;
void man ()
  int is temp;
  m+ 9[5] = { 10, 35, 32, 13, 26};
  int n = 51200 (a) / size of (90) i
  prints ("Before sorting array elements are : In");
  ¿(d, p) + 10kg
  bubble (a,n);
  printf ("In after sorting array elements-In");
  prot (a,n);
```

Before sorting array elements are
10 35 32 13 26

After sorting array elements are
10 13 26 32 35.

Bucket Sort Algorithm &

The data items in the bucket surt are distributed in Ferm of buckets.

Bucket sort is a sorting algorithm that sepretes elements into multiple groups said to be buckets. Elements in bucket sort are first uniformly divided into groups called buckets, and then they are sorted by any other sorting algorithm. After that, elements are gathered in sured mamer.

Advantages of bucket soit are :-

- Bucket sort reduces no of comparisons
- · It is asymptotically fast because of uniform distribut.
 -tion or elements.

limitations of bucket sort are:

- . It may or may not be a stable sorting algorithm
- · It is not useful of we have a large array box it increases the cost.
- note some entra spare is tequired to sort

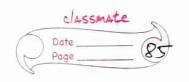
 the buckets.

	The best and average-case complexity	
	of bucket suff is 0 (n+k), worst-case	
	complexity of bucket sort is O(n2), where nis	
	number of îtems.	
	THE TOP ITE TO S.	
	Bucket sort is commonly used:	
	with floating - point values.	
	when input is distributed uniforming over a	
	range.	
Libell 4	Algorithm:	
	Bucket sort (A [])	
1.00	Let B [O Dall bo a book	
2.	· n= rength [A].	
3.	for i=0 to on-1	
4.	· make BTiJ an empty 19st	
	· +01 (=) +0 h	
6.		
7	for i=0 to ny	
8.	do sort 19ct Brit win in	
<u>g.</u>	concertenate ists \$[0], B[] B[n-i] together	
-	Po codor. blj bln-1 together	
	END.	
بالتباليسي	and the state of t	
	Complexity:	
	Time complexity:	
	was a second to the second	
-	case time complexity	
-		
	Best case o(n+k).	

	Average case	0(n+k)	
	worst case	$O(n^2)$	
2.	space complexity:		
	d .		
	Space compresity	0 (n k)	
	stable	YES	
	Implementation of	OUCKAL SOUT IN CO.	
	Implementation of bucket sort in c:-		
	#include <stdio.ht< th=""><th></th></stdio.ht<>		
	int getmax (int a		
	3		
	2 mar =9[0];		
	for (3/1 = 1; i < n	i tt	
	11		
	if (a[i] > man)	y and the second second	
	masu= 9[i];		
	beturn max;		
	9 (90) (1)	7 80+ n)	
	void bucket (Intal], int n)		
	\$	(a, b)	
	n + max = get max(a,n)		
	For (Bot 1 = 0 six = max sitt)		
		511)	
	\$		

16

```
bycket [i]=0 3
 For(ht i=0 ; ixn ; itt)
   bucket [a[i]] ++;
for (inti= 0; j=0; ix= max iit+)
  while (bucket [i] >0)
    a[j++]=1;
    bucket [i] -- ;
(a tai, [] p tai) TA tang biov
  for (inti=0; ixn; itt)
 print F(" o/od", a [i]);
int main ()
 int a[]= $54,12,84,57,69,41,9,53;
 int n = size of (a) / size of a (o) ;
 prints ("Before sorting array elements are: Ih");
 printarr(a,n);
 bucket (q,n);
 printe ( "In After sorting army elements are: h");
 pantarr(a,n);
```



1	a first frame and the second second
	output :-
	Before sorting array elements are:
	54 12 84 57 69 41 9 5 After suiting on
	and elements are :-
	5 9 12 41 54 57 6g 84.
ı	

Heap Sort Algorithm :-

creating min-heap or mux-hoop using the elements by

two main operations:

· Build a heap H, using the element of array.

· Repeatedly delete the root element of heap formed

In 1st phase.

What is heap ?

35

A heap is a complete binary tree, and binary tree is a tree in which node can have utmost two children.

Algorithm :Heap Sort (arr)
Build MaxHeap (arr)
For i = length (arr) to 2

swap arr [i] with arr [i]
heap-size [arr] = heap-size [arr]? 1
maxHeapily (arr, 1)

End.

Rulldman	
Buildmarteap	(arr)
S. Calan	

BuildmaxHeap (arr)

hoop-3120 (arr)= length (arr)

for i = length (arr) /2 to 1

MaxHeapity (arr, i)

completity :-

-			, and a second s
	(Qse	Time complexity	space complexity
	Best	O(nlogn)	0(1).
	Average	o(n logn)	C. intract
	worst	O (nlogn)	

Imprementation of Heap sort:

Pocluciex stato n>

1 function to heapity a suptree. Here is i' the inder of root node in array all, and hi is size

of heap */

void heapity (inta (], intn, inti)

int largest = i

m+ left = 2 * 1+1

n+ nght = 2 " i+2

if (lost x n gg a [lest]) a [largest] largest= left;

```
is (right xn && a [right] > a [largest]
 largest = right;
 19 (largest 1=1)
 $ Int temp = a siT:
    a[i] = a [largest];
    a [largest] = temp;
    neapily (a, n, largest);
void happsort (inta[], int n)
  Por (int i= 1/2-1; ixo, i-)
    heapify (a,n,i);
  for (inti= n-1; 170; 1-).
  3 mt temp = a [o];
     د [ ] و = [ و] ه
     9 [i] = temp;
  heapify (a,i,o);
} void print Acr (int arr[], inth)
   for (m+i= o; ixn j+ti)
       printe ( "dod", arr[i]);
       printf (" ");
3 mt man ()
  int a[] = { 48, 10, 23, 43, 28, 26, 13;
 int n = size of (a) / size of (a[0]);
  phinte ("Acting array elements are - In");
```

printfr(a,n);

neapsort (a,n);

printfr(a,n);

printfr(a,n);

return o;

ב

OUtput: Before sorting array elements are:
48 10 23 43 28 26 1

After surting array elements are
1 10 23 26 28 43 48.

Insertion sort Algorithm :-

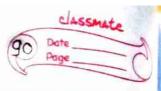
Insertion sort works similar to the sorting of playing cards in hands. It is assumed that the first card. The idea behind the insertion sort is that first make take one element, iterate it through sorted array, complexity of insertion sort in the average case and worst rure is $O(n^2)$, where n is number of items.

Theretion sort is less efficient than the other surting algorithms like heap surt, quick surt and merge sort etc.

Insortion sort has various advantages such as:
simple implementation.

- · Efficient for small date sets.
- . Adaptive i.e it is appropriate for data sets that are already substantially sorted.

```
complexity:-
are time complexity:
But rase o(n)
                             space complexity o(1).
Average case o(n2)
worst rave o(n2)
Implementation of insention sort :-
#include <stdio-h>
void insert (int a [], int n)
  int i, j, temps
  for (i=1;ixn,i++) $
    temp = a[17;
   while (j>=0 && temp == 9[]]
    a[j+i] = a[j];
   a[jti] = temps
void print Arr (int aff, intr)
   for (i=0; ix n; i++)
   print ("god", a[i])
 int main (
  intal] = $ 12,31, 25,8,32,17};
```



in+n = size of (a) / size of a[o]); prints ("Before surring array elements are - In") printarr (a, n); insert (a,n); printf ("In After sorting array elements are - In") prin-Arr(a,n); return o; output: Before sorting array elements are 12 31 25 8 32 17 After sorting array elements are -8 12 17 25 31 32 Merge Sort Algorithm :merge sort is the sorting technique that follows divide and anguer approach. This will be very helpful and interesting. merge surt is similar to the quick ext algorithm as it uses the divide and conquer approach to surt elements. Algorithm:arr is given array, beg is starting element and end is last element of a may MERGE - SORT (arr, beg, end) if beg rend Set mid = (beg + end)/2

MERGE - SCRT (arr, beg, mid)

MERGE_ SORT (arr, mid+1, end)

MERGE (arr, beg, mid, end)

```
end of if
FIND MERGE_SORT.
implementation of merge sort:
1. function of merge the subarrays of a[] *1
void merge (intaff, int beg, int mid, int end)
  Intinik;
  int n1 = mid - bogt1;
  Int na = end - mid;
  int Left Array [n], Right Array [n2];
 1 copy data to temp arrays x1
 for (inti=o; ix n1; itt)
 Left Array [i] = a [ beg ti]s
 for (int j=0; j < n2; j++)
 Right Array [j] = a [mid+1+j];
 1=0;
 j = 03
k = beg;
while (ix ni & jxh2)
  if CleftAmay [i]x = RightAmay [i])
   a[K] = LeftArray[i];
   1++ ;
 a[k]=RightArray[j];
```

-1

Complexity:-

	the state of the s		
	Cq.s-e	Time complexity	space complexity
	Best case	o(n*10gn)	o(n).
	Avg. rase	o(n*10gn)	
	worst rase	0(n*logn)	
1			