	Date. Page No. Jutorial - 3
1.	int linearcearch (int our [], int n, int ky) {
No. berti	for (int i=0; i <n; i+t){<="" th=""></n;>
The state of	if (aur[i] == kuy)
	gettern ";
	J
The same	eletwin -1;
	3
2.	iterative insertion sort
	wold insertionsort (int aver[], int n){
-	int i, j, t = 0;
100	for (i=1', i <n; i++){<="" th=""></n;>
	t = aur [i];
	j = (-1;
	while (j> = 0 88 + < our (j)) {
	arr (j+1) = arr (j);
25	j;
	3
	arer (j+1) = t;
	3
	13
	recuriere insertion sort
	noid insertionsout (int aver (), int n) {
,	if (n<=1)
1	retwen;
/	insertionsout (aux, n-1);

	Date. — Page No
	last = avr [n-1];
	j=n-2;
	while (y>=0 88 aver(j)>last)
	our [j+1] = our [j];
	J',
-	3
-	acce [j+1] = last;
	1
	Insection sout is called online sociting
	because it does not need to know
	sout and the information is requested
	solit and the information is requisted
	while the algorithm is surming.
3.11)	bubble sout - Time complexity - Best case = $O(n^2)$ whose tease = $O(n^2)$ Space complexity = $O(1)$
(a)	Time Complexity - Best case - O(m2)
	Whout case = 0 cm2)
	space complexity = O(1)
iù	Selection sout -
	Space complexity = 0(1) Selection sout - Yime complexity - Best care - 0(12)
	word case - o(n2)
	space complexity = O(1)
(iii)	Merge sout -
	Time complexity - Best Case - o (neogn)
	Space complenity = O(1) Merge sout - Time complenity - best case - o(nlogn) Worst case - o(nlogn)
	space Complexity - O(n)

			Date. — Page No.	
(v)	Insection	sout -		
	Time con Space con	plenity - Bo plenity - 01	est Care - 0	(n), Would
11)	Quick se	net -		
	Time con	plenih - 1	Best case -	- O(mloran)
			lout care	- 0(m2)
	Space a	omplering.	- o(n)	- O(nlogn) - O(n2)
w.	i) Heap sout - Vine complexity - Best case - O (nlogn) Worst case - O (nlogn) space complexity - O(1)			
	Lime Complexible - best (are a Confirma)			
	Whout cour - O (mega)			
	space co	implerity.	-0(1)	(1904)
	1			
Ч.	Souting	inplace	Stable	Online
	Selection	V		
	Invetion			-
	Merge			
	Steap			
	Bubble	V		
5	iterative	Dinney su	1911	
4	int bina	vypearch (int 19957	
2		0	int	int l, Int Tt,
_	wel	rile (l<=91)	18	int e, int n, kuy) {
		When I will		

	Dat Pag	re. —
	int m = (l+x)/2)
	if (aver[m]== KU	4)
	setwen m;	
	if (aur [m] < key) TIC.
	l= m+1;	Best Case - (1)
-	else	Aug. (ax = O(legn)
	4=m-1;	Worst Car =
	3	O (logn)
	requen -1;	
- 3		
	9 1	
nec	wrine binary search	10012011
int	binarysearch (int aru [], is	key) s
	int m = (2+4)/2	
	if (our [m] = 2 key)	
	altern m;	
	elle if (aur [m] > key	()
	ohe server binaryseard	(wee, l, mid-)
	er ye	
	retoin binaryseauce	(aug, midtl, or, key);
	7	
	uphern -1; T.C.	
Ļ	return -1; Best Ca	ne = 0(1)
3	Aug. Car	e = O(logn) are = O(logn)
	work C	are = 0 (logn)

	Date. ————————————————————————————————————
	finear Search T.C.
	Best Case: O(1)
	Avg. Case: O(n) Worst (ase: O(n)
	Worst (are 10(n)
1) 61	Recurrence selation for binary securitive search - $T(n) = T(n/2) + 1$
7.	A[i] + A[j] = k
8 .	Quick Sout is the fastest general- purpose sout. In most practical situations, quicksout is the method of choice: It stability is important be space is available, merge sout might be lest.
9.	Inversion count for an accay indicates - how fair (or close) the accay is from being sorted. If the accay is already sorted, then the inversion count is 0, but if the

	Date. — Page No
	array is sorted in the reverse order, the inversion court is the
	maximum.
	aver [] = 27,21,31,8,10,1,20,6,4,5} # include < bils / stdc++, 4>
	using namespace std;
	int merge-soit (int are (), int temp (),
	int left, int ngut);
	int merge (int aur [], int tempsint lett,
	int mid, int ught);
	int mergesort (int aur [], int auray, size) { int temp [auray size]; eleturn merge-sout (aur, temp, o,
	eltern merge-sout (aver, temp, 0,
	auray_size-1);
	int merge-sort (int arce [], int temp [], int
	left, int right) ¿
	int mid, inv-count = 0:
	if (right > left) {
	mid = itah left + (sight-left)/10'-
	inv-count + = merge-sort (aver, temp, left,
	in count + - merge sort (are temp brants
	inv_count + = merge_sort (aver, temp left, inv_count + = merge_sort (aver, temp left, inv_count + = merge (aver, temp mid+1, hight); left, mid+1, night);
12	Letwen in count;

	Date. ————————————————————————————————————
int merge (int	ver [], int temp[], int
left	int mid, int right) &
int i, j, k,	inv-count =0;
i = left;	
j=mid;	
K = left;	
	=mid-1) 88 (j<= right)) {
	or (i) <= aur[j])
	emp [1<++] = avr[i++];
else d	- [1-1-1 2010 [0]
40	mp[[t++] = acca[]++];
1 M	N-Count = inv-count + (mid-i);
4	
}	
while (i<	2 mid -1)
temp [k++]= avr[i+];
while (j<	= right)
temp	(++) = aver [j++];
for (1= le	ft; je= night; (++)
]= temp(i);
getern is	w_count;
3	
int main (){	1- 21 5 0 10 1 20 6.4 5 }
1007 /1491	
int as Z in	¿7,21,51,8,10,1,20,6,4,5}; e of (arr)/size of (arr[0]);

	Date. ————————————————————————————————————
	int ans = merges out (are, n);
	cout « " no g'inversion are " «
	sieheren O';
	3
10 ,	The woest case the complexity of
4	quick sout is O(n2). The worst care
	occurs when the pecked pirat is
	alwain in enterene of mallest or agray
	element. This happen when the
	avoidy is solited or reverse source
	and either first or last element
	is picked as pivot:
	The loest case of quick sout is when we will select pivot as a mean
	clement.
	acritico de la companya della companya della companya de la companya de la companya della compan
11	perwerence relation of:
(1)	Recurrence relation of: Merge Sout $\Rightarrow T(n) = 2T(n/2) + n$ Duick Sout $\Rightarrow T(n) = 2T(n/2) + n$
(b)	Quick Sout => T(n)= 2T (n/2)+n
-	Merge sout is more efficient 8 works
	faster than quick sout in rase of
1-11	larger array size or datasets.
-	
	is $O(n^2)$ whereas $O(n \log n)$ for neige sout.
	neege soert.

	Date. — Page No
12.	Stable selection sout
10:	
	noed stableselectionspect (int men [].
	void stableselectionspect (int aver [],
	for (int i=0; i <n-1; i++)="" i<="" th=""></n-1;>
	Ent min = 1;
	for (int j=i+1; j <n; j++){<="" th=""></n;>
	if (aver[min] > aver[i])
	min = j;
	int key - aver (min);
	while (min>i) &
	aver [min] = a [min-1];
	nin;
	2014 Fil = 4011
	all [i] = leey;
	1
	int main () 5
	int aser[] = \$4,5,3,2,4,13;
	int n = size of (aver) / size of (arr [0]);
	stableselectionsort (aur, n);
	jou (int i=0; l <n', l++).<="" th=""></n',>
	cout « aur [i] « ",
	cout « endl;
	reper 0',
	3

	Date. — Page No
13.	The casiest way to do this is to use enternal sorting. We divide our source file into temperary temporary files of size equal to the size of the RAM & first sout these files.
•	External Souting: If the input data is such that is cannot adjusted in the memory entirely at once it needs to be sorted in a naved disk, floopy disk or any other storage device. This is called extremal sorting.
•	Internal sorting: If the input data is such that it can adjusted in the main memory at once it is called internal sorting.