	Date. ————————————————————————————————————
	Jutocial - 3
1.	int linearcearch (int our (), int n, int ky) {
	for (int i=0; i <n; i+t){<="" th=""></n;>
	if (aur [i] == key)
	return ";
	}
	eletwin -1;
	3
2.	iterative insertion sort
	wold insertionsort (int aur[], int n){
	int $i, j, t = 0;$
	for (i=1; i <n; &<="" i++)="" th=""></n;>
	t = over [i];
	1=1-1;
	while (j> = 0 88 x < over (j)) {
	ara (j +1] = avar (j);
	1'
	3
	acc (j+1) = t;
	3
	3
	recurive insertion sort
	noid insertionsout (int aver (), int n){
	if (n<=1)
	netwen;
	insectionsout (aux, n-1);

=

	Page No.
	last = avr [n-1];
	j = N - 2;
	while (ix>=0 88 aver(i)>last)
	over [j+1] = over [j];
,	1;
	3
	acrec [j+1] = last;
	3
	Insertion sout is called online sorting
	because it does not need to know
	anything about what values it will
	sout and the information is requested
	while the algorithm is surming.
	/
3. W	bubble sout - Time complexity - Best case = $O(n^2)$ Worst case = $O(n^2)$ Space complexity = $O(1)$
	Time complexity - Best case = O(n2)
	Worst case = 0 (n2)
10,	Space complexity = $O(1)$ Selection sout - Yime complexity - Best case - $O(n^2)$ Word case - $O(n^2)$ Space complexity - $O(1)$
(u	Selection sout -
	Time complexity - Best care - 0(12)
	Woult care - o(n2)
	space complemy = O(1)
(iii)	Merge sout -
	Jime company - Best Case - o (neogn)
	Space complexity = O(1) Merge sout - Time companity - best Care - o(nlogn) Worst care - o(nlogn)
	space Complexity - O(n)

			Date. — Page No.	
				1
(v)	Insution	sout -		
	Time con	plenity - Be	st Care - 01	n) would
	space con	plenity - Be plonity - Ol	() la	11 - O(n2)
		· · · · · · · · · · · · · · · · · · ·		
v)	Quick S	Dect -		
	Time com	rplenity - P	est case -	O(nlogn)
		·	lout care	$-o(n^2)$
	Space o	omplering -	-o(n)	
	'	1 0		
าน์	Heap s	out -		
	Vine co	mplenity -	Best Care-	O(nlogn)
			loust case	- O(neogn)
	space co	implerity -	-0(1)	
		1	,	
4.		inplace	Stable	Online
	selection	V		
	Injuction	V		~
	Herge		~	
	Quick	~		
	Heap	V		
	Bubble		V	
	1			
1	Meratine	Dinory se	uch	
1	int bino	sinsvy sic vypearch (.	int aur []	int e int n
1			int	Kuy) §
	he	hile (l<=91)	{	0)
	11			

Date. — Page No
int m = (l+x)/2;
if (aver [m]== key)
setwin m;
if (aver [m] < key) T.C.
 l= m+1; Best Case-Q1)
else Arg. Cax = Olegn
er=m-1; Wert Car=
 3 O(logn)
repen -1;
 3
 recursive binary search
 ant binaugearch (int au), inte, into
 4 (47-2)
 int m = (l+2)/2; y (our [m] = 2 key)
sepen m;
 elle if (aver[m] > key)
netwer binarysearch (were, l, mid-)
else "Kieg);
never binarysearch (aug, midt, or,
(Key);
}
report -1; T.C.
Best Case = 0(1)
Noux Can = 0 (logn)
Work Can = 0 (logn)

		Page No.
_		
_		finiar Search T.C.
_		Best Case: O(1)
_		Avg. Case: O(n) Worst case: O(n)
_		worst (an i O(n)
1	61	Recurrence delation for linary
_		recurrence relation for binary
_		T(n) = T(n/2) + 1
_	<u>+</u>	A[i] + A[j] = K
_		
-		
	٨.	Desire dans de
	8.	Quick Sout is the fastest general -
		most perotical
		signations quicksout is the method
1_		of choice. It stability is important & space is available, merge sout
7		might be best.
		magra se sen.
	9.	Inversion count 104 111
_		indicates - how fair (or close) the
		acray is secon locing with
		the array is already souted. It
_		the inversion count is o, but if the
		or, all y in

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 ara (1, int temp (). int left, int ngut); int meage (int our [], int temp() int lett, int mid, int night); int mergesort (int aur [], int averay, rize) { int temp [averay_size]; eleturn merge-sout (aver, temp, o, array_size -1); int merge-sort (int arce[], int temp [], int left, int right) & int mid, inv-count =0; if (right > left) { mid = inght left + (right. Inv-count + = merge_sort (aver, temp in count + - merge sort (aver, temp night); in- went + = merge (aver, temp, left, mid +1, n'get); Euroun in- count;

Date	
Page No.	

	int merge (int are [], int temp[], int
	left, int mid, int ngut) &
,	int i, j, k, inv-count =0;
	i = left;
	i= mid;
	K = left;
	whèle ((i = mid-1) && (j <= right)) }
	if (aver (i) < = aver [i])
	temp [1<+] = aver [i++];
	else d
	temp[(t++) = ava[j++];
	inn-count = inn-count +
	inv-count = inv-count + (mid-i);
	3
	}
	while (i<= mid -1)
	temp [k++] = avr[i++];
	while (ic= right)
	temp [k++] = aver [j++]; for (i= left; je=night; i++)
	Jon (i= left; je= night; i++)
	aver(i)= temp(i);
	gepern inv_count;
	13
	int main (){
	int aur [] = \$7,21,51,8,10,1,20,6,4,55;
N	int n = ine of (agen)/ine of (agen [0]);

	Page No.
	int ans = mergesout (aver, n); cout « no ginversion are « cout »;
	cout « " no of inversion are " «
	eleheren O';
	{
10 ,	The uspent case time complexity of
70.	The woest case time complexity of quick sout is $O(n^2)$. The worst case
	occuers when the picked pirot is
	all my and enterene of mallet or carging
	element. This happen when input
	avorag is souted on neverse souted
	and either jeust or last element
	1 SI DIAMON 1
	The lest case of quick sout is when we will select pivot as a mean
	we will select pivot as a mean
	clement.
-	CONTENSE:
11 .	Recurrence relation of:
2)	Merge Sout = T(n) = 2T (n/2)+n Duick Sout => T(n) = 2T (n/2)+n
	Quick speet => T(n)= 2T(n/2)+n
—	Merge spect is more efficient 8 noorks
	Merge sout is more efficient 8 noorks faster than quick sout in rare of
	larger array size or datasets.
	religion complement in which west
	report case complexity for quick sout
	is $O(n^2)$ whereas $O(n \log n)$ for neige sout.

	Page No.
12.	Stable selection spect
	void stableselectionspect (int acc [], int n) {
	for (int i=0; i <n-1; 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 - arr (min);
	while (min>i) {
	aver [min] = a[min-1];
-	min;
	3
	arr [i] = leey;
	<u>j</u>
]
	int main (){
	int aser[] = 34,5,3,2,4,13;
	int n = size of (avoi) / size of (aver [0]);
	stableselectionsort (aver, n);
	Jou (int i=0; l <n', l++).<="" th=""></n',>
	cout << aur [i] << " ";
	cout « endl;
	reper 0',
	3

	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.
	0
•	External sorting: If the input data is
	such that is cannot adjusted in the
	memory entirely at once it needs
	to be solved in a nava cust floor
	is called external sorting.
	is called external sorting.
. •	Internal sociting: If the input data is such that it can adjusted in the main memory at once it is called internal sociting.
	is such that it can adjusted in the
	main memory at once it is called
	internal souting.