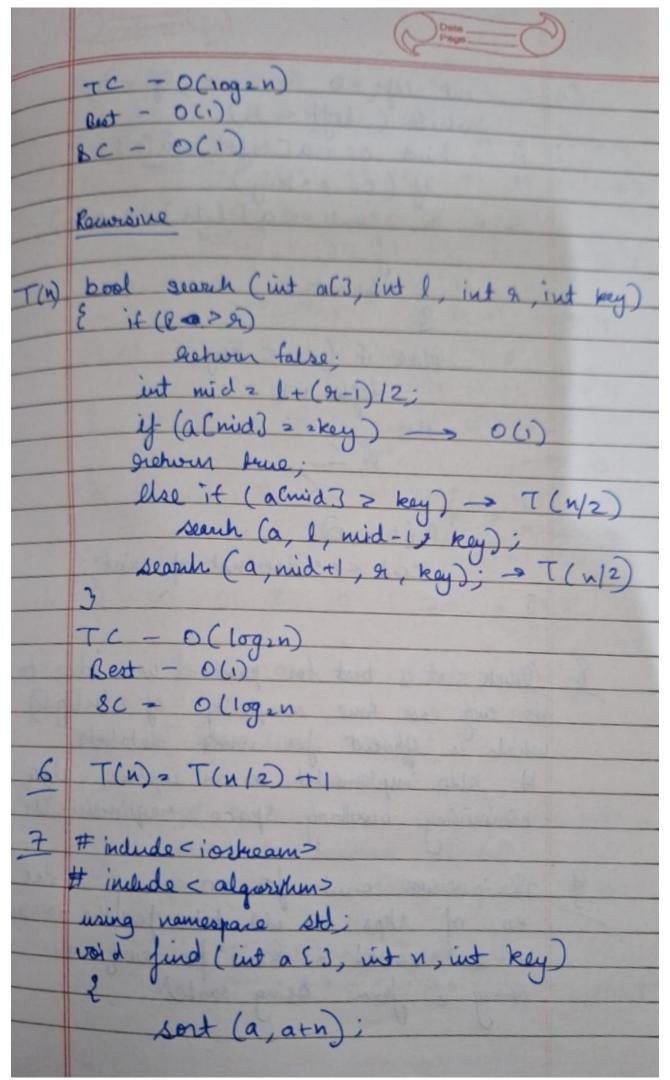
	Assignment-2
-	
1	int search (int acz, int n, int key)
	for (1 = 0; i < n; i++) { 2 if (a(i) == key)
	netry i
	Stehmen 1; Stehmen 1;
	Sehven -1;
0	T. A.
	Iterative >
	void sort (int a (), int n) { for (int i 21; i < n; i tt)
da 8	{ key = a(i];
فللغا	j = 1-1;
adia	while (j >= 0 & dr a (j 3 > key)
See 14	a [j+1] = a [j]
	j= j-1;
	3
-	a Cj+1] = key
	3 Land Mark Land Land Land Land Land Land Land Land
	to the contract of the contrac

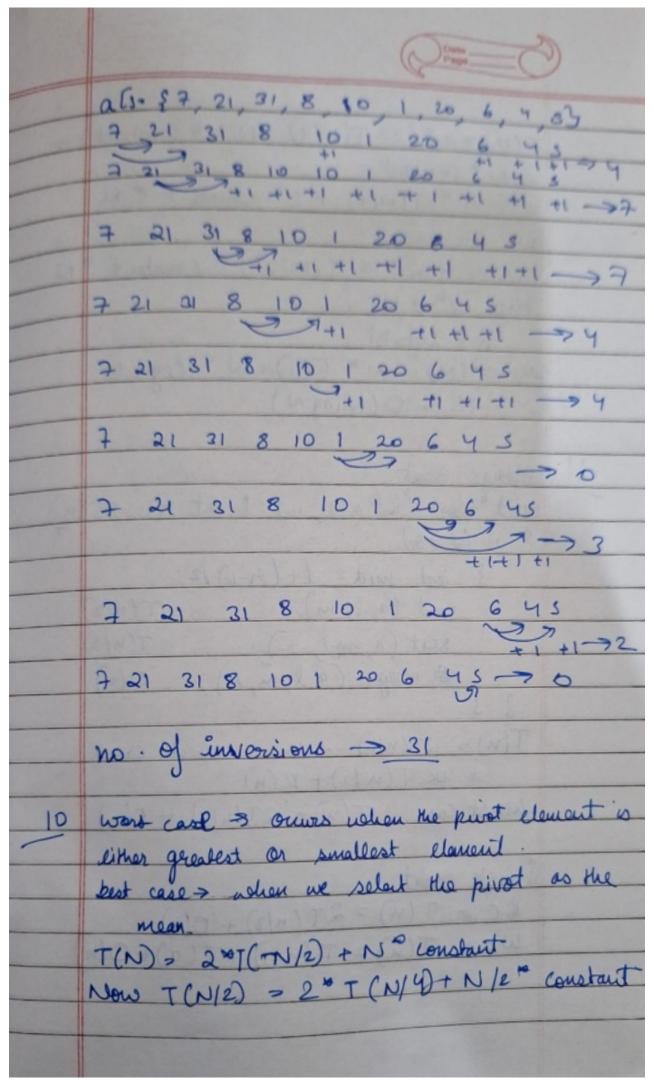
Recursive > void insertion (int al), int n) Eit (nead) Detween ; insertion (a, n-1); int last = aln-D; int j = n-2; while (j==0 & la a [j3 > last) ¿ alj+13 zalj] a Citil == last; In insertion sorting, by adding elements at last of the array, sorting is not affected. Therefore, transition is online sorting algorithm Inplace sorting algo - Bubble sort, selection sort insection sort Stable sorting algo - bubble sort External sorting algo - K way merge sort Internal sorting algo - Bubble sort, insertion sort, selection sort

3 - Bubble sort -Best case TC = O(n2) and case = 0 (n2) worst " = 0 (n2) space complainty 20(1) Selection sort -TC -> Rest = O(n2) Worst = O(n2) SC = O(1) Insertion sort TC = Best = O(n) worst = O(n2) Aug = 0(12) Space comploxity = O(1) TC = O(n+k) SC = O(n+k) Quick sort -TC - Best case > O(nlogn) sc - O(logn)

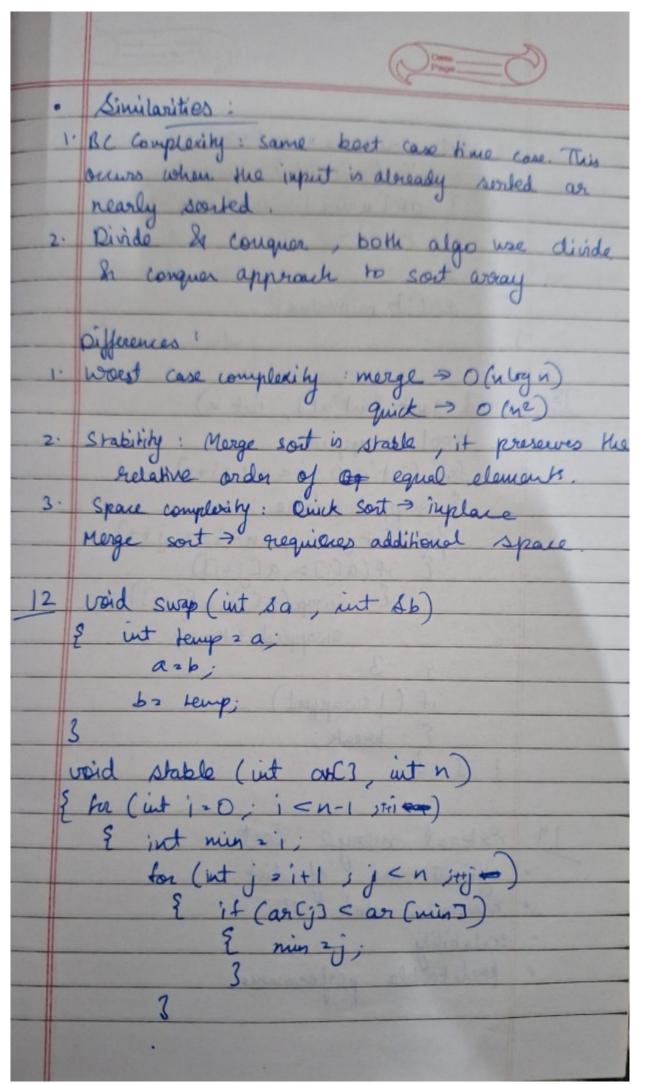
	Oate Page
	Merge sort - TC - Best - O(nlogn) wast - O(nlogn) Aug - O(nlogn) SC - O(n)
13	Heap sort TC - Best - O(nlog n) Aug - O(n log n) Worst - O(nlog n)
	Implace Stable Online Bubble sort Bubble surt Turetion sort Selection " Merge" Insertion " Treertion" Counting "
3	Iterative — int BS (int a E3, int n, int key) { int l = 0, h = n-1; while (l < = h) { int nid = (l+h)/2; if (a [mid] = = key) action hid; also if (a [mid] < key) b = l = nid +1; else if (a [mid] > key) h = nid -1; J oreturn -1;
No.	



ent left =0, Qu=n-1, \$ =0; while (left < a) Sint cs = aslefts + asas; if (cs = = key) ¿ cont « a [left] « a [2] else if (cs < key) if (+ 220) cout << " does not of exist"; 8 Quick sort is best for practical uses due to its arg. case time complexity of o(nlogn) It also implemented as an implace algo, minimising auxillary space requirements The inversion count for an array is the no. of steps it will take for the array to be sorted or how for away an array is from being sorted.



17 6	P A LOS TO THE REAL PROPERTY OF THE PARTY OF
	T(N) = 2 * (2 * T(N/4) + N/2 * constant) +
	N * constant
- 112	= 4 * T(N/4) + 2 * constant * N
	we can say
G	T(N) - 2 x x T(N/2 h) x constant * N
	then 2k 2 N
198	
	SO T(N) = N = T(N+N) + 0 - M
1 12	50 T(N)= N2 T(1)+N4 log 2 N TC = O(Nlog N)
	12 2 O (Niog N)
11	Managara
	Merge sort
	void sort (int all int l, int e) T(n)
	{ int mid = 1+ (9-1)/2;
1	sort (a, l, m); T(n/2)
0	sort (a, ny 1, 22). T(n/2)
- 100	sort (a, my 1, 22). T(n/2) 32 merge (a, l, m, h); O(n)
	TID
	T(n) = 2T(n/2) + n Best case
	2 X T(n/2)+B(n)
	worst case: T(n)= RT(n/2) + O(n)
	The state of the s
	linck sort
	BC = T(n) = 2T(n/2) + O(n)
	WC= T(n)= T(n-1)+T(0)+O(n)
	THE PROPERTY AND ASSESSMENT OF THE PARTY OF



int minvalue 2 ar (min); while (min >i) ar [amin] = ar [nin - 1); min -- ; ar (ilz minvalue) void sort (int al3, int n) for (intizo; i < n-1; i+t) { swapped = false; for (int j=0; j < n-i-1; j+t)

E if (a[j] > a[j+1])

E Swap (a[j], a[j+1]): Swapped 2 teme: if (I swapped) E break; Exteral morge sort efficient use of disk minimize disk I/D · scalability o predictable performance

	Open
	THE REPORT OF THE PARTY OF THE
5	External scerting - that can handle marrive amounts of data when the data being sorted does not fit into the main memory h instead must reside in a slower external memory. Eg > k way merge sort
I	Evernal sorting - does not require extra