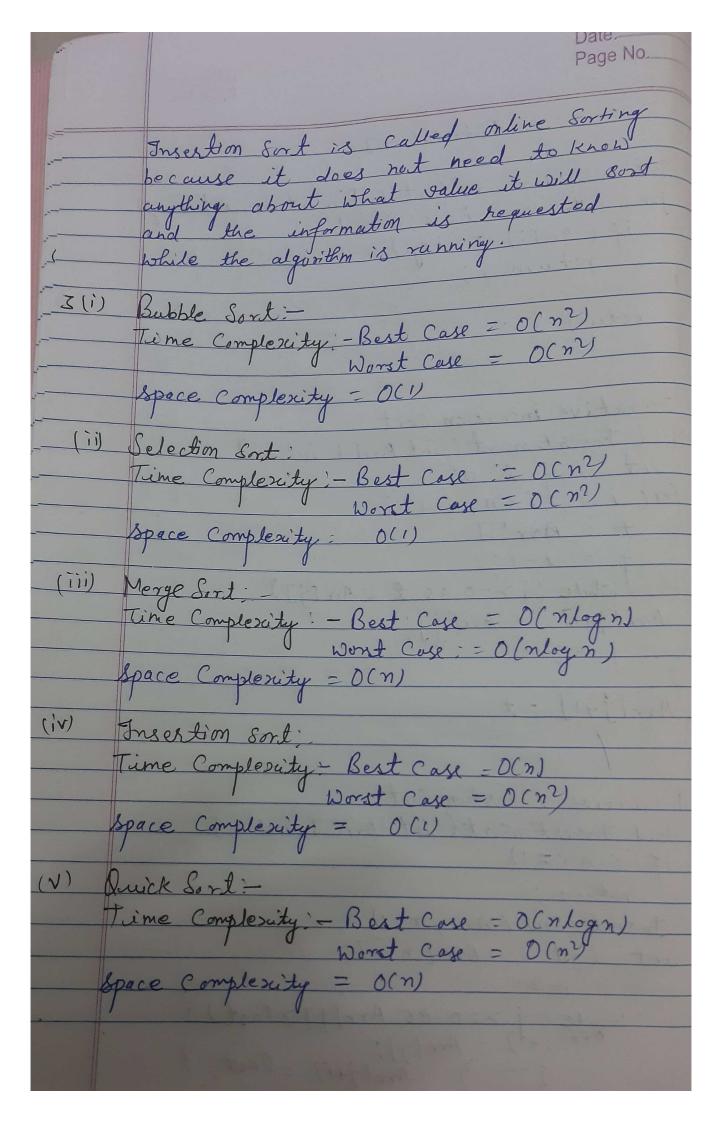
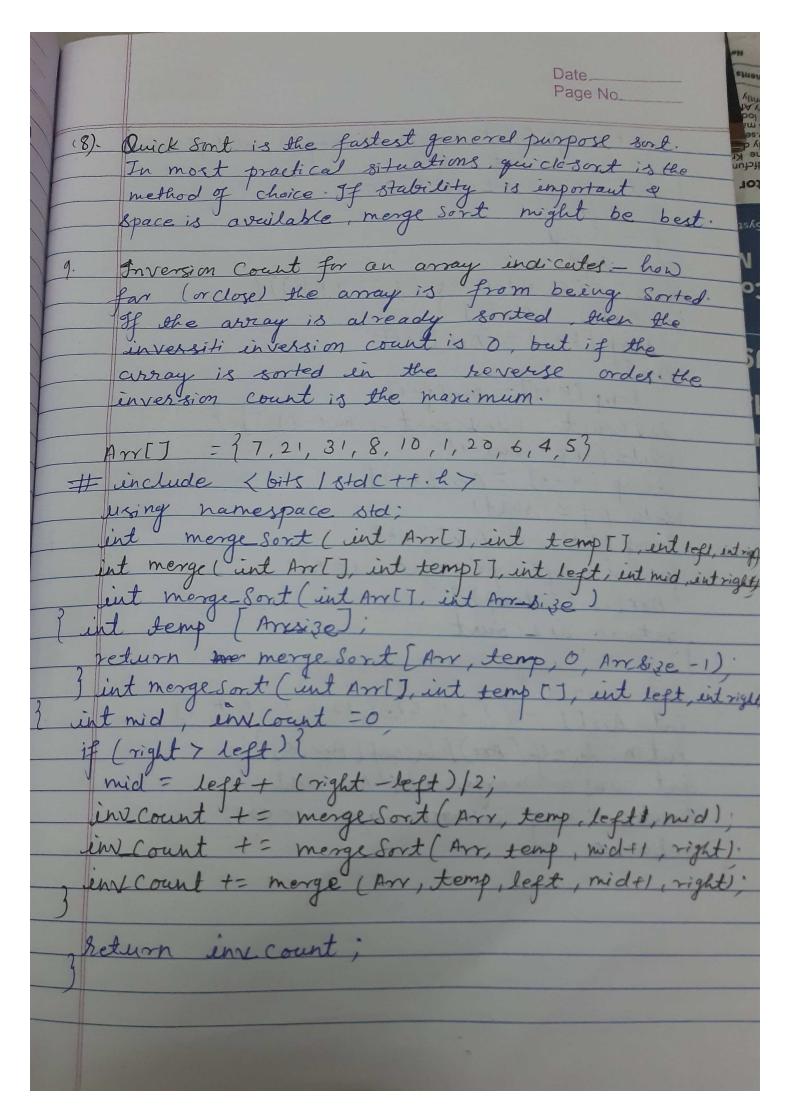
Page No. Tutorial - 3 linearSearch (int Arr [], int n, int key)
int i=0; i < n; i++) if (Arr[i] == key)
return i; return -1; Iterative insertion sort Void Insertionsort (int ArriJ, int n) i(n: i++) { t = Antij; I while (j>=0 as t < Arr[j]) Arr [j+1] = Arr [j]; Arr[1+1] = t; Recursive ingertion Sort Void InsertionSort (int Arr [], int n) return: InsertionSort (Arr, n-1). ent l'ast : Arr [n-17: While ( j >= 0 & Arr[j] > last) {
Arr[j+1] = Arr[j]; - Quet: 3 --; 3 Am[j+1] = last; 3



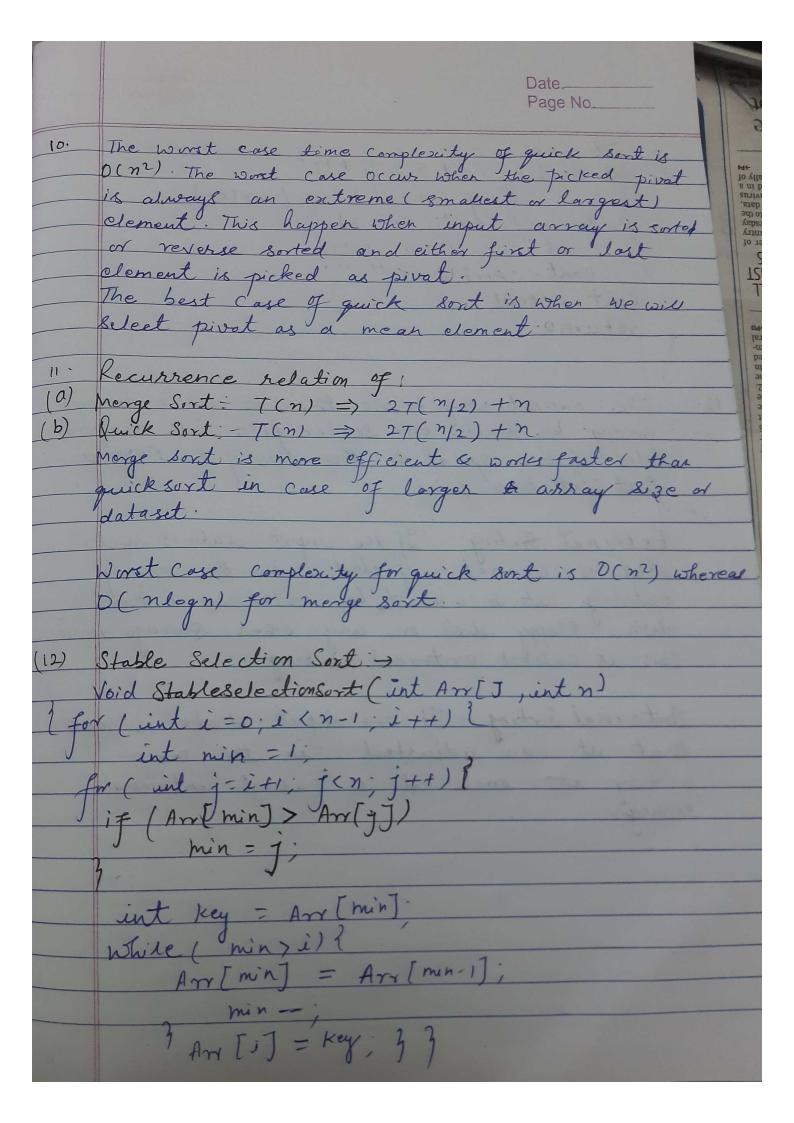
(Vi) Heap Sorti- Time Complexity: Best Case = D(nlogn) Worst Case = D(nlogn) Space Complexity = O(1)	
(4) Sorting Algo implace Stable Online  Selection  Merge  Quick  Heap  Bubble	
Iterative Binary Search:  int Binary Search (int Arr[], int l, int w, int k    While (1 < r)   2;  if (Arr[m] = key)  return m;  if (Arr[m] < key) 11 Right che search  l= m+1;	ey)
else  7 = m-1;  3  return -1; "Not found case.	T.
Note: Time complexity @ Best case = 0(1)  Best case = 0(logn)  Bookst Case = 0(logn)	

Lecursive Binary Search:int Binary Search (int AMI), int I, int = (1+x) /2) f (Arr [m] = = key return m; Binarysearch (Arr, l, mid-1, key) else if [Arr[m] > key) return return Binary search (Arr, mid +1, 7, return -1; = 0(1) = 0 ( lagn/ Best Case Word Case = 0(logn) Arg. Case incur Sourch T-c:-Best Case = O(1) Arg. Cass = 0(n) D(n) Worst Cage = (6) Recurrence relation for binary secursive Acij + Acjj = K 7.



Page No. int merge (int Arr I) int temp [], int left, int may int right)

i, j, K, in Count = 0; While ( (i <= mid-1) & (j <= right))? Lemp [K++] = Arr [i++]; temp[K++]: Ars[j++] = inv count + (mid-i); Lemp [1<++] - Arr [i++]) yeturn inscount int main() Arr [] = {7,21,57,8,10,1,20,6,4,5} int n: size of (Arr) / Size of (Arr [o])
ut ans = mergesort (Arr, n) cout << " no of ienversion ary" << org; returno;



Page No. int main() = 34, 5, 3, 2, 4, 19, int n: Size of (Mr) / Size of (Arr (0)) Stableselection sort (Arr, n); for (inti=o, i<n, i+t) Contection (1) Cout exendly returno; The easiest way to do this is to use externel sorting. We divide our Source file into tempora; files of size equal to the size of the RAM & fint sort these files. 13. External Sorting: - If the input data is such that is cannot adjusted in the memory disk floppy disk or any other storage device.
This is called external sorting. Internal Sorting: - If the input data is such that it can adjusted in the main memory at once it is called internal