Tutorial 3

1. Write linear search pseudocode to search an element in a sorted array with minimum comparisons.

```
for(0 to n) {
  If (key == A[i])
  { swap(A[i], A[0])
      return 0;
  }
}
```

2. Write pseudo code for iterative and recursive insertion sort. Insertion sort is called online sorting. Why? What about other sorting algo discussed in lectures?

```
Void InsertionSort (int arr[], int n)
{
    int i, temp, j;
    for i ← 1 to n.
    { temp ← arr[i];
      j = i-1;
      while(j >= 0 and arr[j] > temp)
      {
        arr[j+1] ← arr[j];
        j = j-1;
      }
      arr[j+1] = temp;
    }
}
```


Recursive:

```
Void recursive insertion(int arr[], int n) {
    if (n <= 1)
        return
    recursive_insertion(arr, n-1);
    int val = arr[n-1];
    int pos = n-2;
    while(pos >= 0 and arr[pos] > val)
    {
        arr[pos+1] = arr[pos];
        pos = pos-1
    }
    arr[pos+1] = val;
}
```

An online algorithm is one that can process its input piece by piece in a serial fashion ie the order that the input is fed to the algorithm, without having the entire input available from the beginning.

Insertion sort considers one input element per iteration and produces a partial solution without considering future elements, thus insertion sort is an online algorithm.

3. Complexity of all sorting algo that has been discussed.

Sorting	Best case	Worst case
Bubble	$O(n^2)$	$O(n^2)$
Selection	$O(n^2)$	$O(n^2)$
Insertion	$O(n)$	$O(n^2)$
Merge	$O(n\log n)$	$O(n\log n)$
Quick	$O(n\log n)$	$O(n^2)$
Heap	$O(n\log n)$	$O(n\log n)$

4. Divide all sorting algo into Inplace/Stable/online sorting

	Inplace	Stable	Online
Bubble	✓	✓	x
Selection	✓	x	x
Insertion	✓	✓	✓
Merge	x	✓	x
Quick	x	x	x
Heap	✓	x	✓

5. Write recursive/iterative pseudocode for binary search. Time and space complexity of Recursive and Iterative of Linear and Binary.

```
int BinarySearch(int arr[], int l, int r, int x)
{
while (l <= r) {
    int m = (l+r)/2;
    if (arr[m] == x)
        return m;
    if (arr[m] < x)
        l = m+1;
    else
        r = m-1;
}
return -1;
}
```

Binary - $O(1)$ (Best case)
$O(\log n)$ (Worst case)
space: $O(\log n)$

Linear - $O(1)$ Best case
$O(n)$ Worst case
space: $O(1)$

6. Recurrence Relation for Recursive Binary search:
Since every time the array is divided by 2. so

$$T(n) = T\left(\frac{n}{2}\right) + 1$$

7. Find two indexes $A[i]+A[j]=K$ in minimum TC.

Method = Hashing

* You need to ~~scan~~ traverse the array only once using hashmaps. If check for each element whether there exists a value sum-x in hash table, if no then add it to hash, if yes then that is the pair.

O(n)

Q8. Which sorting is best for practical uses?

→ Depends upon data and situation.

Factors deciding a good sorting algo - O(nlogn)

Running time, space, stable, swaps, data already sorted?, will data fit in RAM?

Q9. What do you mean by no of inversions? count in {7,21,31,8,10,1,20,6,4,5} using merge sort.

Inversion count indicates how close the array is from being sorted.

Two elements a[i], a[j] form an inversion if a[i] > a[j] and i < j.

If array is already sorted, Inversion = 0

If array is in reverse order, Inversion = max.

For {7,21,31,8,10,1,20,6,4,5}

Inversion count = 31

10. In which cases Quick sort will give the best and worst case TC?

Worst case = $O(n^2)$ (Sorted or reverse sorted)

Best case = O(nlogn)

11. Write R.R of Merge and Quick Sort in best and worst case? What are the similarities and differences between complexities of two algos and why?

Merge sort — O(nlogn) (in all cases)

$T(n) = 2T(n/2) + n$

$T(1) = 1$

Quick sort

$T(n) = T(n-1) + n$

$T(1) = 1$

O(nlogn) → Best case

$O(n^2)$ = Worst case

12 Selection sort is not stable by default but can you write a version of two also and why?

```
void stableSelection (int a[], int n)
{ for(i=0 to n-1)
    { int min = i
      for(j=i+1 to n)
        if (a[min] > a[j])
          min = j
```

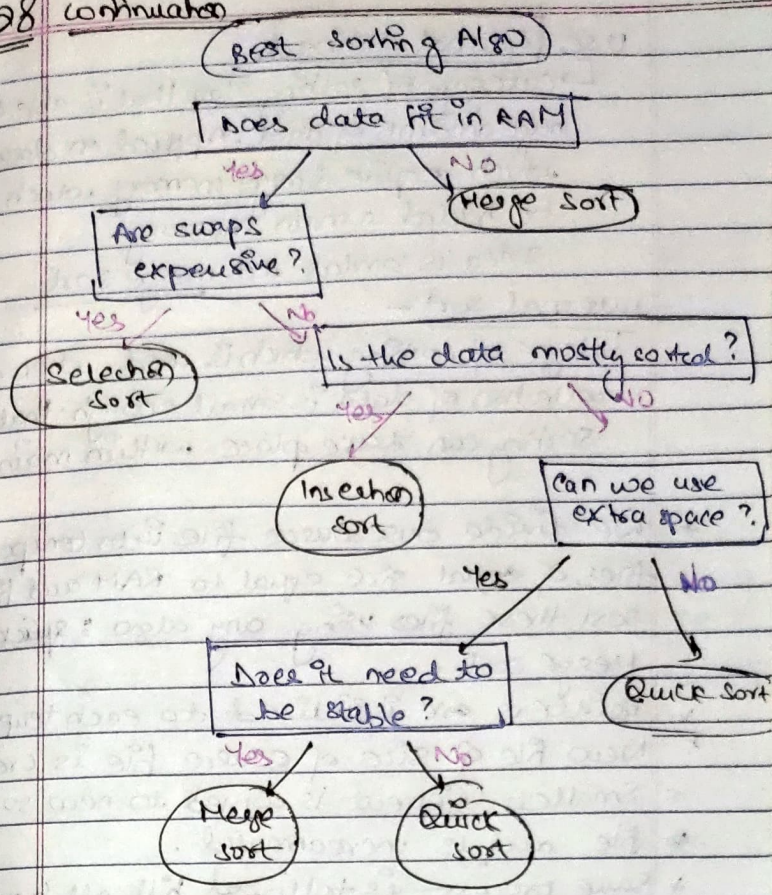
```
int key = a[min];
while(min > i)
{
    a[min] = a[min-1];
    min--;
}
a[i] = key;
}
}
```

It can be made stable if instead of swapping the minimum element is placed in its position without swapping ie placing the number in its position by pushing every element one step forward.

13. Bubble sort scans whole array even when array is sorted. Can you modify the bubble sort so that it doesn't scan the whole array once it sorted.

Run a loop before sorting to check if every time LHS element is smaller element. If yes then the array is already sorted and you need not to run the sorting loops. Or may be in the sorting loop if the one condition before hand in the else part to "continue";

Q8 continuation



Q14 Your computer has a RAM of 2 GB and you are given an array of 4 GB for sorting. Which algorithm you are going to use for this purpose and why? Explain External sorting and Internal sorting.

Use. External sorting.

↳ category of sorting algo that is able to sort huge amounts of data. Applied on data sets which acquire large memory which cannot be added in main memory.

Idea is similar to merge sort.

Internal sort -

↳ type of sorting which is used when entire collection of data is small enough that sorting can take place within main m/m

* We divide our source file into temporary files of equal size equal to RAM and first sort these files using any algo : quick sort/ Merge sort.
* Pointers are initialized to each temp file
* New file of size of source file is created
* Smallest element is copied to new source file and is increamented.
* Same process is followed till all have traversed.

and a new file which has sorted integers

Basic idea is to divide larger file into smaller temp files, sort temp files and then creating new file using these temp files