Tutorial -> 3

Q2)

```
Search (arr, n, x)
  if (arr [n-1] == x)
    {
    return "found";
    }
  int backup = arr[n-1];
  arr[n-1] = x;
  for (int i=0; i++)
      {
      if (arr[i] == x)
      {
       arr(n-1] = backup
      if (i<n-1)
        return "found";
      else
        return "Not found";
      }
   }
  }
```

Q4) Bubble sort, insertion sort, selection sort are inplace sorting algorithm. Bubble & Insertion sort can be applied as stable algorithm.

- Merge sort ~~sort~~ is a stable but not an inplace algorithm.

- Insertion sort is also used for online sorting.
- Quick sort is not stable but is an in-place algorithm.
- Heap Sort is an in-place but not stable
- Selection sort is also an online sorting algorithm.

Q) Time Complexity:

BC = O(1)

Avg. Case = O(n log n)

Worst = T(n) = T(n/2) + C

Recurrence relation

$$T(n) \begin{cases} C & \text{if } i >= 1 \\ T(n/2) + C & \text{otherwise} \end{cases}$$

Q) Recursive :-

```
int binary (int arr[], int low, int
                high, int x)
{
 if (low > high)
 {
 return -1;
 }
int mid = (low + high) / 2;
 if (X == arr[mid])
    return mid;
```

```
else if (x < arr[mid])
    return binary(arr, low, mid-1, x);
else
    return binary(arr, mid+1, high, x);
}
```

③ Algorithm — Time Complexity

Algorithm	Best	Avg.	Worst
Selection	$\Omega(n^2)$	$\Theta(n^2)$	$O(n^2)$
Bubble	$\Omega(n)$	$\Theta(n^2)$	$O(n)$
Insertion	$\Omega(n)$	$\Theta(n^2)$	$O(n)$
Heap	$\Omega(n \log n)$	$\Theta(n \log n)$	$O(n \log n)$
Quick	$\Omega(n \log(n))$	$\Theta(n \log n)$	$O(n \log n)$
Merge	$\Omega(n \log(n))$	$\Theta(n \log n)$	$O(n \log n)$

④ Quick sort is best in terms of practical use as most wide sorting algorithms at present.

→ Has running time of $O(n^2)$ that makes it susceptible in applications

→ Most often than not runs at $O(n \log n)$

→ High space efficiency by executing in place.

→ q.sort utility in C programming language is powered quick sort.

classmate
Date
Page

(10) Worst Case : $O(n^2) \rightarrow O(n^2)$

→ The worst case occurs when the partition process picks up greatest or smallest element as pivot.

Best Case :

→ The best case occurs when the position process a picks the middle element as pivot.

(14) Sorting 4GB data using 2GB RAM :

- Read 2GB of data in main memory & sort it by quick sort.
- Write sorted data to disk
- Repeat it all until all data is in sorted 2GB chunks i.e 4/2 = 2 chunks.
- Which now need to be merged in a single file.
- Perform 2 way merge & store the result in buffer [needs 200MB] from each sorted chunk into input & allocation for output buffer.

(4)

```
(i=0; i<n; i++)
{
for (j=0; j<n; j++)
 {
  a[i] + a[j] ; k
 {
 }
```