Aditya Thapliyal
F-40

Tutorial - 3

(3-1)
Aditya

Q1 ~~for (i=0 to n)~~ ~~if(a~~

```
for (i=0 to n)
{ if (arr[i]==key)
     break;
}
```

Q2)

```
void insertion sort (int arr[], int n)
{ for (int i = 1 ; i<n ; ++i)
   {
       j = i-1
       x = arr[i]
       while (j>-1 && arr[j]>x)
        { arr [j+1] = arr [j];
           j--;
        }
       arr[j+1] = x;
   }
}

void insertion (int arr[], int n)
{
    if (n<=1) return
    insertion ~~s~~ (arr, n-1);
    int last = arr[n-1];
    int j =n-2
    while (j >= 0 && arr[j] > last)
      {  arr[j+1] = arr[j];
          j--;
      }
    arr[j+1] = last;
}
```

(2-2) Ady

Insertion sort doesn't need to know anything about what value it will sort during running & hence called online sort.

Other Sorting Algos

1) Bubble Sort
2) Quick Sort
3) Merge Sort
4) Selection Sort
5) Heap Sort

(Imp) Complexity.

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

Q M)

Inplace	stable	Online
Bubble Selection	Merge	Insertion
Insertion	Bubble	
Quick	Insertion	
Heap	~~Heap~~ Count Sort	

← Sorting Type

3-3 Aditya

Binary Search

Iterative

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

Recursive

```
int bs (int ar , int l, int r , int k)
{ while (l<=r)
  { m= (l+r) /2
   if( K == ar[m]
   return m;
   else if (key < arr[m])
    return bs(ar, l, mid-1, k)
   else
   return bs (arr, mid +1, r, k) ;
 }
return - 1
```

Linear search - O(n)

Binary = O(log n)

Que

$T(n) = T(n/2) + 1$ - ①

$T(n/2) = T(n/4) + 1$ - ②

$T(n/4) = T(n/8) + 1$ - ③

$T(n) = T(n/2) + 1$

$T(n/4) + 1 + 1$

$T(n/8) + 1 + 1 + 1$

⋮

$T(n/2^k) + 1$ (k times) Let $2^k = n$

$k = \log n$

$T(n) = T(n/n) + \log n$

$T(n) = T(1) + \log(n)$

$T(n) = O(\log n)$ →.

Que Best Sorting for practical use

Quick sort is the fastest general sort.
It is stable & has the avg and best running time of O(nlogn)

Que 31

(10) Quick sort given the worst time complexity in

1) The array is sorted and either the first or the last element is selected as a pivot.

2) ~~Worst~~ Best case when the pivot is a mean element.

(11)

Merge sort →

Best case → $T(n) = 2T(n/2) + O(n)$

Worst case → $T(n) = 2T(n/2) + O(n)$

Quick Sort

Best case $T(n) = 2T(n/2) + O(n) \to O(n\log n)$

Worst case → $T(n) = T(n-1) + O(n) \to O(n^2)$

Que

To prevent bubble sort from scanning the whole array if it is sorted already then we can use a counter to check if any exchanges were made. If not then we break the loop and conclude that the array is sorted

```
void bubble ( int ar , int n)
{      int cnt=0;
  for (int i=0 ; i<n ; ++i )
   {
      for (int j=0; j<n ; ++j )
        {
            If (ar[j] > ar[~~ar~~j+1])
           { Swap (ar[j]), ar(j+1) cnt ++;}
         }
      if( !cnt)  break;
  }
```