Abhinav Rawat
BTech CST SPL-1(44)
2017446

Tutorial 3

Design and analysis of algorithms [DAA]

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

Ans: 1
```
boolean linear_Search(int a[], int e, int n)
{
  int flag=0;
  for(int i=0;i<n;i++)
  {
     if(a[i]==e)
        flag=1;
     else
        flag=0;
  }
  if(flag==1)
     return true;
  else
     return false;
}
```

Que 2 Write pseudocode for iterative and recursive insertion sort. Insertion sort is called online sorting. Why? What about other sorting algorithms that has been discussed in lectures?

Ans: 2 Iterative -
```
void insertionSort(int a[], int n)
{
   for(int i=1; i<n; i++)
   {
      int value=a[i];
      int j=i;
```

```
while(j>0 && a[j-1]>value)
{
  a[j]=a[j-1];
  j--;
}
a[j]=value;
}
}
```

Recurssive -

```
void insertionSort(int arr[], int i, int n)
{
  int value=arr[i];
  int j=i;
  while(j>0 && arr[j-1]>value)
  {
    arr[j]=arr[j-1];
    j--;
  }
  arr[j]=value;
  if(i+1<=n)
  {
    insertionSort(arr,i+1,n);
  }
}
```

Insertion sort is called an online sorting algorithm because insertion sort considers an input element per iteration and produces a partial solution without considering future elements.

2017446

Que 3 Complexity of all sorting algorithms that has been discussed in lectures

Ans: 3

	Best	Average	Worst
Bubble sort -	$O(n^2)$	$O(n^2)$	$O(n^2)$
Selection sort -	$O(n^2)$	$O(n^2)$	$O(n^2)$
Insertion sort -	$O(n)$	$O(n^2)$	$O(n^2)$
Heap Sort -	$O(n\log(n))$	$O(n\log n)$	$O(n\log n)$
Quick Sort -	$O(n\log n)$	$O(n\log n)$	$O(n^2)$
Merge Sort -	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$
Count Sort -			

Que 4 Divide all the sorting algorithms into implace/stable/online sort

Ans: 4

Sorting algorithm	Inplace	Stable	Online
Bubble sort	Yes	Yes	No
Selection sort	Yes	NO	No
Insertion sort	Yes	Yes	Yes
Quick Sort	Yes	NO	No
Merge sort	No	Yes	No
Heap sort	Yes	No	No

2017446

Ques 5 Write recursive/iterative pseudo code for binary search. What is the Time and Space complexity of Linear and Binary Search?

Ans:5 Iterative -

```
int binarySearch(int a[], int x)
{
    int low=0 , high= a.length-1;
    while (low <= high)
    {
        int mid=(low+high)/2;
        if (x==a[mid])
            return mid;
        else if (x<a [mid])
            high = mid-1;
        else
            low = mid+1;
    }
    return -1;
}
```

Recursive -

```
int binarySearch(int a[], int low, int high, int x)
{
    if (low> high)
        return -1;
    int mid= (low+high)/2;
    if(x==a[mid])
        return mid;
    else if(x< a[mid])
        return binarySearch(A,low,mid-1,x);
    else
        return binarySearch(A,mid+1, high, x);
}
```

2017446

Time complexity → Iterative - $O(\log N)$
Recursive - $O(\log N)$

Space complexity → Iterative - $O(1)$
Recursive - $O(\log N)$

Ques 6 Write recurrence relation for binary recursive search.

Recurrence relation - $T(n) = T(n/2) + 1$

Derivation -

1st step - $T(n) = T(n/2) + 1$

2nd step - $T(n/2) = T(n/4) + 1 \ldots\ldots [T(n/4) = T(n/2^2)]$

3rd step - $T(n/4) = T(n/8) + 1 \ldots\ldots [T(n/8) = T(n/2^3)]$.

⋮

K th step = $T(n/2^{k}-1) = T(n/2^k) + 1$ (K times)

Adding all equations

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

$\frac{n}{2^k} = 1$

$n = 2^k$

$\log n = k$

$k = \log n$

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

$T(n) = O(\log n)$

2017446

Que 7 Find two indices such that $A[i]+A[j]=k$ in minimum time complexity

```
vector<int> find(arr[], k, n)
{
    vector <int> sol;
    for i=0 to n-1
        for j=0 to n
            if arr[i]+arr[j]=k
                sol.pushback(i)
                sol.pushback(j)
            return sol.
```

2017446

Que 8 Which sorting is best for practical uses? Explain

Ans Quicksort is the fastest general-purpose sort. In most practical situations quicksort is a method of choice. If stability is important and space is available, merge sort might be best. In some performance-critical applications, the focus may be on just sorting numbers, so it is resonable to avoid the costs of using references and sort primitive types instead.

Que 9 What do you mean by number of inversions in an array? Count the number of inversions in Array arr[] = {7, 21, 31, 8, 10, 1, 20, 6, 4, 5} using merge sort.

Ans: 9 Inversion count for an array indicates how far or close the array is from being sorted. If the array is alredy sorted then the inversion count is 0, but if the array is sorted in reverse order, the inversion count is the maximum.

Pair of inversions in array arr[] = (7,1) (7,6) (7,4) (7,5) (21,8) (21,10)
(21,1) (21,20) (21,6) (21,4) (21,5)
(31,8) (31,10) (31,1) (31,20) (31,6)
(31,4) (31,5) (8,1) (8,6) (8,4) (8,5)
(10,1) (10,6) (10,4) (10,5) (20,6) (20,4)
(20,5) (6,4) (6,5)

Count = 31

2017446

Que 10 In which cases Quick sort will give the best and the worst Case time complexity?

Ans:10 Best Case- The best case occurs when the partition process always picks the middle element as pivot. Following is the recurrence relation for best case.

$T(n) = 2T(n/2) + \theta(n)$.

Worst Case - The worst case occurs when the partition process always picks greatest or smallest element as pivot. It above parition stratergy is considered where last element is always picked as pivot, the worst case would occur when the array is already sorted in increasing or decreasing order.

Que 11 Write Recurrence relation of Merge and Quick sort in best and worst case? What are the similarities and differences between complexities of two algorithms and why?

Ans. 11 Quick Sort →

Recurrence relation - Best case - $T(n) = 2T(n/2) + \theta(n)$

worst case - $T(n) = T(n-1) + \theta(n)$

Merge Sort →

Recurrence relation - Best case - $2T(n/2) + \theta(n)$

worst case - $2T(n/2) + \theta(n)$

Time complexity -

	Best Case	Worst Case
Quick sort	$O(n \log n)$	$O(n^2)$
Merge sort.	$O(n \log n)$	$O(n \log n)$

2017446

Que 12 Selection sort is not stable by default but can you write a version of stable selection sort.

Ans

```
void stableSelectionSort(int a[], int n)
{
    for(int i=0; i<n-1; i++)
    {
        int min=i;
        for(int j=i+1; j<n; j++)
        {
            if(a[min]>a[j])
                min=j;
        }
        int key=a[min];
        while (min>i)
        {
            a[min]=a[min-1];
            min--;
        }
        a[i]=key;
    }
}
```

Que 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 is sorted.

Ans

```
void bubbleSort (int a[], int n)
{
    int flag, temp;
    for(int i=0; i<n-1; i++)
    {
        flag=0;
        for(j=0; j<n-i-1; j++)
        {
```

```
flag = 1;
if(arr[j] > arr[j+1])
{
    temp = a[j];
    a[j] = a[j+1];
    a[j+1] = temp;
}
}
if(flag == 0)
    break;
}
}
```

Que 14 Your computer has a RAM (Physical memory) of 2 GB and you are given an array of 4 GB of sorting. Which algorithm you are going to use for this purpose and Why? Also explain the concept of External and Internal Sorting.

Ans:14 As the size of given array exceeds the size of RAM ~~exceeds~~ ~~the size of~~ therefore we will use K-way merge sort as sorting technique as it takes a part of array & sort it, whole array is not loaded into main memory altogether.

External sorting - This algo loads a part of array and sort it whole array is not loaded into the RAM. especially used to sort array of large size.
eg - k-way merge sort.

Internal sorting - These algo needs whole array to reside in RAM during execution
ex - bubble sort.

2017446