21/09/2020

1981A0SN5
CSE-D

Data Structures

Assignment - 1

① Assume that there is a list {22, 22, 22, 22, 22, 22, 223}. What happens when selection sort is applied on the list? Explain.

Ans:

Selection Sort:

The selection sort algorithm sorts an array by repeatedly finding the minimum element from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array.

→ The subarray which is already sorted.

→ Remaining subarray which is unsorted.

In every iteration of selection sort, the minimum element from the unsorted sub array is picked and moved to the sorted subarray.

For the given array {22, 22, 22, 22, 22, 22, 223}, since all the elements are same, there will be no swapings. But $\frac{(N^2+N)}{2}$ comparisions will be done. Finally we get the same input array as output.

(2) Sort the following list of names using Insertion sort: Varun, Amar, Karthik, Ramesh, Bhuvan, Dinesh, Piroz and Ganesh.

Ans: Insertion sort:

Insertion sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.

Time complexity: $O(n^2)$

space complexity: $O(1)$

→ For the given array of names, the algorithm goes as follow...

→ Temp Amar: Varun, Amar, Karthik, Ramesh, Bhuvan, Dinesh, Piroz, Ganesh

→ Temp Karthik: Amar, Varun, Karthik, Ramesh, Bhuvan, Dinesh, Piroz, Ganesh

→ Temp Ramesh: ~~Amar~~, ~~Varun~~ Karthik, Varun, Ramesh, Bhuvan, Dinesh, Piroz, Ganesh.

→ Temp Bhuvan: Amar, Karthik, Ramesh, Varun, Bhuvan, Dinesh, Piroz, Ganesh.

→ Temp Dinesh: ~~Amar~~, Bhuvan, ~~Dinesh~~, Karthik, Ramesh, ~~Varun~~, ~~Piroz~~ Ganesh.

→ Temp Dinesh: Amar, Bhuvan, Karthik, Ramesh, Varun, Dinesh, Piroz, Ganesh.

→ Temp Piroz: Amar, Bhuvan, Dinesh, Karthik, Ramesh, Varun, Piroz, Ganesh.

→ Temp Ganesh: Amar, Bhuvan, Dinesh, Piroz, Karthik, Ramesh, Varun, Ganesh

Final array :

Anvar, Bhuvan, Dinesh, Firoz, Ganesh, Karthik, Ramesh, Varun.

CODE:

```
Import java.util.*;
Public class Insertion sort {
    Public static void main(String[] args) {
        String arr[]= {"Varun","Karthik","Bhuvan","Dinesh",
                       "Firoz","Ganesh"};
        int n = arr.length;
        System.out.println("Before sorting :");
        for(String i : arr){
            System.out.println(i+" ");
        }
        String sorted[] = Insertion sort(n, arr);
                          System.out.println("After sorting");
        for(String i : sorted) {
            System.out.print(i+" ");
        }
    }
    Public static String[] insertion sort(int n, String[] arr){
        String key, temp;
        for(int i=1; i<n; i++){
            int j = i-1;
            key = arr[i];
            while(j>=0 && key.charAt(0)<arr[j].charAt(0)){
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp";
                j--;
            }
        }
        return arr;
    }
}
```

(3) Sort the following numbers using Quick sort:
67, 54, 9, 21, 12, 65, 56, 43, 34, 79, 70, 45.

Ans. pivot
67

(67), 54, 9, 21, 12, 65, 56, 43, 34, 79, 70, 45 R → l
key > 45

45, 54, 9, 21, 12, 65, 56, 43, 34, 79, 70, 45 (67)

left → right

45, 54, 9, 21, 12, 65, 56, 43, 34, (67), 70, 79

(45) 54 9 21 12 65 56 43 34 (70) 79
key ← key ←

34 54 9 21 12 45 56 43 34
←

L → R

34 43 9 21 12 65 56 (45) 54.

34 43 9 21 12 (45) 56 65 54.

(34) 43 9 21 12 (56) 65 54
key ← key ←

12 43 9 21 (34) (59) 65 56
key key

12 43 9 21 (34) 54 65 (56)
key.

12 21 9 34 43 54 56 65

(12) 21 9

9 21 (12)

9 12 21

9 12 21 34 43 45 54 56 65 67 70 79.

(9) Implement Linear Search & Binary search using Recursion.

Sol: Linear search using Recursion:

```
import java.util.Scanner;
Public class LinearSearch {
    Pubelic static void main (String[] args) {
        int arr[] = {6, 18, 40, 18, 13, 30, 28, 38};
        Scanner inp = new Scanner(System.in);
        int key = inp.nextInt();
        int pos = search(arr, 0, key, arr.length);
        if (pos == -1)
            System.out.println("Posibion of" + key + "is"
                                + pos);
        else
            System.out.println("Element not found");
    }
    Public static int search (int a[], int i, int key, int n) {
        if (a[i] == key)
            return i+1;
        else if (i == n-1)
            return -1
        else {
            i++;
            return (a, i, key, n);
        }
    }
}
```

Binary search using Recursion:

```
class BinarySearch {
    int binarySearch(int arr[], int l, int r, int x) {
        if (r >= l) {
            int mid = l + (r - l) / 2;
            if (arr[mid] == x)
                return mid;
            if (arr[mid] > x)
                return binarySearch(arr, l, mid - 1, x);
            return binarySearch(arr, mid + 1, r, x);
        }
        return -1;
    }
    Public static void main(String[] args) {
        BinarySearch ob = new BinarySearch();
        int arr = {2, 3, 10, 9, 7};
        int n = arr.length;
        int x = 10
        int result = ob binarySearch(arr, 0, n-1, x);
        if (result == -1)
            System.out.println("Element not Present");
        else
            System.out.println("Element found at" + result);
    }
```

(5) Explain in brief the various factors that determine the selection of an algorithm to solve a computational problem.

Ans: There are various that define the selection of the algorithm like hardware, processor, Ram etc. of the device and space complexity and time complexity of the algorithm. Majority the selection of an algorithm to solve a computational problem depends on time complexity and space complexity.

Time complexity:

It is the computational complexity that describes the amount of time it takes to run an algorithm. It is commonly estimated by counting the number of elementary operations performed by the algorithm.

Algorithms runtime may vary among different inputs of the same size, mostly the "worst case time complexity" is taken into consideration.

eg: Insertion sort ($O(n^2)$), Quick sort : $O(n^2)$

Space complexity:

Space complexity of an algorithm is the amount of memory required to solve the instance of the computational problem based on the inputs given. It is also expressed asymptotically in big 'O' notation.

Eg: insertion sort : $O(n)$, Quick sort: $O(n)$.