Vijay Ganesh Panchapakesan (1001861777)

CSE-5311 DAA

Assignment-1

Name: Vijay Ganesh Panchapakesan

Student_Id:1001861777

1) You have an array containing integers from 1 to 10 (not in order) but one number is missing (there is 9 numbers in the array).

a) write a pseudo code to find the missing number

PseudoCode:

```
For i in range(1,11):
    Sum1=sum1+i #Find the sum of first 10 Natural numbers by means of running sum
Loop through the array:
    Sum=the sum of all elements in an array
Missingno=sum1-sum
```

b) what is the worst case run time complexity of your suggested solution

The worst case run time of the above algorithm is O(n)

2) You are given an array of integers:

a) Write a pseudo code to find all pairs of numbers whose sum is equal to a particular number

```
dict={}
oparr=[] #Output
Loop through the array:
        If(element in the dict):
                Push(sum-element,element) into oparr #where sum is the given sum
        else:
                dict[sum-element]=element #adding element to sum-element key
print oparr
```

b) what is the worst case run time complexity of your suggested solution

The worst case run time of the above algorithm is O(n)

3) You are given an array of integers:

a) write a pseudo code to remove duplicates from your array

```
dict={}
midlen=Mid of the array
Loop array till midlen:
    If(element in dict):
```

```
            dict[element]+=1 #Increase the count by 1
        else:
            dict[element]=1
    Loop array from midlen till end of the array:
        If(element in dict):
            dict[element]+=1 #Increase the count by 1
        else:
            dict[element]=1
    output=print the keys of the dict
```

b) what is the worst case run time complexity of your suggested solution

The worst case run time of the above algorithm is O(n/2)=O(n)

4) you are given 2 sorted arrays:

a) write a pseudo code to find the median of the two sorted arrays

```
let arr1 and arr2 be two sorted arrays.

i=0,j=0,oparr=[]

while(i<length of arr1 && j<length of arr2):

        if(arr[i]<arr[j]):

                push arr[i] into oparr

                i++

        else:

                push arr[j] into oparr

                j++

while(i<length of arr1): #for pushing rest of the elements into the array

        push arr[i] into oparr

        i++

while(j<length of arr2)

        push arr[j] into oparr

        j++

len=length of oparr

if(len%2==0):
```

```
        median=(oparr[len/2 -1]+ oparr [len/2])/2
    else:
        median= oparr[Math.floor(len/2)]
    print median
```

b) what is the worst case run time complexity of your suggested solution

The worst case run time of the above algorithm is O(n)

4) Use one of the sorting algorithms to sort the following array input = { 4, 1, 2, 7, 10, 1, 2, 4, 4, 7, 1, 2, 1, 10, 1, 2, 4, 1, 2, 7, 10, 1, 2};

I am going to use the Insertion sort Algorithm:

Algorithim goes like this: (In Javascript)

```
var len = array.length
  for (let i = 1; i < len; i++) {
    var cur = array[i]
    let j = i - 1
    while (j > -1 && cur < array[j]) {
      array[j + 1] = array[j]
      j--
    }
    array[j + 1] = cur
    console.log(array)
  }
```

Step1:

i=1,j=0,cur=1

Since cur<arr[j] swap arr[j+1] with arr[j] and j--.By doing so j=-1 so it comes out of the loop .So arr[j+1]=4 and out of the loop we are putting the value of cur to the arr[j+1].So arr[j+1]=arr[-1+1]=1

O/P after first iteration [1, 4, 2, 7, 10, 1, 2, 4, 4, 7, 1, 2, 1, 10, 1, 2, 4, 1, 2, 7, 10, 1, 2]

Step 2:

i=2,j=1,cur=2

Since cur<arr[j] swap arr[j+1] with arr[j] and j--.So now arr[j+1]=4 and j—and j now becomes 0.Since j>-1 but cur>arr[j] so comes out of the loop and arr[j+1]=arr[1]=cur=2

O/P after the second iteration= [1, 2, 4, 7, 10, 1, 2, 4, 4, 7, 1, 2, 1, 10, 1, 2, 4, 1, 2, 7, 10, 1, 2]

Step 3:
i=3,j=2,cur=7
Since cur>arr[j] there will not be a while loop execution and arr[j+1]=arr[3]=7 and there will not be any change in the array and the o/p looks like this after 3
O/P after the Third iteration: [1, 2, 4, 7, 10, 1, 2, 4, 4, 7, 1, 2, 1, 10, 1, 2, 4, 1, 2, 7, 10, 1, 2]

Step4:
i=4,j=3,cur=10
Since cur>arr[j] there will not be a while loop execution and arr[j+1]=arr[4]=10 and there will not be any change in the array and the o/p looks like this after 5
O/P after 4th Iteration is : [1, 2, 4, 7, 10, 1, 2, 4, 4, 7, 1, 2, 1, 10, 1, 2, 4, 1, 2, 7, 10, 1, 2]

Step 5:
i=5,j=4,cur=1
Since j>-1 and cur<arr[j], arr[j+1]=10 which means that arr[5]=10 and j--.Now the j=3 and j>-1 and cur<arr[j+1],arr[j+1]=7 i.e arr[4]=7 and j--.Now j=2,j>-1 and cur<arr[j+1],arr[j+1]=4 i.e arr[3]=4

Step by Step O/p:
Iteration 1 : 1,4,2,7,10,1,2,4,4,7,1,2,1,10,1,2,4,1,2,7,10,1,2
Iteration 2 : 1,2,4,7,10,1,2,4,4,7,1,2,1,10,1,2,4,1,2,7,10,1,2
Iteration 3 : 1,2,4,7,10,1,2,4,4,7,1,2,1,10,1,2,4,1,2,7,10,1,2
Iteration 4 : 1,2,4,7,10,1,2,4,4,7,1,2,1,10,1,2,4,1,2,7,10,1,2
Iteration 5 : 1,1,2,4,7,10,2,4,4,7,1,2,1,10,1,2,4,1,2,7,10,1,2
Iteration 6 : 1,1,2,2,4,7,10,4,4,7,1,2,1,10,1,2,4,1,2,7,10,1,2
Iteration 7 : 1,1,2,2,4,4,7,10,4,7,1,2,1,10,1,2,4,1,2,7,10,1,2
Iteration 8 : 1,1,2,2,4,4,4,7,10,7,1,2,1,10,1,2,4,1,2,7,10,1,2
Iteration 9 : 1,1,2,2,4,4,4,7,7,10,1,2,1,10,1,2,4,1,2,7,10,1,2
Iteration 10 : 1,1,1,2,2,4,4,4,7,7,10,2,1,10,1,2,4,1,2,7,10,1,2
Iteration 11 : 1,1,1,2,2,2,4,4,4,7,7,10,1,10,1,2,4,1,2,7,10,1,2
Iteration 12 : 1,1,1,1,2,2,2,4,4,4,7,7,10,10,1,2,4,1,2,7,10,1,2
Iteration 13 : 1,1,1,1,2,2,2,4,4,4,7,7,10,10,1,2,4,1,2,7,10,1,2
Iteration 14 : 1,1,1,1,1,2,2,2,4,4,4,7,7,10,10,2,4,1,2,7,10,1,2
Iteration 15 : 1,1,1,1,1,2,2,2,2,4,4,4,7,7,10,10,4,1,2,7,10,1,2
Iteration 16 : 1,1,1,1,1,2,2,2,2,4,4,4,4,7,7,10,10,1,2,7,10,1,2
Iteration 17 : 1,1,1,1,1,1,2,2,2,2,4,4,4,4,7,7,10,10,2,7,10,1,2
Iteration 18 : 1,1,1,1,1,1,2,2,2,2,2,4,4,4,4,7,7,10,10,7,10,1,2
Iteration 19 : 1,1,1,1,1,1,2,2,2,2,2,4,4,4,4,7,7,7,10,10,10,1,2
Iteration 20 : 1,1,1,1,1,1,2,2,2,2,2,4,4,4,4,7,7,7,10,10,10,1,2
Iteration 21 : 1,1,1,1,1,1,1,2,2,2,2,2,4,4,4,4,7,7,7,10,10,10,2
Final Output:

Iteration 22 : 1,1,1,1,1,1,1,2,2,2,2,2,2,4,4,4,4,7,7,7,10,10,10

6a) When does the worst case of Quick sort happen and what is the worst case run time complexity in terms of big O?

When the pivot is taken as the smallest element of the array and the worst case run time complexity is $O(n^2)$

6b) When does the best case of bubble sort happen and what is the best case run time complexity in terms of big O?

The best case for the bubble sort is when the array is already sorted and the run time complexity for the bubble sort for the best case is O(n)

6c) What is the runtime complexity of Insertion sort in all 3 cases? Explain the situation which results in best, average and worst case complexity

Best case:
Time complexity is O(n).The best case is array is sorted in ascending order
Eg: arr=[1,2,3,4,5]

Average case: Is when the array is neither completely sorted in ascending order nor completely sorted in a descending order
Time complexity is $O(n^2)$
Eg: arr=[1,2,4,3,5]

Worst case : Is when the array is completely sorted in reverse order
Time complexity: $O(n^2)$
Eg: arr=[5,4,3,2,1]