Name: Shefali Mittal

Student ID:1002055262

Assignment Id: Home work #1

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 (15 points).

Approach:

1. Declare a dictionary with key pair values(key as number 1 to 10 and value as index)and having all numbers from 1 to 10.
2. Apply for loop in array and if value is found in dictionary, then remove item from dictionary.
3. At the end of the loop print the item remaining in dictionary and we get the missing item.

Pseudocode (Python code along with comments):

```
# Define hashmap

dict={1:0,2:1,3:2,4:3,5:4,6:5,7:6,8:7,9:8,10:9}

# Take array of number arr[i] as input

a=[7,6,4,8,3,10,2,5,1]

#Apply for loop in input array from 0 to n
for i,val in enumerate(a):
#If value is found in dictionary then pop that number from dictionary
    if val in dict:
        dict.pop(val)
 #print the output which will give the missing number
print(list(dict.keys())[0])
```

b) what is the worst case run time complexity of your suggested solution (5 points)

In each case the loop will run for number of items in array so complexity will be same in each case O(n) for the loop and dictionary has constant time complexity which is O(1) as in this case there are only 10 elements . **So worst case run time complexity as per big O notation 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 (15 points).

Approach:

1. Take an empty array
2. Apply loop in input array from 0 to length of array-1
3. Apply another loop in input array from index of first array+1 to length of array
4. If target sum is equal to sum of 2 numbers then append it in empty array

Pseudocode(comments along with Python code):

Let input array be a,sum be num_sum,empty array is b.

Sample input a=[2,3,4,7,6,8,1,5,8,6,4,3,2,6]

num_sum=5

b=[]

#Apply for loop in array a from 0 to length of array-1

for i in range(0,len(a)-1):

    #apply another for loop in array a from i+1 to length of array

    for j in range(i+1,len(a)):

```
    #check if sum of 2 numbers is equal to num_sum


     if num_sum==a[i]+a[j]:
#if yes then append the values in empty array b
        b.append([a[i],a[j]])
print(b)
```

Output: [[2, 3], [2, 3], [3, 2], [4, 1], [1, 4], [3, 2]]

b) what is the worst case run time complexity of your suggested solution(5 points)

It has nested loops so worst time complexity is $O(n^2)$

3. You are given an array of integers:

a) write a pseudo code to remove duplicates from your array (15 points).

Approach:

1. Take an empty array b
2. Apply for loop for n-1 elements
3. Check if input array element not in empty array then append that element in empty array
4. Replace input array with temporary array
5. Print the array a

Pseudocode(Python code along with comments):

```
#Sample input array
a=[9,9,0,5,6,7,9,8,8,7,0,4,7,55,55]
```

```
#Empty array
b=[]

#apply for loop in array a from 0 to length of array -1 elements
for i in range(0,len(a)-1):
#Check if the element in array a is not in b
    if a[i]not in b:
#if this above condition is true then append the values in array b
        b.append(a[i])
#Assign value of array b to array a
a=b
#print the output
print(a)
#Output=[9, 0, 5, 6, 7, 8, 4, 55]
```

b) what is the worst case run time complexity of your suggested solution(5 points)

The loop in array a has complexity O(n) and it has a lookup inside this loop for another array so **the worst run time complexity is $O(n^2)$ for this solution.**

4. you are given 2 sorted arrays:

a) write a pseudo code to find the median of the two sorted arrays (combined) (15 points).

Approach:

1. Declare a new empty array
2. Use 2 pointers for each array and compare the values of both array and will insert smaller values in new array.So by this we get sorted array at the end.
3. We will check the length of array and whichever array is not fully traversed we need to add the remaining elements of that array in new array
4. We get the sorted new array and will find the mid point of that array
5. If number of elements are even then median is average of 2 middle values
6. If number of elements are odd then median mid value.
7. Print the output

Pseudocode(Python code along with comments):

```
#Input array 1

a=[8,9,10,11,11,14,14,14]

#Input array2

b=[5,6,7,12,12,12]

#Empty array

c=[]


#Initialize i for use in while loop

i=0

#Initialize j for use in while loop

j=0

#Length of array a stored in variable l

l = len(a)

#Length of array b stored in variable m
```

```
m = len(b)


#While loop runs till i is less than l and j is less than m
while i<l and j<m:
#if element in array a is less than element in array b then append array a element in empty array c and increment i by 1


    if a[i]<b[j]:
        c.append(a[i])
        i=i+1
#otherwise append value of array b in array c and increment j by 1
    else:
        c.append(b[j])
        j=j+1
#Use below conditions to check if entire array traversed or not


#if after above loops are executed and value of j is less than length of b -1 then append remaining element to array c
if j<m-1:
    for x in range(j,m):
        c.append(b[x])
#if after above loops are executed and value of i is less than length of a -1 then append remaining element to array c


if i<l-1:
    for y in range(i,l):
        c.append(a[y])
```

```
#Find the middle element of array
mid=int(len(c)/2)
#Check if length of array is even then median is sum of 2 middle elements divided by 2
if len(c)%2==0:
    median=(c[mid-1]+c[mid])/2
#If length of array is odd then it is equal to middle element
else:
    median=c[mid]

#Print the value of median
print(median)
```

b) what is the worst case run time complexity of your suggested solution(5 points)

Worst run time complexity of this solution is O(m+n) and if each array has n number of elements so worst time complexity is O(2n) and **in terms of big O the worst run time complexity is O(n).**

5. Answer the following questions:

a) When does the worst case of Quick sort happen and what is the worst case run time complexity in terms of big O? (5 points)

The worst case of Quick sort occurs when array is sorted in ascending or descending order and we pick the first or last element as pivot.

The worst run time complexity in terms of bigO is $O(n^2)$

b)When does the best case of bubble sort happen and what is the best case run time complexity in terms of big O? (5 points)

When the array is already sorted then we have the best case for bubble sort. The run time complexity in terms of big(O) is O(n) for best case.

c) What is the runtime complexity of Insertion sort in all 3 cases? Explain the situation which results in best, average and worst case complexity. (10 points)

Run time complexity of Insertion sort in all 3 cases is :

1. Best case: O(n)
2. Average case: $O(n^2)$
3. Worst case: $O(n^2)$

In insertion sort there are 2 loops, the outer loop which traverse the array and count as 1,2,3…n elements of array. While the inner loop performs the swap. Run time complexity of outer loop is always O(n). The run time complexity of inner loop depends on number of swaps O(I).

- Situation which results in best case is when array is already sorted in increasing/ascending order as in this case there will be no swaps.
- Situation which results in average case is when we have an array with random elements ,we can say that array is partially sorted so we need to perform swap for half of the elements. Time complexity will be n(n-1)/4 which $O(n^2)$ in terms of bigO.
- Situation which results in worst case is when array is sorted in descending order (reversely sorted).So we need to traverse the entire array and perform swaps for each element.