Assignment1

January 25, 2019

0.0.1 Name: Amit Makashir

0.0.2 IU username: abmakash

0.0.3 Q1.

Illustrate the operation of Insertion-sort algorithm on array A=<31,41,59,26,41,58>.

```
def insertionSort(A):
    for i in range(len(A)):
        j = i
        while (j>0 and A[j-1]>A[j]):
            swap(A[j],A[j-1])
            j = j-1
    return A
```

0.0.4 Solution:

We start by looping over the array.

1. i=0, A[0]=31; As this is the first element of the array we will skip this element A=<31,41,59,26,41,58>
2. i=1, A[1]=41; Check if this element is less than the previous elements. As 41>31 we will skip to the next iteration A=<31,41,59,26,41,58>
3. i=2, A[2]=59; As 59 is greater than the previous elements we will skip to the next iteration A=<31,41,59,26,41,58>
4. i=3, A[3]=26; As 26<59 we will swap these elements. A=<31,41,26,59,41,58>

   As 26<41 we will swap again A=<31,26,41,59,41,58>

   As 26<31 we will swap again A=<26,31,41,59,41,58>
5. i=4, A[4]=41; As 41<59 we will swap these elements. A=<26,31,41,41,59,58>

   As 41==41, we will skip to next iteration A=<26,31,41,41,59,58>
6. i=5, A[5]=58; As 58<59 we will swap these elements. A=<26,31,41,41,58,59>

After looping through each element we get a sorted array.

0.0.5 Q2.

The input to the algorithm Unknown illustrated below is an array A of N numbers. (1) what is the output of the algorithm? (2) using big-O notation to show the running time of the algorithm. Input: Array A of N numbers; Unknown(A) for j = 1 to N-1 if A[N] < A[j] exchange A[j] and A[N] Output A[N];

0.0.6 Solution:

1. As we keep swapping elements at A[N] with A[j] whenever A[N]<A[j], the output will be the largest element in the array.
2. As we will iterate through every element in the array once, the running time of this algorithm is O(n)

0.0.7 Q3.

Given the input of k sorted arrays, each containing N distinct integers in increasing order, devise an O(kN) algorithm to output the number of integers that occur in each of the k input arrays

0.0.8 Solution:

```
# Where A is a 2-d array of k sorted arrays
def IntersectingElements(A):
    intersection = A[0]
    n = len(A[0])

    for i in range(1,len(A)):
        currentList = A[i]
        # Let i for looping intersection and j for looping currentList
        i = 0
        j = 0
        temp_intersection = []

        while j<n and i<len(intersection):
            if currentList[j] == intersection[i]:
                temp_intersection.append(currentList[j])
                i += 1
                j += 1
            elif currentList[j] < intersection[i]:
                j += 1
            else:
                i += 1

        intersection = temp_intersection

    return intersection
```

0.0.9 Q4.

An array A[1..n-1] contains all integers from 0 to n except two numbers. It would be easy to determine to the two missing numbers by using an auxiliary array B[0..n] to record which numbers appear in A. Here, we want to avoid the additional storage of B with the size O(n). Devise an algorithm to determine the two missing integers in O(n) time under this constraint. (Note, you can still use additional constant memory as temporary storage; you should use only comparison operation, but not other operations such as additions and multiplications.)

0.0.10 Solution:

Psuedocode for this problem is as following:
function getMissingNumbers(A): # Append -1 two times at the end. -1 denotes blank space A.append(-1) A.append(-1)

```
for i in (1,len(A)):
    val = A[i]
    while val != i and val != -1:
        temp = A[val]
        A[val] = val
        A[i] = temp
        val = A[i]

# the -1s are at the position of missing numbers, just print them
for i in (1,len(A)):
    if A[i] == -1:
        print(A[i])
```

0.0.11 Q5.

Let A[1..n] be an array of n distinct numbers. If i < j and A[i] > A[j], then the pair (i, j) is called an inversion of A. a) List the five inversions of the array <2, 3, 8, 6, 1>. b) What array with elements from the set {1, 2, . . . , n} has the most inversions? How many does it have? c) What is the relationship between the running time of insertion sort and the number inversions in the input array? Justify your answer.

0.0.12 Solution:

a) Following are indices of five inversion pairs of the array <2, 3, 8, 6, 1> :

1. (1,5)
2. (2,5)
3. (3,4)
4. (3,5)
5. (4,5)

b) A descending sorted array would have the most inversions. It will have n-1 inversions for the first element, n-2 for the second and 1 for the second last element. The solution would be 1+2+..n-1 which is **(n-1)(n)/2**

c) The running time of insertion sort would be directly proportional to the number of inversions. This is because, in insertion sort all the inversions would be swapped. If there are no inversions, there will be no swapping which means the array is already sorted.

0.0.13 Q6

You are given two sorted lists of size m and n. Devise an O(log m + log n) time algorithm for computing the kth smallest element in the union of the two list.

0.0.14 Solution:

Psuedocode for this problem is as following:

In [86]:
```
# https://stackoverflow.com/questions/4607945/how-to-find-the-kth-smallest-element-in
```

```
def findKsmallest(a,b,k,i,j,step):
    a_last = len(a)-1
    b_last = len(b)-1

    if k == 2:
        if b[j] > a[i+1]:
            return a[i+1]
        elif a[i] > b[j+1]:
            return b[j+1]

    if i<=0 or j<=0 or i>=a_last or j>=b_last:
        return max(a[i],b[j])

    if a[i] > b[j+1]:
        # increase j and decrease i
        dist_from_last = len(b)-1-j
        step = min(dist_from_last,step,i)
        i = i - step
        j = j + step
        return findKsmallest(a,b,k,i,j,int(step/2))

    elif b[j] > a[i+1]:
        # increase i and decrease j
        dist_from_last = len(a)-1-i
        step = min(dist_from_last,step,j)
        i = i + step
        j = j - step
        return findKsmallest(a,b,k,i,j,int(step/2))

    return max(a[i],b[j])

def splitArrays(a,b,k):
    m = len(a)
```

```
        n = len(b)

        if k%2 == 0:
            i = int(k/2)
            j = int(k/2)
        else:
            i = int(k/2) + 1
            j = int(k/2)

        return i,j


    # Main program

    # test case
    a = [1,2,2,3,5,7,8,19,30]
    b = [3,4,5,14,16,18]

    a.insert(0,float("-inf"))
    a.append(float("inf"))
    b.insert(0,float("-inf"))
    b.append(float("inf"))

    k = 3

    if k == 1:
        print(min(a[1],b[1]))
    else:
        i,j = splitArrays(a,b,k)
        print(findKsmallest(a,b,k,i,j,int(k/2)))
```

```
2
```

0.0.15 Q7

Given an array of numbers as the input, devise an algorithm to generate a random permutation of the array, such that each number has equal probability to be placed in each position in the output array.

https://en.wikipedia.org/wiki/Fisher%E2%80%93Yates_shuffle

0.0.16 Solution:

```
def RandomPermutation(A):
    for i in range(len(A)):
        # Generate a random integer in the range of length of A
        j = random integer such that i <= j < n
```

```
# swap the randomly generated index with current index
A[i],A[j] = A[j],A[i]
```