**Merge Two Sorted Array with o(1) space**

Approached 1: Insertion sort approached here

* Loop from the first position to the last of the first.
* Take the first element of b array always
* Check the a[i]>b[0]
  + If b[0] is greater then , swap with a[i];
* At the swap , we need to sort out the array b fully

Space complexity o(1)

Time Complexity o(n)\*o(mlogm)

**Approached 2: Gap Algorithm here**

Ceil function gap= (a+b)/2;

**Find the Departure Destination City : 23-05-2022**

**Algorithm**

* First to find out the starting position
  + Make a reverse pair of value as a key and key as a value
  + Now just use a for loop and check whether the reverse pair key is present in dataset or not
  + If it does not contains make it as a starting point.

After getting the starting point do this coding

String to =dataSet.get(startingPoint);

**while**(to!=**null**)

{

System.***out***.print(startingPoint+"->"+to+", ");

startingPoint=to;

to=dataSet.get(to);

}

**Find the Substring count in the string in java :24-05-2022**

**Question :**

**String a=”abcdabcghggabcaabcbcbcabc”**

**Count of abc in the string**

**Algorithm**

**Int i=3;**

**while**(i!=a.length())

{

**if**(a.charAt(i)=='c')

{

**if**(a.charAt(i-1)=='b' && a.charAt(i-2)=='a')

{

c++;

}

}

i++;

}

**Steps**

1. **While loop is executing when the variable I does not equal to length of array.**
2. **We need to see the character is C or not , if c please check the previous is b and next to previous to a**
3. **If the 2 point follow then increase the count.**
4. **After the condtion , increase the i++;**

**Longest Sub contiguous sum Subarray in Java : 27-05-2022**

**Algorithm**

Kandane’s Algorithm

Array = -2,-3,4,-1,2,1,5,-3

1. Take two variable meh and msf
2. Meh=0 , msf=minimum value of integer
3. For loop is executing form o to a.length
4. Take a every sum inside meh
5. Now check the condition meh is greater than msf
6. If found greater msf=meh
7. If meh<0
8. Then reset the msf

Code here

**int** meh=0;

**int** msf=Integer.***MIN\_VALUE***;

**for**(**int** i=0;i<a.length;i++)

{

meh=meh+a[i];

**if**(meh>msf)

{

msf=meh;

}

**if**(meh<0)

{

meh=0;

}

}

System.***out***.println(msf);

**Space complexity -o()**

**Time Complexity – 0(2)**

**Sort 0 and 1 in single time**