Understanding the problem statement

Given a number, we have to find the smallest number with same set of digits that is greater than the given number.

Input : 76348 Output : 76384

we also have 76834 greater than 76348 but it is not the smallest one possible.

Approach 1

```
Find all the permuations of the given number. We have n! numbers
possible. From all these numbers find
the numbers which are greater than the given number, and maintain the
track of the smallest number out of
all the greater numbers than given number.
```

Time Complexity : O(n!)

Space Complexity : O(1)

Approach 2 (Own approach)

1. Start from right and keep the track of maximum number till now.
2. while traversing from right,If we dont find a number smaller than the maximum number till now then maximum number is not possible.
3. If we find a smaller number, the take number which is just greater than the current number from the right side list, and replace it with current number. Also sort the entire right side array to current number in increasing order

Implementation

In [57]:
```
def nextBiggest(num,size):
    max_right = num[size-1]
    flag = False
    for i in range(size-1,0,-1):
        max_comp = num[i]
        if num[i] > max_right:
            j = i
            temp = num[i]
            while(j+1 < size and num[j+1] <= temp):
                num[j] = num[j+1]
                j = j+1
            num[j] = temp
        elif num[i] < max_right:
            flag = True
            temp = num[i]
            for k in range(i+1,size):
                if num[k] > num[i]:
                    num[i] = num[k]
```

```
                    num[k] = temp
                    break
            break
        max_right = max(max_right,max_comp)
    if flag:
        return num
    else:
        return "number not possible"
```

In [58]:
```
num = [2,1,8,7,6,5]
size = len(num)
nextBiggest(num,size)
# [2, 5, 1, 6, 7, 8]
```

Out[58]: [2, 5, 1, 6, 7, 8]

In [59]:
```
num = [9,7,6,5,4]
size = len(num)
nextBiggest(num,size)
# ans
# 'number not possible'
```

Out[59]: 'number not possible'

In [60]:
```
num = [1,2,3,4,5,6,7,8,4,9,8,7,6,5,4,3,2,1]
size = len(num)
nextBiggest(num,size)
# ans
# [1, 2, 3, 4, 5, 6, 7, 8, 5, 1, 2, 3, 4, 4, 6, 7, 8, 9]
```

Out[60]: [1, 2, 3, 4, 5, 6, 7, 8, 5, 1, 2, 3, 4, 4, 6, 7, 8, 9]

Approach 2 (RBR)

> Logic is same. Code wise optimization is done for the above approach

1. Find the first place from right where left digit is less than the right digit n[i-1] < n[i]. --> O(n)

   (We need not keep track of max_right as above because once we have seen n[i+1] > n[i+2],then only we go to n[i]. so its enought to check just n[i+1] since its any way greater than n[i+2])

2. Find the smallest digit larger than n[i-1] to the right.Lets call it n[k]. --> O(n)

3. Swap the two digits n[i-1],n[k]. -->O(n)

4. Sort all the numbers that are right to the n[i-1] in non decreasing order. --> O(n)

We can also just reverse because numbers are in decreasing order.

Time Complexity : O(n)

Space Complexity : O(1)

Implementation

In [3]:
```
def swap(arr,i,j):
    temp = arr[i]
```

```
    arr[i] = arr[j]
    arr[j] = temp
    return arr

def arr_reverse(arr,i,j):
    while(i<=j):
        temp = arr[i]
        arr[i] = arr[j]
        arr[j] = temp
        i+=1
        j-=1
    return arr

def nextBig(arr,size):
    for i in range(size-1,-1,-1):
        if (i == 0):
            print("number not possible")
            return None
        if arr[i] > arr[i-1]:
            break
    num = arr[i-1]
    smallest = i
    for j in range(i+1,size):
        if arr[j] > num and arr[j] < arr[smallest]:
            smallest = j
    arr = swap(arr,i-1,smallest)
    arr = arr_reverse(arr,i,size-1)
    return arr
```

In [4]:
```
num = [2,1,8,7,6,5]
size = len(num)
nextBig(num,size)
# [2, 5, 1, 6, 7, 8]
```

Out[4]: [2, 5, 1, 6, 7, 8]

In [5]:
```
num = [9,7,6,5,4]
size = len(num)
nextBig(num,size)
# ans
# 'number not possible'
```

number not possible

In [6]:
```
num = [1,2,3,4,5,6,7,8,4,9,8,7,6,5,4,3,2,1]
size = len(num)
nextBig(num,size)
# ans
# [1, 2, 3, 4, 5, 6, 7, 8, 5, 1, 2, 3, 4, 4, 6, 7, 8, 9]
```

Out[6]: [1, 2, 3, 4, 5, 6, 7, 8, 5, 1, 2, 3, 4, 4, 6, 7, 8, 9]

In [ ]: