Next Permutation

Saturday, November 20, 2021 12:48 AM

```
// Java program to find next greater
// number with same set of digits.
import java.util.Arrays;

public class nextGreater
{
   // Utility function to swap two digit
   static void swap(char ar[], int i, int j)
   {
      char temp = ar[i];
      ar[i] = ar[j];
      ar[j] = temp;
   }

   // Given a number as a char array number[],
   // this function finds the next greater number.
   // It modifies the same array to store the result
   static void findNext(char ar[], int n)
   {
      int i;

      // I) Start from the right most digit
      // and find the first digit that is smaller
      // than the digit next to it.
      for (i = n - 1; i > 0; i--)
      {
         if (ar[i] > ar[i - 1]) {
            break;
         }
      }

      // If no such digit is found, then all
      // digits are in descending order means
      // there cannot be a greater number with
      // same set of digits
      if (i == 0)
      {
         System.out.println("Not possible");
      }
      else
      {
         int x = ar[i - 1], min = i;

         // II) Find the smallest digit on right
         // side of (i-1)'th digit that is greater
         // than number[i-1]
         for (int j = i + 1; j < n; j++)
         {
            if (ar[j] > x && ar[j] < ar[min])
            {
               min = j;
            }
         }

         // III) Swap the above found smallest
         // digit with number[i-1]
         swap(ar, i - 1, min);

         // IV) Sort the digits after (i-1)
         // in ascending order
         Arrays.sort(ar, i, n);
         System.out.print("Next number with same" +
                          " set of digits is ");
         for (i = 0; i < n; i++)
            System.out.print(ar[i]);
      }
   }

   public static void main(String[] args)
   {
      char digits[] = { '5','3','4','9','7','6' };
      int n = digits.length;
      findNext(digits, n);
   }
}

Time Complexity: O(N*logN)
Auxiliary Space: O(1)
```

OUTPUT

```
Intial Arrays ::[1, 2, 3, 6, 5, 4]
l-->0
 k and l are : 2 --- 5
[1, 2, 4, 3, 5, 6]

Process finished with exit code 0
```

```
package dsaProblems;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class Next_Permutation_Main{
static List<Integer>nextPermutation(int N,int arr[]){
//codehere
List<Integer>list=new ArrayList<>();
int k=arr.length-1;


int n=arr.length;

for(k=n-2;k>=0;k--){
if(arr[k]<arr[k+1]){
break;
}
}
int l=0;
if(k<0){
//reversefromstarttoend
reverse(arr,0,n-1);

}
else{
for(int i=n-1;i>k;i--)
{
if(arr[i]>arr[k])
{
System.out.println("l-->"+l);

l=i;
break;

}
}
//swap
System.out.println("kandlare:"+k+"---"+l);

swap(arr,k,l);

//reverse
reverse(arr,k+1,n-1);

fillElements(arr,list);
printList(list);



}


return list;

}

private static void printList(List<Integer>list){
System.out.println(list);
```

}

Time Complexity: O(N*logN)
Auxiliary Space: O(1)

```
}

privatestaticvoidprintList(List<Integer>list){
System.out.println(list);
}

privatestaticvoidfillElements(int[]arr,List<Integer>list){
for(intele:arr)
list.add(ele);
}

privatestaticvoidreverse(int[]arr,inti,intj){

while(i<=j){
swap(arr,i,j);
i++;
j--;


}

}

privatestaticvoidswap(int[]arr,inti,intj){
inttemp=arr[i];
arr[i]=arr[j];
arr[j]=temp;

}

publicstaticvoidmain(String[]args){
intarr[]={1,2,3,6,5,4};
System.out.println("IntialArrays::"+Arrays.toString(arr));
nextPermutation(arr.length,arr);


}

}
```