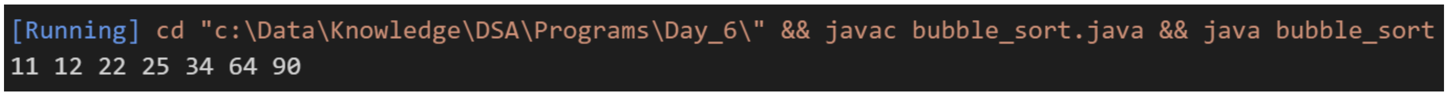
1. Bubble sort:

CODE:

```
import java.util.*;
class bubble_sort{
   static void bubblesort(int a[], int n){
      int i, j, t;
      boolean swapped;
      for(i=0;i<n-1;i++){
         swapped=false;
         for(j=0;j<n-i-1;j++){
            if(a[j]>a[j+1]){
               t=a[j];
               a[j]=a[j+1];
               a[j+1]=t;
               swapped=true;
            }
         }
         if(swapped==false)
            break;
      }
   }
   static void printarr(int a[], int len){
      int i;
      for(i=0;i<len;i++)
         System.out.print(a[i]+" ");
      System.out.println();
   }
   public static void main(String[] args){
      int a[]={64,34,25,12,22,11,90};
      int n=a.length;
      bubblesort(a,n);
      printarr(a,n);
   }
}
```

OUTPUT:



```
[Running] cd "c:\Data\Knowledge\DSA\Programs\Day_6\" && javac bubble_sort.java && java bubble_sort
11 12 22 25 34 64 90
```

Time Complexity: O(n^2)

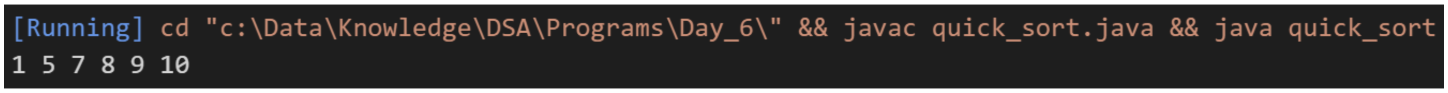
2. Quick sort:

CODE:

```
import java.util.*;
class quick_sort{
   static int parts(int[] a, int l, int h){
      int piv=a[h];
      int i=l-1;
      for(int j=l;j<=h-1;j++){
         if(a[j]<piv){
            i++;
```

```
                swap(a,i,j);
            }
        }
        swap(a,i+1,h);
        return i+1;
    }
    static void swap(int[] a, int i, int j){
        int t=a[i];
        a[i]=a[j];
        a[j]=t;
    }
    static void quicksort(int[] a, int l, int h){
        if(l<h){
            int pi=parts(a,l,h);
            quicksort(a,l,pi-1);
            quicksort(a,pi+1,h);
        }
    }
    public static void main(String[] args){
        int[] a={10,7,8,9,1,5};
        int n=a.length;
        quicksort(a,0,n-1);
        for(int v:a){
            System.out.print(v+" ");
        }
    }
}
```

OUTPUT:



```
[Running] cd "c:\Data\Knowledge\DSA\Programs\Day_6\" && javac quick_sort.java && java quick_sort
1 5 7 8 9 10
```

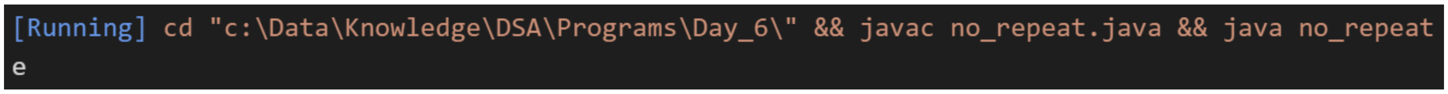
Time Complexity: O(n log n)

3. Non repeating character:

CODE:

```
import java.util.*;
class no_repeat{
    static final int MAX_CHAR=26;
    static char no_repeat_char(String s){
        int[] visit=new int[MAX_CHAR];
        Arrays.fill(visit,-1);
        for(int i=0;i<s.length();i++){
            if(visit[s.charAt(i)-'a']==-1)
                visit[s.charAt(i)-'a']=i;
            else
                visit[s.charAt(i)-'a']=-2;
        }
        int ind=Integer.MAX_VALUE;
        for(int i=0;i<MAX_CHAR;i++){
            if(visit[i]>=0)
        ind=Math.min(ind, visit[i]);
    }
    return (ind==Integer.MAX_VALUE ? '$' : s.charAt(ind));
  }
  public static void main(String[] args){
    String s="racecar";
    System.out.println(no_repeat_char(s));
  }
}
```

OUTPUT:



```
[Running] cd "c:\Data\Knowledge\DSA\Programs\Day_6\" && javac no_repeat.java && java no_repeat
e
```

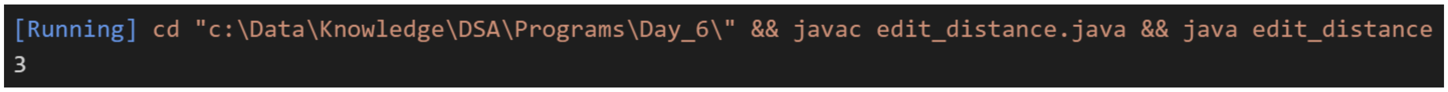
Time Complexity: O(n)

4. Edit distance:

CODE:

```
public class edit_distance{
  private static int mindist(String s1, String s2, int m, int n, int[][] memo){
    if(m==0) return n;
    if(n==0) return m;
    if(memo[m][n]!=-1) return memo[m][n];
    if(s1.charAt(m-1)==s2.charAt(n-1)){
      memo[m][n]=mindist(s1,s2,m-1,n-1,memo);
    }
    else{
      memo[m][n]=mindist(s1,s2,m-1,n-1,memo);
      int insert=mindist(s1,s2,m,n-1,memo);
      int replace=mindist(s1,s2,m-1,n-1,memo);
      int remove=(mindist(s1,s2,m-1,n,memo));
      memo[m][n]=1+Math.min(insert, Math.min(remove, replace));
    }
    return memo[m][n];
  }
  public static int mindis(String s1, String s2){
    int m=s1.length(), n=s2.length();
    int[][] memo=new int[m+1][n+1];
    for(int i=0;i<=m;i++){
      for(int j=0;j<=n;j++){
        memo[i][j]=-1;
      }
    }
    return mindist(s1,s2,m,n,memo);
  }
  public static void main(String[] args){
    String s1="GEEXSFRGEEKKS";
    String s2="GEEKSFORGEEKS";
    System.out.println(mindis(s1,s2));
  }
}
```

OUTPUT:



```
[Running] cd "c:\Data\Knowledge\DSA\Programs\Day_6\" && javac edit_distance.java && java edit_distance
3
```

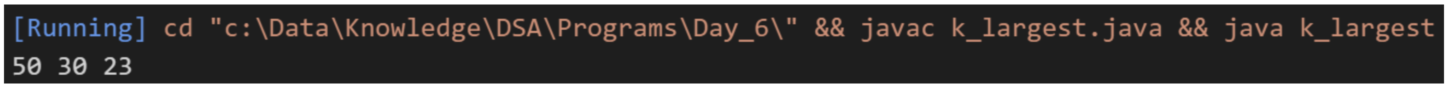
Time Complexity: O(m*n)

5. k largest elements:

CODE:

```
import java.util.*;
public class k_largest{
   public static List<Integer> klarge(int[] a, int k){
      PriorityQueue<Integer> minheap=new PriorityQueue<>(k);
      for(int i=0;i<k;i++){
         minheap.add(a[i]);
      }
      for(int i=k;i<a.length;i++){
         if(a[i]>minheap.peek()){
            minheap.poll();
            minheap.add(a[i]);
         }
      }
      List<Integer> r=new ArrayList<>();
      while(!minheap.isEmpty()){
         r.add(minheap.poll());
      }
      Collections.reverse(r);
      return r;
   }
   public static void main(String[] args){
      int[] a={1,23,12,9,30,2,50};
      int k=3;
      List<Integer> r=klarge(a,k);
      for(int el:r){
         System.out.print(el+" ");
      }
   }
}
```

OUTPUT:



```
[Running] cd "c:\Data\Knowledge\DSA\Programs\Day_6\" && javac k_largest.java && java k_largest
50 30 23
```

Time Complexity: O(n log k)

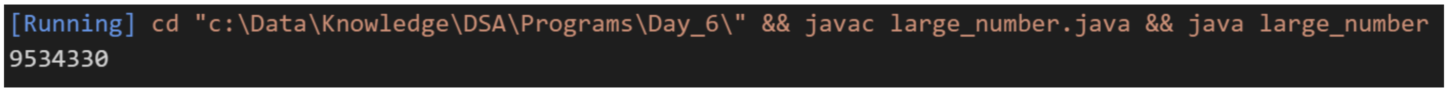
6. Form the largest element:

CODE:

```
import java.util.Arrays;
import java.util.Comparator;
public class large_number{
   public static String large_no(String[] a){
      Comparator<String> comp=(X,Y)->(X+Y).compareTo(Y+X);
      Arrays.sort(a,comp.reversed());
```

```
        if(a[0].equals("0")){
            return "0";
        }
        StringBuilder r=new StringBuilder();
        for(String n:a){
            r.append(n);
        }
        return r.toString();
    }
    public static void main(String[] args){
        String[] a={"3","30","34","5","9"};
        System.out.println(large_no(a));
    }
}
```

OUTPUT:



```
[Running] cd "c:\Data\Knowledge\DSA\Programs\Day_6\" && javac large_number.java && java large_number
9534330
```

Time Complexity: O(n log n)