Tanmai Kalisipudi

Big-O in a Program

DIRECTIONS: Classify the time efficiency of this program by writing the order of magnitude, in Big-O notation, of each statement in the blanks on the right. Then write the time efficiency in Big-O notation for each method.

```
import javax.swing.JOptionPane;

public class statpkg

{

   public static final int MAX = 10000;                                   // O(1)

   public static void main (String[] args)                                //main( ) O(n^3)

   {

      double [] list = new double[MAX];                                   // O(1)

      list = readArray(list);                                             //O(n^2)

      System.out.println("Mean = " +  mean(list));                        // O(n)

      System.out.println("Standard deviation = " + stDev(list));          // O(n^2)

      print(list);                                                        // O(n^3)

   }

   public static double[] readArray(double[] list)                        //readArray( ) O(n^2)
   {

      int n = 0;                                                          // O(1)

      double height = 0;                                                  // O(1)

      height = Double.parseDouble

            (JOptionPane.showInputDialog("Enter height:   -1 to stop")); // O(1)

       while(height > -1)                                                 // O(n)  ⎫
                                                                                   ⎪
       {                                                                           ⎪
                                                                                   ⎪
          if ( n >= list.length )                                         // O(1)  ⎪
                                                                                   ⎪
             list = resize(list, 2);                                      // O(n)  ⎪
                                                                                   ⎬ O(n^2)
          list[n] = height;                                               // O(1)  ⎪
                                                                                   ⎪
          n++;                                                            // O(1)  ⎪
                                                                                   ⎪
          height = Double.parseDouble                                              ⎪
                                                                                   ⎪
              (JOptionPane.showInputDialog("Enter height:   -1 to stop")); // O(1) ⎪
                                                                                   ⎪
       }                                                                           ⎭

      list = resize(list, n);                                             // O(n)

      return list;                                                        // O(1)

   }
public static double[] resize (double[] list, int n)                //resize() O(n)
{
   if (n == 2)                                                      // O(1)
      n = 2 * list.length;                                          // O(1)
   double [] newList = new double[n];                               // O(1)
   for (int i = 0; i < Math.min(n, list.length); i++)               // O(n)  }
       newList[i] = list[i];                                        // O(1)  }  O(n)
   return newList;                                                  // O(1)
}

public static double mean(double[] list)                            //mean( ) O(n)
{
   double sum = 0;                                                  // O(1)
   int n  = list.length;                                            // O(1)
   for(int i=0; i<n; i++)                                           // O(n)  }  O(n)
         sum += list[i];                                            // O(1)  }
   return (sum / n);                                                // O(1)
}

public static double stDev(double[] list)                           // stDev( ) O(n^2)
{
   double diff, sum = 0;                                            // O(1)
   int n = list.length;                                             // O(1)
   for(int i=0; i<n; i++)                                           // O(n)  }
   {                                                                         }
      diff = list[i]-mean(list);                                    // O(n)  }  O(n^2)
      sum  = sum + diff*diff;                                       // O(1)  }
   }
   return Math.sqrt(sum / (n - 1));                                 // O(1)
}

public static void print(double[] list)                             //print( ) O(n^3)
{
  int n = list.length;                                              // O(1)
  for(int i=0; i<n; i++)                                            // O(n)   }
     System.out.println ("[ " + list[i] + " ]  "                              }  O(n^3)
             + (list[i] – mean(list)) / stDev(list) );              // O(n^2) }
}
}
```

bigO from FCPS APCS C++ Workbook by Sally Bellacqua and Mary Johnson, adapated to JAVA