Guido Riccardi 758248

Vantaggiato Marianna 755470

Caso di studio di 'Integrazione e Test'

Informatica e Tecnologie per la Produzione del Software
A.A. 2023/2024

Caso di studio 'VaRi'

11 luglio 2024

Introduzione

Questo codice contiene una classe chiamata AdvancedApacheMathFunction che definisce due metodi statici per calcolare la media geometrica della somma dei quadrati di due array di numeri in virgola mobile

La prima funzione geometricMeanOfSumOfSquares Calcola la media geometrica della somma dei quadrati degli elementi di due array di uguale lunghezza, se i due array hanno lunghezza differente o sono vuoti lancia un'eccezione, una volta eseguito il calcolo esegue il controllo verificando che la media non sia negativa.

La seconda funzione geometricMeanOfSumOfSquaresVariableLength Calcola la media geometrica della somma dei quadrati degli elementi di due array di lunghezza variabile considerando 0 per le posizioni mancanti nel caso gli array abbiano lunghezze diverse.

Homework 1

BlackBox

1°Step – Understanding the requirements (what the program must do, inputs and outputs)

-**'geometricsMeanOfSumOfSquares'**

```
public static double geometricMeanOfSumOfSquares(double[] array1, double[]
array2) {
      if (array1.length != array2.length) {
      throw new IllegalArgumentException("Le lunghezze degli array devono
essere uguali");
      }

      double sumOfSquares = 0.0;

      for (int i = 0; i < array1.length; i++) {
      sumOfSquares += array1[i] * array1[i] + array2[i] * array2[i];
      }
```

```
    double meanOfSquares = sumOfSquares / array1.length;
    return Math.sqrt(meanOfSquares);
}
```

Il metodo calcola la media geometrica della somma dei quadrati degli elementi di due array con lunghezza uguale.

Il metodo prende in **input** due array di numeri reali (array1 e array2), controlla che abbiano uguali lunghezze e somma il quadrato di ogni elemento. Successivamente calcola la media di questi due array e restituisce come **output** la radice della media ottenuta.

-'geometricsMeanOfSumOfSquaresVariableLength'

```
public static double
geometricMeanOfSumOfSquaresVariableLength(double[] array1, double[]
array2) {
   int maxLength = Math.max(array1.length, array2.length);

   double sumOfSquares = 0.0;

   for (int i = 0; i < maxLength; i++) {
       double value1 = (i < array1.length) ? array1[i] : 0.0;
       double value2 = (i < array2.length) ? array2[i] : 0.0;
       sumOfSquares += value1 * value1 + value2 * value2;
   }

   double meanOfSquares = sumOfSquares / maxLength;
   return Math.sqrt(meanOfSquares);
}
```

Il metodo calcola la media geometrica della somma dei quadrati degli elementi di due array con lunghezza diversa.

Il metodo prende in **input** due array di numeri reali (array1 e array2), somma il quadrato di ogni elemento e imposta a 0 un elemento non presente. Successivamente calcola la media di questi due array e restituisce come **output** la radice della media ottenuta.

2°Step - Explore what the program does for various inputs

Abbiamo provato ad effettuare i test con vari input per coprire una vasta gamma di scenari: i test su valori positivi, negativi e zero sono tipici perché coprono situazioni comuni e limiti che possono influenzare il comportamento.

-**'geometricsMeanOfSumOfSquares'**

Condizioni di soluzione: I due vettori devono avere stessa lunghezza, i valori possono essere sia positivi che negativi ma non possono essere nulli (in questo caso viene generata un'eccezione)

Casi non risolvibili: La funzione genererà un'eccezione nel caso in cui:

- I due vettori abbiano lunghezza diversa
- Un vettore sia nullo
- Entrambi i vettori siano nulli

-**'geometricsMeanOfSumOfSquaresVariableLength'**

Condizioni di soluzione: I due vettori devono essere non nulli

Casi non risolvibili: La funzione genererà un'eccezione nel caso in cui:

- Un vettore sia nullo
- Entrambi i vettori siano nulli

3°Step – Explore inputs, outputs and identify partitions

-'geometricsMeanOfSumOfSquares'

Per identificare le partizioni del metodo `geometricMeanOfSumOfSquares`, possiamo considerare diverse combinazioni di input e osservare come il metodo risponde a ciascuna di esse. Le partizioni possono includere:

Individual Inputs

1. Array con lunghezza diversa:

- Input: `array1 = [1, 2], array2 = [3, 4, 5]`

- Output atteso: Eccezione `IllegalArgumentException` con il messaggio "Le lunghezze degli array devono essere uguali".

2. Array con lunghezza uguale:

- Input: `array1 = [1, 2, 3], array2 = [4, 5, 6]`

- Output atteso: Un valore numerico rappresentante la radice quadrata della media dei quadrati delle somme dei corrispondenti elementi di `array1` e `array2`.

3. Array con elementi positivi:

- Input: `array1 = [1, 2, 3], array2 = [4, 5, 6]`

- Output atteso: Un valore numerico positivo.

4. Array con elementi negativi:

- Input: `array1 = [-1, -2, -3], array2 = [-4, -5, -6]`

- Output atteso: Un valore numerico positivo, poiché la radice quadrata del quadrato di un numero negativo è sempre positiva.

5. Array con elementi misti (positivi e negativi):

- Input: `array1 = [-1, 2, -3], array2 = [4, -5, 6]`

- Output atteso: Un valore numerico positivo.

6. Array con elementi zero:

- Input: `array1 = [0, 0, 0], array2 = [0, 0, 0]`

- Output atteso: Un valore numerico pari a 0, poiché il quadrato di 0 è 0.

-'geometricsMeanOfSumOfSquaresVariableLength'

Per identificare le partizioni del metodo `geometricMeanOfSumOfSquaresVariableLength`, possiamo considerare diversi casi di input e raggrupparli in categorie omogenee. Ecco una suddivisione in partizioni:

1. Array con lunghezza diversa:

- Input: `array1 = [1, 2], array2 = [3, 4, 5]`

- Output atteso: Un valore numerico rappresentante la radice quadrata della media dei quadrati delle somme dei corrispondenti elementi di `array1` e `array2` usando come coefficiente della media il valore massimo tra le due lunghezze degli array.

2. Array con lunghezza uguale:

- Input: `array1 = [1, 2, 3], array2 = [4, 5, 6]`

- Output atteso: Un valore numerico rappresentante la radice quadrata della media dei quadrati delle somme dei corrispondenti elementi di `array1` e `array2`.

3. Array con elementi positivi:

- Input: `array1 = [1, 2, 3], array2 = [4, 5, 6,7]`

- Output atteso: Un valore numerico positivo.

4. Array con elementi negativi:

- Input: `array1 = [-1, -2, -3], array2 = [-4, -5, -6,-7]`

- Output atteso: Un valore numerico positivo, poiché la radice quadrata del quadrato di un numero negativo è sempre positiva.

5. Array con elementi misti (positivi e negativi e zero):

- Input: `array1 = [-1, 2, -3], array2 = [4, -5, 6,7,0]`

- Output atteso: Un valore numerico positivo.

6. Array con elementi zero:

- Input: `array1 = [0, 0, 0], array2 = [0, 0, 0]`

- Output atteso: Un valore numerico pari a 0, poiché il quadrato di 0 è 0.

7. Array con un solo elemento:

- Input: `array1 = [1], array2 = [2]`

- Output atteso: Un valore numerico rappresentante la radice quadrata della media dei quadrati delle somme dei corrispondenti elementi di `array1` e `array2`.

8. Array vuoti:

- Input: `array1 = [], array2 = []`

- Output atteso: Un valore 'NaN'.

Queste partizioni ci consentono di esplorare una vasta gamma di casi di test per garantire che il metodo funzioni correttamente in diversi scenari.

4°Step – Identify boundary cases

-'geometricsMeanOfSumOfSquares'

Per identificare i corner case del metodo `geometricMeanOfSumOfSquares`, possiamo definire l'on point come il punto in cui il metodo dovrebbe funzionare correttamente e restituire un risultato attendibile, mentre l'off point rappresenta situazioni in cui il metodo potrebbe comportarsi in modo imprevisto o restituire risultati non validi.

On Point:

1. *Entrambi gli array hanno la stessa lunghezza e contengono valori validi*: In questo caso, il metodo dovrebbe calcolare correttamente la radice quadrata della media dei quadrati delle somme e restituire un risultato attendibile.

2. *Entrambi gli array contengono lo stesso valore in ogni posizione*: Anche in questo caso, il metodo dovrebbe restituire un risultato corretto, poiché la somma dei quadrati degli stessi valori sarà uguale in entrambi gli array.

Off Point:

1. *Gli array hanno lunghezze diverse:* Se gli array `array1` e `array2` hanno lunghezze diverse, il metodo lancerà un'eccezione. Questo è un off point perché il metodo non è progettato per gestire situazioni in cui gli array hanno lunghezze diverse.

2. *Gli array contengono valori NaN o infiniti:* Se uno o entrambi gli array contengono valori NaN (Not a Number) o infiniti, il comportamento del metodo potrebbe essere imprevisto e potrebbe restituire un risultato non valido. Questo è un off point perché il metodo non è progettato per gestire queste situazioni in modo appropriato.

Identificando e considerando questi on point e off point, possiamo progettare e implementare test adeguati per verificare il comportamento del metodo in diverse situazioni.

-'geometricsMeanOfSumOfSquaresVariableLength'

Per identificare i corner case del metodo `geometricMeanOfSumOfSquaresVariableLength`, possiamo definire l'on point come il punto in cui il metodo dovrebbe funzionare correttamente e restituire un risultato attendibile, mentre l'off point rappresenta situazioni in cui il metodo potrebbe comportarsi in modo imprevisto o restituire risultati non validi.

On Point:

1. *Entrambi gli array contengono valori validi*: In questo caso, il metodo dovrebbe calcolare correttamente la radice quadrata della media dei quadrati delle somme e restituire un risultato attendibile.

2. *Entrambi gli array contengono lo stesso valore in ogni posizione*: Anche in questo caso, il metodo dovrebbe restituire un risultato corretto, poiché la somma dei quadrati degli stessi valori sarà uguale in entrambi gli array.

Off Point:

1. *Gli array contengono valori NaN o infiniti:* Se uno o entrambi gli array contengono valori NaN (Not a Number) o infiniti, il comportamento del metodo potrebbe essere imprevisto e potrebbe restituire un risultato non valido. Questo è un off point perché il metodo non è progettato per gestire queste situazioni in modo appropriato.

Identificando e considerando questi on point e off point, possiamo progettare e implementare test adeguati per verificare il comportamento del metodo in diverse situazioni.

5°Step – Devise test cases

-'geometricsMeanOfSumOfSquares'

T0 - `testArrayWithDifferentLengths_ThrowsException()`**,** esegue un test su due array di lunghezza differente e verifica che sollevi un eccezione.

T1 - `testOneEmptyArray()`, esegue un test su due array, di cui uno vuoto e verifica che sollevi un eccezione.

T2- `testArrayWithEqualLengths()`, esegue un test su due array con lunghezza uguale e con valori ammessi, e verifica che il risultato sia esatto.

T3- `testArrayWithNegativeValues()`, esegue un test su due array con lunghezza uguale e con valori negativi ma ammessi, e verifica che il risultato sia positivo.

T4- `testArrayWithMixValues()`, esegue un test su due array con lunghezza uguale e con valori misti (positivi e negativi), e verifica che il risultato sia positivo.

T5- `testArrayWithZeroValues()`, esegue un test su due array con lunghezza uguale e con tutti i valori '0' e verifica che il risultato sia zero.

T6- `testArrayWithSingleElement()`, esegue un test su due array con un singolo elemento e verifica che il risultato sia esatto.

T7-`testGeometricMeanOfSumOfSquares_WithEmptyArrays()`,esegue un test su due array vuoti e verifica che il risultato lanci un eccezione.

T8- `testArrayWithExtremeValues()`, esegue un test su due array di uguale lunghezza con dei numeri estremamente grandi, e verifica che il risultato sia corretto e non si verifichi overflow.

-'geometricsMeanOfSumOfSquaresVariableLength'

T1- `testEmptyArrays()`, esegue un test con due vettori nulli e controlla che il risultato si un NaN

T2- `testOneEmptyArrays()`, esegue un test con un vettore nullo

T3- `testSingleElementArrays()`, esegue un test con un vettori composti da un singolo elemento

T4- `testOneArraySingleElement()`, esegue un test con un vettore composto da un singolo elemento e l'altro da 3 elementi

T5- `testArrayWithEqualLengths()`, esegue un test con due vettori di pari lunghezza e verifica che il risultato ottenuto sia quello atteso

T6- `testArrayWithNegativeValues()`, esegue un test con vettori composti da elementi negativi e verifica che il risultato sia positivo

T7- `testArrayWithZeroValues()`, esegue un test con vettori di lunghezza diversa ma composti da elementi tutti uguali a zero

T8- `testArrayWithMixedValues()`, esegue un test con vettori composti da elementi positivi e negativi

T9- `testArrayWithMixedValuesEqualLenght()`, esegue un test con vettori composti da elementi positivi e negativi e lunghezza uguale

T10- `testArrayWithExtremeValues()`, esegue un test con vettori composti da elementi con un delta di valori elevato

6°Step – Automate test cases

-'geometricsMeanOfSumOfSquares'

T0 - testArrayWithDifferentLengths_ThrowsException()

```
@Test
public void testArrayWithDifferentLengths_ThrowsException() {
   double[] array1 = {1, 2};
   double[] array2 = {3, 4, 5};

   assertThrows(IllegalArgumentException.class, ()-> {
       geometricMeanOfSumOfSquares(array1, array2);
   });
}
```

T1 - testOneEmptyArray()

```
@Test
public void testOneEmptyArray() {
   double[] array1 = {1, 2, 3};
   double[] array2 = {};
   assertThrows(IllegalArgumentException.class, ()-> {
       geometricMeanOfSumOfSquares(array1, array2);
   });
```

```
}
```

T2- `testArrayWithEqualLengths()`

```
@Test
public void testArrayWithEqualLengths() {
   double[] array1 = {1, 2, 3};
   double[] array2 = {4, 5, 6};
   double expected = Math.sqrt((double) (1 + 16 + 4 + 25 + 9 + 36) / 3); // Calcolo manuale dell'output atteso
   double result = geometricMeanOfSumOfSquares(array1, array2);
   Assertions.assertEquals(expected, result, 0.01); // Verifica che il risultato ottenuto sia vicino all'output atteso con una tolleranza di 0.0001
}
```

T3- `testArrayWithNegativeValues()`

```
@Test
public void testArrayWithNegativeValues() {
   double[] array1 = {-1, -2, -3};
   double[] array2 = {-4, -5, -6};
   double result = geometricMeanOfSumOfSquares(array1, array2);
   assertTrue(result > 0); // Verifica che il risultato sia positivo
```

```
}
```

T4- `testArrayWithMixValues()`

```
@Test
public void testArrayWithMixValues() {
    double[] array1 = {-1, 2, -3};
    double[] array2 = {-4, -5, 6};
    double result = geometricMeanOfSumOfSquares(array1, array2);
    assertTrue(result > 0); // Verifica che il risultato sia positivo
}
```

T5- `testArrayWithZeroValues()`

```
@Test
public void testArrayWithZeroValues() {
    double[] array1 = {0, 0, 0};
    double[] array2 = {0, 0, 0};
    double result = geometricMeanOfSumOfSquares(array1, array2);
    assertEquals(0, result, 0.0001); // Verifica che il risultato sia
zero
}
```

T6- `testArrayWithSingleElement()`

```
@Test
public void testArrayWithSingleElement() {
   double[] array1 = {3};
   double[] array2 = {4};
   double expected = Math.sqrt(3*3 + 4*4); // Calcolo manuale
dell'output atteso
   double result = geometricMeanOfSumOfSquares(array1, array2);
   assertEquals(expected, result, 0.0001); // Verifica che il
risultato ottenuto sia vicino all'output atteso con una tolleranza di
0.0001
}
```

T7-`testGeometricMeanOfSumOfSquares_WithEmptyArrays()`

```
@Test
public void testGeometricMeanOfSumOfSquares_WithEmptyArrays() {
   double[] array1 = {};
   double[] array2 = {};

   assertThrows(IllegalArgumentException.class, ()-> {
       geometricMeanOfSumOfSquares(array1, array2);
```

```
    });
}
```

T8- `testArrayWithExtremeValues()`

```
@Test
public void testArrayWithExtremeValues() {
    double[] array1 = {18056, 3, 0};
    double[] array2 = {0, -4, -51045};
    double result = geometricMeanOfSumOfSquares(array1, array2);
    assertEquals(31260.257, result, 0.001);
}
```

-'geometricsMeanOfSumOfSquaresVariableLength'

T1- `testEmptyArrays()`

```
@Test
public void testEmptyArrays() {
    double[] array1 = {};
    double[] array2 = {};
```

```
    double result = geometricMeanOfSumOfSquaresVariableLength(array1,
array2);


    assertTrue(Double.isNaN(result)); // Verifica se il risultato è NaN
}
```

T2- `testOneEmptyArrays()`

```
@Test
public void testOneEmptyArray() {
    double[] array1 = {1, 2, 3};
    double[] array2 = {};
    double result = geometricMeanOfSumOfSquaresVariableLength(array1,
array2);
    assertEquals(2.160, result, 0.001); // Adjust the expected result
as per calculation
}
```

T3- `testSingleElementArrays()`

```
@Test
public void testSingleElementArrays() {
    double[] array1 = {4};
```

```
    double[] array2 = {3};

    double result = geometricMeanOfSumOfSquaresVariableLength(array1,
array2);

    assertEquals(5, result, 0.001); // Adjust the expected result as
per calculation

}
```

T4- `testOneArraySingleElement()`

```
@Test

public void testOneArraySingleElement() {

    double[] array1 = {3, 4, 5};

    double[] array2 = {6};

    double result = geometricMeanOfSumOfSquaresVariableLength(array1,
array2);

    assertEquals(5.35, result, 0.01); // Adjust the expected result as
per calculation

}
```

T5- `testArrayWithEqualLengths()`

```
@Test

public void testArrayWithEqualLengths() {
```

```
    double[] array1 = {1, 2, 3};

    double[] array2 = {4, 5, 6};

    double expected = Math.sqrt((double) (1 + 16 + 4 + 25 + 9 + 36) /
3); // Calcolo manuale dell'output atteso

    double result = geometricMeanOfSumOfSquaresVariableLength(array1,
array2);

    Assertions.assertEquals(expected, result, 0.01); // Verifica che il
risultato ottenuto sia vicino all'output atteso con una tolleranza di
0.0001

}
```

T6- `testArrayWithNegativeValues()`

```
@Test

public void testArrayWithNegativeValues() {

    double[] array1 = {-1, -2, -3};

    double[] array2 = {-4, -5};

    double result = geometricMeanOfSumOfSquaresVariableLength(array1,
array2);

    assertTrue(result > 0); // Verifica che il risultato sia positivo

}
```

T7- `testArrayWithZeroValues()`

```
@Test
public void testArrayWithZeroValues() {
   double[] array1 = {0, 0, 0};
   double[] array2 = {0, 0};
   double result = geometricMeanOfSumOfSquaresVariableLength(array1,
array2);
   assertEquals(0, result, 0.0001); // Verifica che il risultato sia
zero
}
```

T8- `testArrayWithMixedValues()`

```
@Test
public void testArrayWithMixedValues() {
   double[] array1 = {19, 3, 0};
   double[] array2 = {-4, 5};
   double result = geometricMeanOfSumOfSquaresVariableLength(array1,
array2);
   assertEquals(11.704, result, 0.001);
}
```

T9- `testArrayWithMixedValuesEqualLenght()`

```
@Test
public void testArrayWithMixedValuesEqualLenght() {
   double[] array1 = {19, 3, 0};
   double[] array2 = {0, -4, 5};
   double result = geometricMeanOfSumOfSquaresVariableLength(array1,
array2);
   assertEquals(11.704, result, 0.001); // Verifica che il risultato
il valore atteso
}
```

T10- `testArrayWithExtremeValues()`

```
@Test
public void testArrayWithExtremeValues() {
   double[] array1 = {18056, 3};
   double[] array2 = {0, -4, -51045};
   double result = geometricMeanOfSumOfSquaresVariableLength(array1,
array2);
   assertEquals(31260.257, result, 0.001);
}
```

7°Step – Augment the test suite with creativity and experience

I test precedentemente ideati coprono già tutti i possibili casi del lancio del metodo al variare dei valori inseriti.

Structural testing

Dopo aver effettuato lo specification-based testing, procediamo con lo structural testing. Abbiamo letto l'implementazione del codice e abbiamo eseguito la suite di test con un tool di code coverage, per capire quali linee di codice non fossero coperte da test.

```
package org.example;

public class AdvancedApacheMathFunction {

    public static double geometricMeanOfSumOfSquares(double[] array1, double[] array2) {
        if (array1.length != array2.length) {
            throw new IllegalArgumentException("Le lunghezze degli array devono essere uguali");
        }
        if (array1.length == 0) {
            throw new IllegalArgumentException("Gli array non devono essere vuoti");
        }

        double sumOfSquares1 = 0.0;
        double sumOfSquares2 = 0.0;


        for (int i = 0; i < array1.length; i++) {
            sumOfSquares1 += array1[i] * array1[i] ;
            sumOfSquares2 += array2[i] * array2[i] ;
        }


        double meanOfSquares = (sumOfSquares1 + sumOfSquares2) / (array1.length);

        if (meanOfSquares < 0){
            throw new IllegalArgumentException("La media geometrica deve essere non negativa");
        }
        return Math.sqrt(meanOfSquares);
    }


    public static double geometricMeanOfSumOfSquaresVariableLength(double[] array1, double[] array2) {
        int maxLength = Math.max(array1.length, array2.length);

        double sumOfSquares = 0.0;

        for (int i = 0; i < maxLength; i++) {
            double value1 = (i < array1.length) ? array1[i] : 0.0;
            double value2 = (i < array2.length) ? array2[i] : 0.0;
            sumOfSquares += value1 * value1 + value2 * value2;
        }

        double meanOfSquares = sumOfSquares / maxLength;
        return Math.sqrt(meanOfSquares);
    }

    public static void main(String[] args) {
        double[] array1 = {2.0, 3.0, 4.0};
        double[] array2 = {1.0, 5.0, 2.0};

        double resultEqualLength = geometricMeanOfSumOfSquares(array1, array2);
        System.out.println("Media geometrica della somma dei quadrati (uguali lunghezze): " + resultEqualLength);

        double[] array3 = {2.0, 3.0, 4.0, 5.0};
        double resultVariableLength = geometricMeanOfSumOfSquaresVariableLength(array1, array3);
        System.out.println("Media geometrica della somma dei quadrati (lunghezze diverse)  : " + resultVariableLength);
    }
}
```

Homework 2

Introduzione

L'obiettivo del nostro homework 2 è effettuare il property-based testing su entrambi i metodi della classe AdvancedApacheMathFunction che calcola il quadrato della somma di due array e poi calcola la radice della media tra i quadrati.

Property-Based Testing

Abbiamo quindi individuato per i nostri metodi le proprietà principali che deve soddisfare:

-**'geometricsMeanOfSumOfSquares'**

- **Test1:** le lunghezze delle array non devono essere diverse
- **Test2:** la media geometrica non deve essere negativa
- **Test3:** la funzione deve essere simmetrica
- **Test4:** il risultato deve rimanere invariato se vengono aggiunti zero (due vettori)
- **Test5:** il risultato deve rimanere invariato se vengono aggiunti zero (un vettore)

TEST 1

Per quanto riguarda il primo test andiamo a verificare la condizione necessaria per il metodo, quindi che le lunghezze degli array siano uguali. Testiamo per i vettori con grandezza minima 1 e massima 10000, con lunghezze diverse e testiamo che generi correttamente l'eccezione.

```
@Property

@Report(Reporting.GENERATED)

@StatisticsReport(format = Histogram.class)

public void testEqualLengthThrowsException(@ForAll @Size(min = 1, max
= 10000) double[] array1, @ForAll @Size(min = 1, max = 10000)
double[] array2) {
```

```
    Assume.that(array1.length != array2.length);


    assertThrows(IllegalArgumentException.class, () ->

AdvancedApacheMathFunction.geometricMeanOfSumOfSquares(array1,
array2),
            "Le lunghezze degli array devono essere uguali"
    );
    Statistics.collect(array1.length == array2.length);
}
```

Con le statistiche, andiamo a verificare che le condizioni siano vere al 100% ed eseguendo il test la statistica mostra invece che è vera nel 92.6% dei casi perchè nel resto dei casi vengono identificati dei casi limite.

```
timestamp = 2024-06-25T11:28:45.245247015, [AdvancedApacheMathFunctionPropertyBasedTest:testEqualLengthThrowsException] (926) statistics =
      # | label | count |
    ----|-------|-------|--------------------------------------------------------------------------------
      0 | false |   926 | ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■

timestamp = 2024-06-25T11:28:45.246967569, AdvancedApacheMathFunctionPropertyBasedTest:testEqualLengthThrowsException =
                           |----------------------jqwik----------------------
tries = 1000               | # of calls to property
checks = 926               | # of not rejected calls
generation = RANDOMIZED    | parameters are randomly generated
after-failure = SAMPLE_FIRST  | try previously failed sample, then previous seed
when-fixed-seed = ALLOW    | fixing the random seed is allowed
edge-cases#mode = MIXIN    | edge cases are mixed in
edge-cases#total = 64      | # of all combined edge cases
edge-cases#tried = 64      | # of edge cases tried in current run
seed = 5684088431977521760    | random seed to reproduce generated values
```

TEST 2

Per il secondo test andiamo a verificare che con un qualsiasi range di valori generati randomicamente, la media geometrica sia sempre positiva.

```
@Property
@Report(Reporting.GENERATED)
@StatisticsReport(format = Histogram.class)
public void testNonNegativity(@ForAll @NotEmpty @Size(min = 1, max = 10000) double[] array1) {
   // Generare array2 con la stessa lunghezza di array1
   double[] array2 = new double[array1.length];
   Random random = new Random();
   for (int i = 0; i < array2.length; i++) {
       array2[i] = random.nextDouble();
   }

   // Stampare gli array per il debug
   System.out.println("array1: " + Arrays.toString(array1));
   System.out.println("array2: " + Arrays.toString(array2));
```

```
    double result =
AdvancedApacheMathFunction.geometricMeanOfSumOfSquares(array1,
array2);

    assertTrue(result >= 0, "La media geometrica deve essere non
negativa");

    Statistics.collect(result>=0);

}
```

Per quanto riguarda le statistiche, andiamo a verificare che la condizione sia vera al 100%, ed eseguendo il test possiamo notare che è vero.

```
        # | label | count |
     -----|-------|-------|-------------------------------------------------------------------------------
        0 |  true |  1000 | ################################################################################

 timestamp = 2024-06-25T11:42:20.968121, AdvancedApacheMathFunctionPropertyBasedTest:testNonNegativity =
                              |-----------------------jqwik-----------------------
 tries = 1000                 | # of calls to property
 checks = 1000                | # of not rejected calls
 generation = RANDOMIZED      | parameters are randomly generated
 after-failure = SAMPLE_FIRST | try previously failed sample, then previous seed
 when-fixed-seed = ALLOW      | fixing the random seed is allowed
 edge-cases#mode = MIXIN      | edge cases are mixed in
 edge-cases#total = 7         | # of all combined edge cases
 edge-cases#tried = 7         | # of edge cases tried in current run
 seed = 5985910924113616348   | random seed to reproduce generated values
```

TEST 3

In questo test andiamo a verificare che la funzione sia sempre simmetrica, cioè che valga la proprietà commutativa.

```
@Property

@Report(Reporting.GENERATED)

@StatisticsReport(format = Histogram.class)

public void testSymmetry(@ForAll @NotEmpty @Size(min = 1, max =
10000) double[] array1) {

   // Generare array2 con la stessa lunghezza di array1

   double[] array2 = new double[array1.length];

   Random random = new Random();

   for (int i = 0; i < array2.length; i++) {

       array2[i] = random.nextDouble();

   }

   double result1 =
AdvancedApacheMathFunction.geometricMeanOfSumOfSquares(array1,
array2);

   double result2 =
AdvancedApacheMathFunction.geometricMeanOfSumOfSquares(array2,
array1);
```

```
    assertEquals(result1, result2, "La funzione deve essere
simmetrica");

    Statistics.collect(result1==result2);

}
```

Ci aspettiamo che le statistiche restituiscano un risultato del 100%, perchè per l'operazione matematica di somma e moltiplicazione vale la proprietà commutativa.

```
timestamp = 2024-06-25T11:46:04.654041400, [AdvancedApacheMathFunctionPropertyBasedTest:testSymmetry] (1000) statistics =
      # | label | count |
    -----|-------|-------|--------------------------------------------------------------------------------
      0 |  true |  1000 | ################################################################################

timestamp = 2024-06-25T11:46:04.683040500, AdvancedApacheMathFunctionPropertyBasedTest:testSymmetry =
                            |----------------------jqwik----------------------
tries = 1000                | # of calls to property
checks = 1000               | # of not rejected calls
generation = RANDOMIZED     | parameters are randomly generated
after-failure = SAMPLE_FIRST | try previously failed sample, then previous seed
when-fixed-seed = ALLOW     | fixing the random seed is allowed
edge-cases#mode = MIXIN     | edge cases are mixed in
edge-cases#total = 7        | # of all combined edge cases
edge-cases#tried = 7        | # of edge cases tried in current run
seed = 7721927669830893116  | random seed to reproduce generated values
```

TEST 4

Per il quarto test, abbiamo testato che l'aggiunta di un array di zeri ai due normali array, restituisca un risultato uguale, e che quindi l'aggiunta di un terzo array di zeri ai due già analizzati, non influenzi il risultato.

```
@Property

@Report(Reporting.GENERATED)

@StatisticsReport(format = Histogram.class)
```

```
public void testZeroAdditionTwoArrays(@ForAll @NotEmpty @Size(min
= 1, max = 10) double[] array1) {
   // Generare array2 con la stessa lunghezza di array1
   double[] array2 = new double[array1.length];
   Random random = new Random();
   for (int i = 0; i < array1.length; i++) {
       array2[i] = random.nextDouble();
   }

   double result1 =
AdvancedApacheMathFunction.geometricMeanOfSumOfSquares(array1,
array2);

   double[] zeros = new double[array1.length];

   for (int i = 0; i < array1.length; i++) {
       zeros[i] = 0.0;
   }

   double sumOfSquares1 = 0.0;
   double sumOfSquares2 = 0.0;
```

```
    double sumOfSquares3 = 0.0;


    for (int i = 0; i < array1.length; i++) {
        sumOfSquares1 += (array1[i] * array1[i]);
        sumOfSquares2 += (array2[i] * array2[i]);
        sumOfSquares3 += (zeros[i] * zeros[i]);
    }
    double result2 = Math.sqrt((sumOfSquares2 + sumOfSquares1 +
sumOfSquares3)/(array1.length));


    assertEquals(result1, result2, "Aggiungere zero non deve cambiare
il risultato");
    Statistics.collect(result1==result2);
}
```

Ci aspettiamo che le statistiche restituiscano un risultato del 100%, perchè per le operazioni matematiche, aggiungendo zero, il risultato non cambia.

```
timestamp = 2024-06-25T11:50:26.508369500, [AdvancedApacheMathFunctionPropertyBasedTest:testZeroAdditionTwoArrays] (1000) statistics =
      # | label | count |
    -----|-------|-------|--------------------------------------------------------------------------------
      0 |  true |  1000 | ################################################################################

timestamp = 2024-06-25T11:50:26.565904700, AdvancedApacheMathFunctionPropertyBasedTest:testZeroAdditionTwoArrays =
                            |----------------------jqwik----------------------
tries = 1000                | # of calls to property
checks = 1000               | # of not rejected calls
generation = RANDOMIZED     | parameters are randomly generated
after-failure = SAMPLE_FIRST | try previously failed sample, then previous seed
when-fixed-seed = ALLOW     | fixing the random seed is allowed
edge-cases#mode = MIXIN     | edge cases are mixed in
edge-cases#total = 7        | # of all combined edge cases
edge-cases#tried = 7        | # of edge cases tried in current run
seed = -1551814881518183222 | random seed to reproduce generated values
```

TEST 5

Per il quinto test, abbiamo deciso di testare un altro caso di somma di zeri, ma in questo caso abbiamo un solo vettore a cui viene sommato il vettore di zeri, quindi ci ritroviamo nel caso precedente.

```
@Property

@Report(Reporting.GENERATED)

@StatisticsReport(format = Histogram.class)

public void testZeroAdditionOneArray(

        @ForAll @NotEmpty @Size(min = 1, max = 10000) double[] array1)
{



   double[] zeros = new double[array1.length];



   double sumOfSquares1= 0.0;
```

```
    for (int i = 0; i < array1.length; i++) {
        sumOfSquares1 += array1[i] * array1[i] ;
        zeros[i]=0.0;
    }
    double meanOfSquares = sumOfSquares1 / array1.length;
    double resultWithZeros =
AdvancedApacheMathFunction.geometricMeanOfSumOfSquares(array1,
zeros);
    double resultWithouthZeros = Math.sqrt(meanOfSquares);
    System.out.println("media array1: " + meanOfSquares);
    System.out.println("media con zeri: " + resultWithZeros);

    assertEquals(resultWithZeros, resultWithouthZeros, 1e-10,
"L'aggiunta di zeri non deve cambiare il risultato");
    Statistics.collect(resultWithZeros == resultWithouthZeros);
}
```

Per le statistiche, come nel caso precedente ci aspettiamo di avere un risultato al 100%, ed è quello che abbiamo ricavato, perchè come nel caso precedente, l'aggiunta di zeri non cambia il risultato.

```
timestamp = 2024-06-25T11:54:02.386560300, [AdvancedApacheMathFunctionPropertyBasedTest:testZeroAdditionOneArray] (1000) statistics =
       # | label | count |
    -----|-------|-------|-----------------------------------------------------------------------------
       0 |  true |  1000 | ################################################################################

timestamp = 2024-06-25T11:54:02.414564, AdvancedApacheMathFunctionPropertyBasedTest:testZeroAdditionOneArray =
                            |----------------------jqwik----------------------
tries = 1000                | # of calls to property
checks = 1000               | # of not rejected calls
generation = RANDOMIZED     | parameters are randomly generated
after-failure = SAMPLE_FIRST  | try previously failed sample, then previous seed
when-fixed-seed = ALLOW     | fixing the random seed is allowed
edge-cases#mode = MIXIN     | edge cases are mixed in
edge-cases#total = 7        | # of all combined edge cases
edge-cases#tried = 7        | # of edge cases tried in current run
seed = 7515202141499245924    | random seed to reproduce generated values
```

-'geometricsMeanOfSumOfSquaresVariableLength'

- **Test1:** la media geometrica non deve essere negativa
- **Test2:** la funzione deve essere simmetrica
- **Test3:** il risultato deve rimanere invariato se vengono aggiunti zero (due vettori)

TEST 1

Per questo test, abbiamo testato che anche con lunghezza differente, il risultato della media sia sempre positivo.

```
@Property

@Report(Reporting.GENERATED)

@StatisticsReport(format = Histogram.class)

public void testVariableLengthNonNegativity(@ForAll @NotEmpty
double[] array1, @ForAll @NotEmpty double[] array2) {
```

```
   double result =
AdvancedApacheMathFunction.geometricMeanOfSumOfSquaresVariableLength(a
rray1, array2);

   assertTrue(result >= 0, "La media geometrica deve essere non
negativa anche con lunghezze variabili");

   Statistics.collect(result>=0);

}
```

Per quanto riguarda le statistiche, andiamo a verificare che la condizione sia vera al 100%, ed eseguendo il test possiamo notare che è vero perchè partiamo già da dei quadrati, quindi da numeri positivi.

```
timestamp = 2024-06-25T11:56:21.938305534, [AdvancedApacheMathFunctionPropertyDifferentLength:testVariableLengthNonNegativity] (1000) statisti
      # | label | count |
    ----|-------|-------|-----------------------------------------------------------------------------
      0 |  true |  1000 | ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■

timestamp = 2024-06-25T11:56:21.957073320, AdvancedApacheMathFunctionPropertyDifferentLength:testVariableLengthNonNegativity =
                           |----------------------jqwik----------------------
tries = 1000               | # of calls to property
checks = 1000              | # of not rejected calls
generation = RANDOMIZED    | parameters are randomly generated
after-failure = SAMPLE_FIRST | try previously failed sample, then previous seed
when-fixed-seed = ALLOW    | fixing the random seed is allowed
edge-cases#mode = MIXIN    | edge cases are mixed in
edge-cases#total = 49      | # of all combined edge cases
edge-cases#tried = 29      | # of edge cases tried in current run
seed = 4348761430285227230 | random seed to reproduce generated values
```

TEST 2

In questo test andiamo a verificare che la funzione sia sempre simmetrica, cioè che valga la proprietà commutativa

```
@Property

@Report(Reporting.GENERATED)

@StatisticsReport(format = Histogram.class)
```

```
public void testVariableLengthSymmetry(@ForAll double[] array1, @ForAll
double[] array2) {

   double result1 =
AdvancedApacheMathFunction.geometricMeanOfSumOfSquaresVariableLength(array1,
array2);

   double result2 =
AdvancedApacheMathFunction.geometricMeanOfSumOfSquaresVariableLength(array2,
array1);



   assertEquals(result1, result2, "La funzione deve essere simmetrica anche con
lunghezze variabili");

   Statistics.collect(result1==result2);

}
```

Ci aspettavamo un risultato del 100%, invece su 1000 statistiche abbiamo riscontrato 3 casi in cui la condizione era falsa.

```
timestamp = 2024-06-25T11:59:01.427408453, [AdvancedApacheMathFunctionPropertyDifferentLength:testVariableLengthSymmetry] (1000) statistics =
      # | label | count |
    ----|-------|-------|--------------------------------------------------------------------------------
      0 | false |     3 |
      1 |  true |   997 | ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒

timestamp = 2024-06-25T11:59:01.444753486, AdvancedApacheMathFunctionPropertyDifferentLength:testVariableLengthSymmetry =
                            |----------------------jqwik----------------------
tries = 1000                | # of calls to property
checks = 1000               | # of not rejected calls
generation = RANDOMIZED     | parameters are randomly generated
after-failure = SAMPLE_FIRST | try previously failed sample, then previous seed
when-fixed-seed = ALLOW     | fixing the random seed is allowed
edge-cases#mode = MIXIN     | edge cases are mixed in
edge-cases#total = 64       | # of all combined edge cases
edge-cases#tried = 60       | # of edge cases tried in current run
seed = -7217363381973062997 | random seed to reproduce generated values
```

TEST 3

Questo test controlla che aggiungendo un vettore composto da zeri (ma la cui lunghezza non sia maggiore della lunghezza massima tra vettore 1 e vettore 2) la media non sia modificata

```
@Property

@Report(Reporting.GENERATED)

@StatisticsReport(format = Histogram.class)

public void testVariableLengthZeroAdditionTwoArrays(@ForAll @NotEmpty double[]
array1,@ForAll @NotEmpty double[] array2) {

   double result1 =
AdvancedApacheMathFunction.geometricMeanOfSumOfSquaresVariableLength(array1,
array2);



   int maxLengthA = Math.max(array1.length, array2.length);

   double[] zeros = new double[maxLengthA];



   for (int i = 0; i < zeros.length; i++) {

       zeros[i] = 0.0;

   }



   double sumOfSquares1 = 0.0;



   for (int i = 0; i < maxLengthA; i++) {
```

```
        double value1 = (i < array1.length) ? array1[i] : 0.0;
        double value2 = (i < array2.length) ? array2[i] : 0.0;
        double value3 = (i < zeros.length ) ? zeros[i] : 0.0;
        sumOfSquares1 += value1 * value1 + value2 * value2 + value3 * value3;
    }

    double meanOfSquares = sumOfSquares1 / maxLengthA;
    double result2 = Math.sqrt(meanOfSquares);

    assertEquals(result1, result2, "Aggiungere zero non deve cambiare il
risultato");
    Statistics.collect(result1==result2);
}
```

Ci aspettiamo che il risultato delle statistiche sia il 100%, perchè aggiungendo degli zeri il risultato non cambia.

```
timestamp = 2024-06-25T12:03:31.669059195, [AdvancedApacheMathFunctionPropertyDifferentLength:testVariableLengthZeroAdditionTwoArrays] (1000)
       # | label | count |
    -----|-------|-------|-----------------------------------------------------------------------------
       0 |  true |  1000 | ████████████████████████████████████████████████████████████████████████████

timestamp = 2024-06-25T12:03:31.689854122, AdvancedApacheMathFunctionPropertyDifferentLength:testVariableLengthZeroAdditionTwoArrays =
                            |----------------------jqwik----------------------
tries = 1000                | # of calls to property
checks = 1000               | # of not rejected calls
generation = RANDOMIZED     | parameters are randomly generated
after-failure = SAMPLE_FIRST  | try previously failed sample, then previous seed
when-fixed-seed = ALLOW     | fixing the random seed is allowed
edge-cases#mode = MIXIN     | edge cases are mixed in
edge-cases#total = 49       | # of all combined edge cases
edge-cases#tried = 39       | # of edge cases tried in current run
seed = 3049341096797419271    | random seed to reproduce generated values
```

Risultati dei test

Unit Test 'geometricMeanOfSumOfSquares'

ID	T0
Descrizione	Verifica che due array di lunghezza differente generino eccezione
Input	due array di double array1={1,2},array2={3,4,5}
Output	IllegalArgument exception
Stato	
Nome del tester	Marianna Vantaggiato
Note	

ID	T1F
Descrizione	Verifica che se un array è vuoto, venga generata un'eccezione
Input	due array di double array1={1,2,3},array2={}
Output	Errore
Stato	
Nome del tester	Marianna Vantaggiato
Note	Con questo test, ci siamo accorti che era necessaria che venisse gestita l'eccezione nel codice, nel caso in cui un array fosse vuoto

ID	T1
Descrizione	Verifica se un array è vuoto e se viene generata correttamente un'eccezione
Input	due array di double array1={1,2,3},array2={}
Output	IllegalArgument exception
Stato	
Nome del tester	Marianna Vantaggiato
Note	Aggiunta l'eccezione nel codice, il test funziona

ID	T2
Descrizione	Verifica il risultato con due vettori di pari lunghezza con una tolleranza di 0.01
Input	array1 = {1, 2, 3} array2 = {4, 5, 6}
Output	5,507
Stato	
Nome del tester	Guido Riccardi
Note	

ID	T3
Descrizione	Verifica che usando come input valori negativi restituisca sempre un valore positivo
Input	array1 = {-1, -2, -3} array2 = {-4, -5, 6}
Output	5,507
Stato	
Nome del tester	Guido Riccardi
Note	

ID	T4
Descrizione	Verifica che utilizzando valori sia positivi che negativi restituisca sempre valori positivi
Input	array1 = {-1, 2, -3} array2 = {-4, -5, 6}
Output	5,507
Stato	
Nome del tester	Guido Riccardi
Note	

ID	T5
Descrizione	Verifica che inserendo vettori con elementi pari a zero restituisca zero
Input	array1 = {0,0,0} array2 = {0,0,0}
Output	0
Stato	
Nome del tester	Guido Riccardi
Note	

ID	T6
Descrizione	Verifica che inserendo dei vettori composti da singoli elementi restituisca il risultato atteso
Input	array1 = {3} array2 = {4}
Output	25
Stato	
Nome del tester	Guido Riccardi
Note	

ID	T7F
Descrizione	Verifica che se gli array sono vuoti, venga generata un'eccezione
Input	due array di double array1={},array2={}
Output	Errore
Stato	
Nome del tester	Marianna Vantaggiato
Note	Con questo test, ci siamo accorti che era necessaria che venisse gestita l'eccezione nel codice, nel caso in cui i vettori fossero vuoti

ID	T7
Descrizione	Verifica che se gli array sono vuoti, venga generata un'eccezione
Input	due array di double array1={},array2={}
Output	IllegalArgument exception
Stato	
Nome del tester	Marianna Vantaggiato
Note	Dopo aver aggiunto l'eccezione nel codice, il test viene superato

ID	T8
Descrizione	Verifica se la funzione svolge correttamente il suo compito, anche con numeri grandi
Input	due array di double: array1={18056, 3, 0}, array2={0, -4, -51045}
Output	ci aspettavamo il risultato: 31260.257, e la funzione restituisce proprio quello
Stato	
Nome del tester	Marianna Vantaggiato
Note	

Unit Test ‘geometricMeanOfSumOfSquaresVariableLength’

ID	T1L
Descrizione	Verifica se inserendo due array vuoti, la funzione restituisca un valore 'NaN'
Input	due array di double array1={},array2={}
Output	NaN'
Stato	
Nome del tester	Marianna Vantaggiato
Note	

ID	T2L
Descrizione	Verifica che inserendo un array vuoto, la funzione svolga il suo lavoro correttamente
Input	due array di double array1={1, 2, 3},array2={}
Output	Ci aspettavamo 2.160 come risultato, e corrisponde al valore restituito
Stato	
Nome del tester	Marianna Vantaggiato
Note	

ID	T3L
Descrizione	Verifica che inserendo dei vettori composti da singoli elementi restituisca il risultato atteso
Input	due array di double array1 = {4} array2 = {3}
Output	5
Stato	
Nome del tester	Guido Riccardi
Note	

ID	T4L
Descrizione	Verifica che inserendo un vettore composto da singoli elementi restituisca il risultato atteso
Input	due array di double array1 = {3,4,5} array2 = {6}
Output	5,163
Stato	
Nome del tester	Guido Riccardi
Note	

ID	T5L
Descrizione	Verifica che la funzione svolga correttamente il suo lavoro, anche quando gli array hanno la stessa lunghezza
Input	due array di double array1={1, 2, 3},array2={4,5,6}
Output	Ci aspettavamo '30.3' come risultato, e la funzione restituisce correttamente il valore
Stato	
Nome del tester	Marianna Vantaggiato
Note	

ID	T6L
Descrizione	Verifica che inserendo solo numeri negativi, la funzione restituisca un numero positivo
Input	due array di double array1={-1, -2, -3},array2={-4,-5}
Output	Un numero positivo
Stato	
Nome del tester	Marianna Vantaggiato
Note	

ID	T7L
Descrizione	Verifica che inserendo due array di lunghezza differente composti da zero restituisca il risulatto atteso
Input	due array di double array1={0,0,0},array2={0,0,0}
Output	0
Stato	
Nome del tester	Guido Riccardi
Note	

ID	T8L
Descrizione	Verifica che inserendo due array composti da valori positi e negativi restituisca il risultato atteso
Input	due array di double array1={19, 3, 0},array2={-4, 5}
Output	11,704
Stato	
Nome del tester	Guido Riccardi
Note	

ID	T9L
Descrizione	Verifica che due array di lunghezza uguale e valori misti restituiscano un giusto risultato
Input	due array di double array1={19, 3, 0},array2={0, -4, 5}
Output	Ci aspettavamo '11.704' ed il risultato restituito era quello atteso
Stato	
Nome del tester	Marianna Vantaggiato
Note	

ID	T10L
Descrizione	Verifica che due array con valori estremi restituiscano un giusto risultato
Input	due array di double array1={18056, 3},array2={0, -4, -51045}
Output	Risultato atteso: 31260.257, il risultato ottenuto corrisponde.
Stato	
Nome del tester	Marianna Vantaggiato
Note	

Property based test ‘geometricMeanOfSumOfSquares’

ID	T1P
Descrizione	Il test genera 1000 casi in cui controlla che se la lunghezza dei due array sia diversa venga lanciata un'eccezione
Input	Due array di double di lunghezza uguale generata tra 1 e 10000
Output	IllegalArgumentException
Stato	
Nome del tester	Guido Riccardi
Note	

ID	T2P
Descrizione	Il metodo genera 1000 casi in cui vengono definiti due array di lunghezza uguale e lanciata la funzione, non venga calcolata una media geometrica negativa
Input	Due array di double di lunghezza uguale generata tra 1 e 10000
Output	true, perchè viene verificato che la Media geometrica non è negativa
Stato	
Nome del tester	Marianna Vantaggiato
Note	

ID	T3P
Descrizione	Il metodo genera 1000 casi in cui vengono definiti due array di lunghezza uguale e lanciata la funzione, verifichi se la funzione è simmetrica
Input	Due array di double di lunghezza uguale generata tra 1 e 10000
Output	true, perchè viene verificato che la funzione è simmetrica
Stato	
Nome del tester	Marianna Vantaggiato
Note	

ID	T4P
Descrizione	Il metodo controlla che aggiungendo un array di zeri di lunghezza pari a quella di array1 e array 2 non modifichi la media
Input	Due array di double di lunghezza generata tra 1 e 10000
Output	TRUE
Stato	
Nome del tester	Guido Riccardi
Note	

ID	T5P
Descrizione	Il metodo genera 1000 casi in cui viene definito un array e lanciata la funzione, verifichi se sommando un array di zeri della stessa lunghezza, il risultato rimanga invariato
Input	Un array di double di lunghezza generata tra 1 e 10000
Output	TRUE
Stato	
Nome del tester	Marianna Vantaggiato
Note	

Property based test ‘geometricMeanOfSumOfSquaresVariableLength’

ID	T1LP
Descrizione	Il metodo genera 1000 casi in cui vengono definiti due array di lunghezza differente e lanciata la funzione, non venga calcolata una media geometrica negativa
Input	Due array di double di lunghezza differente generata tra 1 e 10000
Output	true, perchè viene verificato che la Media geometrica non è negativa
Stato	
Nome del tester	Marianna Vantaggiato
Note	

ID	T2LP
Descrizione	Il metodo genera 1000 casi in cui vengono definiti due array di lunghezza differente e lanciata la funzione, verifichi se la funzione è simmetrica
Input	Due array di double di lunghezza differente generata tra 1 e 10000
Output	true, perchè viene verificato che la funzione è simmetrica
Stato	
Nome del tester	Marianna Vantaggiato
Note	

ID	T3LP
Descrizione	Il metodo genera 1000 casi in cui vengono definiti due array di lunghezza differente e lanciata la funzione, verificihi che aggiungendo un array di zeri di lunghezza pari o inferiore a quella di array1 e array 2 non modifichi la media
Input	Due array di double di lunghezza differente generata tra 1 e 10000
Output	TRUE
Stato	
Nome del tester	Marianna Vantaggiato
Note	