# Benchmarking Algorithms in Java

Dr Patrick Mannion

patrick.mannion@gmit.ie

## Overview

- Motivation for benchmarking
- Time in Java
- Benchmarking a single run
- Benchmarking multiple statistical runs

# Motivation for benchmarking

- Benchmarking or *a posteriori* analysis is an **empirical** method to compare the relative performance of algorithm implementations
- Experimental (e.g. running time) data may be used to validate theoretical or *a priori* analysis of algorithms
- Various hardware and software factors such system architecture, CPU design, choice of Operating System, background processes, energy saving and performance enhancing technologies etc. can affect running time
- Therefore it is prudent to conduct multiple statistical runs using the same experimental setup, to ensure that your set of benchmarks are representative of the performance expected by an "average" user

## Time in Java

- Dates and times in Java are represented as the number of milliseconds that have elapsed since midnight on January 1 1970 (the "Unix Epoch")
- 1 second = 1,000 milliseconds = 1,000,000,000 nanoseconds
- Each millisecond since the Unix Epoch has a specific timestamp
- Stored using the **long** type in Java (i.e. a timestamp is a large integer)
  - e.g. long currentTime = System.currentTimeMillis(); // current time in millisec
- System.nanoTime() gives a higher resolution
  - e.g. long currentTime = System.nanoTime(); // current time in nanoseconds

## Benchmarking a single run

```java
// log the start time (in nanoseconds)
long startTime = System.nanoTime();

// call the method that you want to benchmark
methodToTest(input);

// log the end time (in nanoseconds)
long endTime = System.nanoTime();

// calculate the time elapsed (in nanoseconds)
long elapsed = endTime-startTime;

// convert from nanoseconds to milliseconds
double timeMillis = elapsed/1000000.0;
```

# Benchmarking multiple statistical runs

```java
// number of times to test the method
int numRuns = 10;

// array to store time elapsed for each run
double[] results = new double[numRuns];

// benchmark the method as many times as specified
for(int run=0; run<numRuns; run++) {
  long startTime = System.nanoTime();
  methodToTest(input);
  long endTime = System.nanoTime();
  long elapsed = endTime-startTime;
  double timeMillis = elapsed/1000000.0;

  // store the time elapsed for this run
  results[run] = timeMillis;
}
```

## Useful references

- Java 8 System class documentation: https://docs.oracle.com/javase/8/docs/api/java/lang/System.html
- Java Date and Time overview: https://www.tutorialspoint.com/java/java_date_time.htm

- Discussion of benchmarking issues in Java (advanced material): https://www.ibm.com/developerworks/library/j-benchmark1/