Benchmarking Algorithms in Python

Dr Patrick Mannion

patrick.mannion@gmit.ie

GMIT
INSTITIÚID TEICNEOLAÍOCHTA NA GAILLIMHE-MAIGH EO
GALWAY-MAYO INSTITUTE OF TECHNOLOGY

Overview

- Motivation for benchmarking
- Time in Python
- 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 Python

- Dates and times in Python are represented as the number of seconds that have elapsed since midnight on January 1st 1970 (the "Unix Epoch")
- Each second since the Unix Epoch has a specific timestamp
- Can import the time module in Python to work with dates and times
  - e.g. start_time = time.time() # gets the current time in seconds
  - e.g. 1555001605 is Thursday, 11 April 2019 16:53:25 in GMT

Benchmarking a single run

```
1   import time
2
3   # log the start time in seconds
4   start_time = time.time()
5
6   # call the function that you want to benchmark
7
8   # log the end time in seconds
9   end_time = time.time()
10
11  # calculate the elapsed time
12  time_elapsed = end_time - start_time
```

Benchmarking multiple statistical runs

```
1   import time
2
3   num_runs = 10  # number of times to test the function
4   results = []  # array to store results for each test
5
6   # benchmark the function
7   for r in range(num_runs):
8       # log the start time in seconds
9       start_time = time.time()
10
11      # call the function that you want to benchmark
12
13      # log the end time in seconds
14      end_time = time.time()
15
16      # calculate the elapsed time
17      time_elapsed = end_time - start_time
18
19      results.append(time_elapsed)
```

GMIT
INSTITIÚID TEICNEOLAÍOCHTA NA GAILLIMHE-M
GALWAY-MAYO INSTITUTE OF TECHNOLOGY

Useful references

- Python 3 time documentation: https://docs.python.org/3/library/time.html
- Python Date and Time overview: https://www.tutorialspoint.com/python/python_date_time.htm

- Discussions of benchmarking issues in Python (advanced material): https://www.peterbe.com/plog/how-to-do-performance-micro-benchmarks-in-python https://www.blog.pythonlibrary.org/2016/05/24/python-101-an-intro-to-benchmarking-your-code/