Second Programming Assignment

Small Group Project

The purpose of this assignment is the creation of a scientific report on the measurement of run-times. You have to use Python or — if you must — C or C++ in order to do the assignment. Java is known for its lengthy garbage collection process that makes timing very unreliable.

Compare the run-times of the following strategies for calculating the *n*th Fibonacci number.

1. Using recursion without caching or memoization
2. Using recursion with an LRU cache of 50.
3. Using recursion with full memoization. (You should not keep the dictionary of memoized results between invocations).
4. Creating a list of all Fibonacci numbers until *n* and then return the last value.
5. Maintain the last two Fibonacci numbers in memory and then update them.

Raw numbers are not very useful, we also need to know error margins. Therefore, instead of reporting just observations, you need to make many and then determine error bounds.

In order to obtain could error bounds, you use the following method. First, you calculate the average of 50 runs. You repeat this 20 times. For example, on my machine, I obtained the following times (in seconds) for the recursive version of Fibonacci on input 22:

```
22
0.0089312707
0.0087678637
0.0085762837
0.0085739627
0.0085331099
0.0085901193
0.0087923036
0.0085898339
0.0085531757
0.0085288228
0.0085281050
0.0086248029
0.0085248201
0.0085610053
0.0086873000
0.0086450434
0.0090014142
0.0086983541
0.0087246703
0.0087848511
```

All values are averages. We now determine the average of the averages and the size of the confidence interval with an $\alpha = 0.05$. That means that we have a 95% assurance that the true time is in the confidence interval. If you use Excel, then you can use the built in functions for AVERAGE and CONFIDENCE. The latter uses the standard deviation.

The reason to use averages as the individual numbers is that the averages are approximately normally distributed. There is another confidence interval for the Student's t-distribution. If you look closely at the averages, then you will see that the numbers seem to be correlated. This is because I was running other processes on my computer while I was getting these times. Don't do this, try to keep your computer only occupied with the numbers that you are obtaining.
The average is 0.008660856 and the confidence interval is 6.00856E-05. You now need to plot the time series with error bars. You have several options. You can use MS Excel (or something similar) but the user-interface is a true pain. I use Mathematica, but Matlab (which is free for you) also has a nice chart printing capability that is uncharacteristically easy to use.

The range of values to display is up to you. However, since you run each function for each argument 1000 times, you can certainly stop when the individual run-time becomes close to 1 second. Since 1000 seconds is 16.667 minutes, gathering data is not very fast. Luckily, even undergraduates need to sleep (or they use a lab machine). For the more efficient methods, you do not need to measure every integer value. However, in order to make charts comparable, you should select the same range of values for all three fast methods.

Handins:

Labelled charts with error bars for the five implementations.

Source code