Lab #15

Physics 91SI Spring 2013

Objective: In this lab, you will use the profiler cProfile to optimize your implementation of *The Game of Life* from the last lab. You will also profile your C code from lab 13 to create faster algorithms for matrix operations.

As usual, log on to `corn` and clone over the starter repository:

```
hg clone /afs/ir.stanford.edu/class/physics91si/src/lab15 lab15
```

This repository contains a file `init.py`, which will pull the files from your submission of lab14. If you have not already pushed lab14 to the submission directory, do this first: from within lab14, run

```
hg push /afs/ir.stanford.edu/class/physics91si/submissions/user/lab14
```

Run `./init.py` from within lab15 to copy over the contents of lab14.

As usual, remember to `hg commit` often to save your changes, and submit your code the end of class.

Part 0: The Game of Life

If you have not already implemented the Game of Life in Cython or pure C, continue working on this assignment until you have a working version of `life_cy.pyx`.

Also, add `import sys` to the top of `life.py` and add these two lines to the end:

```
if __name__==“__main__”:
      life(40, 0.2, plotting=False, version=sys.argv[1])
```

This allows us to run the Game of Life from the command line, where the string following the filename “`life.py`” is the version we want to run. <u>We cannot run the profiler on `life.py` without these lines.</u>

Part 1: Optimizing Life

As before, we can run `%timeit` in iPython to get a sense of overall runtime and scaling, but now we want to view statistics on specific function calls. From the command line, run the python profiler to view these statistics on your code:

```
python -m cProfile -o life_python life.py python
```

Load these statistics from within iPython:

```
import pstats
pp = pstats.Stats("life_python")
```

and try sorting the data by different parameters (http://docs.python.org/2/library/profile.html has a list of examples).

If you find any 'hot loops' in your code that take up most of the total computation time, see if you can speed up these loops in particular. What factor of speedup do you see? Make sure to commit after each major change you make to the algorithm.

Once you've improved your algorithm for the python version of life, run the profiler with the Cython or C versions and make appropriate optimizations to your code. How does your speedup compare to that of the python version?

Part 2: Linear Algebra

In lab 13 you wrote algorithms in `matrices.c` to add and multiply matrices. Using the C profilers mentioned in lecture, let's see how well we can speed up this code.

Run the script `init2.py` to copy over the files from lab 13. Once you have `matrices.c`, run GProf to view statistics on your code. Remember to use the `-pg` flag when compiling `matrices.c`.