COP5612 – Fall 2013

Alin Dobra

August 29, 2018

- **Due Date:** September 10, Midnight
- One submission per group
- Submit using eLearning
- **What to include:**
  - `README` file including group members, other requirements specified below
  - `Lastnames.zip` the code for the project

1 Problem definition

An interesting problem in arithmetic with deep implications to *elliptic curve theory* is the problem of finding *perfect squares that are sums of consecutive squares*. A classic example is the Pythagorean identity:

$$3^2 + 4^2 = 5^2 \tag{1}$$

that reveals that the sum of squares of $3, 4$ is itself a square. A more interesting example is Lucas' *Square Pyramid*:

$$1^2 + 2^2 + ... + 24^2 = 70^2 \tag{2}$$

In both of these examples, sums of squares of consecutive integers form the square of another integer.

The goal of this first project is to use Elixir and the actor model to build a good solution to this problem that runs well on multi-core machines.

2 Requirements

Input: The input provided (as command line to your program, e.g. `my_app`) will be two numbers: N and k. The overall goal of your program is to find all k consecutive numbers starting at 1 and up to N, such that the sum of squares is itself a perfect square (square of an integer).

Output: Print, on independent lines, the first number in the sequence for each solution.

Example 1:

```
mix run proj1.exs 3 2
3
```

indicates that sequences of length 2 with start point between 1 and 3 contain 3,4 as a solution since $3^2 + 4^2 = 5^2$.

Example 1:

```
mix run proj1.exs 40 24
1
```

indicates that sequences of length 24 with start point between 1 and 40 contain 1,2,...,24 as a solution since $1^2 + 2^2 + ... + 24^2 = 70^2$.

Actor modeling: In this project you have to use exclusively the actor facility in Elixir (**projects that do not use multiple actors or use any other form of parallelism will receive no credit**). A model similar to the one indicated in class for the problem of adding up a lot of numbers can be used here, in particular define *worker* actors that are given a range of problems to solve and a *boss* that keeps track of all the problems and perform the job assignment.

README file In the README file you have to include the following material:

- Size of the *work unit* that you determined results in best performance for your implementation and an explanation on how you determined it. Size of the work unit refers to the number of sub-problems that a worker gets in a single request from the boss.
- The result of running your program for
  `mix run proj1.exs 1000000 4`
- The running time for the above as reported by time for the above, i.e. run `time scala project1.scala 1000000 4` and report the time. The ratio of CPU time to REAL TIME tells you how many cores were effectively used in the computation. If your are close to 1 you have almost no parallelism (points will be subtracted).
- The largest problem you managed to solve.

3 BONUS – 15%

Use remote actors and run you program on 2+ machines. Use your solution to solve a really large instance such as: `mix run proj1.exs 100000000 20`. To get the bonus points you have to give a demo to the instructor and explain your solution.