INFO-F-404: Real-Time Operating Systems

2013 – 2014 Project 2: The Ulam spiral

1 Main goal

Implement a multiprocessor program that can generate an *Ulam spiral*.

2 Prime numbers and Ulam spiral

A natural number p is prime only if it is disible by itself and by 1 (with an exception: 1 is not a prime number). Numbers that are not prime are called composite numbers. There exist infinite number of primes, many problems related to prime numbers and huge number of applications that use properties of prime numbers.

The Ulam spiral is a techniques that is used to visualize prime numbers. Main idea is to put all natural numbers on a grid into a spiral and then to highlight all primes, see Figure 1. We may represent numbers by pixels and create an image where pixels for prime numbers and composite numbers have different color. In that case Ulam spiral might look like Figure 2.

17	16	15	14	13
18	5	4	3	12
19	6	1	2	11
20	7	8	9	10
21	22	23	24	25

Figure 1: Begining of the Ulam spiral

2.1 Generating prime numbers

There are a lot of different algorithms that might be used to generate primes. A prime number sieve is a type of fast algorithm that might be used to generate prime numbers.

We suggest you to look at the following algorithms :

- Trial division applied to odd numbers (naive algorithm) and its variants and optimizations,
- Sieve of Eratosthenes and its optimized versions (e.g. sieve of Atkin),
- Wheel factorization

There also exist many algorithms that can verify that a given number is a prime or a composite, such algorithms are called primality tests and compositeness tests, for example:

- Miller–Rabin test

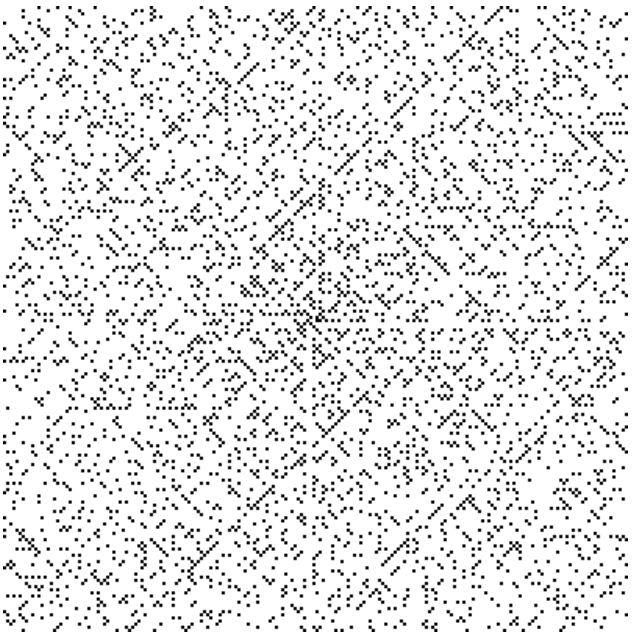

Figure 2: Ulam spiral in form of a picture. Image from Wikipedia.

- Fermat primality test
- Adleman–Pomerance–Rumely primality test

You are not restricted to those algorithms and may choose any other algorithm for this project. You can find more information about algorithms for prime number generation in books *"Prime numbers and computer methods for factorization"* by Hans Riesel and *"Primality testing and integer factorization in public-key cryptography"* by Song Y. Yan (both available at ULB library).

Those algorithms might be implemented in a multiprocessor environment using one of the two naive schemes:

1. each process generates (or verifies) numbers in its own subset of numbers e.g. we might choose to distribute all numbers among 3 processors using the following rule: process P_i ($i \in \{1, 2, 3\}$) works with numbers $j \times 3 + i \mid j = 0, 1, 2, \ldots n$.
2. for a given number p, each process verifies if it is prime in its own range e.g. Suppose we use 3 processors and we want to test if 33 is prime using a naive trial division (i.e. we already tested numbers between 1 and 32), process 1 might test numbers in the interval [1;11], process 2 in [12;23] and process 3 in [24;32]. The first process to find that 33 is not prime will notify others.

You may choose to implement any miltiprocessor scheme of your choice. For more efficient techniques plese read papers *"Two fast parallel prime number sieves"* by Jonathan Sorenson and *"A fully distributed prime numbers generation using the wheel sieve"* by Gabriel Paillard.

3 Project details

We ask you to implement a program called `ulamSpiralGenerator` using C++ language and MPI library available on HYDRA.

A user should be able to choose the size of the spiral. Each number should be represented as a small square (e.g. 1 pixel, 4 pixels).

Program `ulamSpiralGenerator` should use several cores to generate the picture. At the end of the execution the program `ulamSpiralGenerator` should also print the time spent to generate the picture : time that each process spent working and the total time (wall clock time). It should also print the number of primes as well as the total number of numbers on the picture.

A user should be able to run your program using the following line :

`mpirun -np <N> ./ulamSpiralGenerator <size> <filename_prefix>`

`N` is the number of parallel processes used, `size` is the minimum number of primes that the user want to generate (but your program has to create a complete square that might containt more primes). Finally, `filename_prefix` is the prefix of the filename (to which you should append the number of primes that is on the picture).

For example, in case if user enters `mpirun -np 2 ./ulamSpiralGenerator 6 mySmallSpiral` the program should create an image 5×5 squares with 9 primes and call the file `mySmallSpiral_9primes` (see Figure 1). The program's output should look like that :

```
Time spent:
Process 1 : 0,00013 sec
Process 2 : 0,0002 sec
Total time : 0,00021 sec

Statistics:
Generated primes : 9
Composite numbers : 16
Total : 25 numbers
```

The size of each square (that represent one number), color scheme, format of the image file (e.g. png, ppm) as well as the algorithm that you use to generate prime numbers are left to the choice of developers.

You should also write a short report that contains:

1. description of algorithms that you used to generate prime numbers,
2. short description of your code (diagrams) and implementation choices,
3. a description of the protocol that you used to implement the program (messages transiting between processes),
4. a section where you describe difficulties that you met during this project (and solutions that you found),
5. a section that describe result of your project.

4 Bonus part

This part is not mandatory, but it is worth some bonus points (only if all other parts work).

This time we propose an artistic bonus and a performance bonus. You might create a very fast program that can quickly generate big Ulam spirals (and may the best team win). You might also choose to add some artistic components (depending on the user's choice) to the resulting picture; he might chose a color scheme or use use varations of colors in the same picture (e.g. gradient) as well as represent only a part of the Ulam spiral (e.g. in a circle).

5 Submission and planning

This project should be done in groups of 2, you may choose your partner. This project has to be submitted before 23:59:59 o'clock on December 2 of 2013. Your project has to work properly (compile and execute) on HYDRA (hydra.ulb.ac.be).
To submit (in a *zip* file) a folder that contains at least following files:

- all your C++ sources,
- a Makefile (that creates all executable files),
- A short report (pdf format, about 4-5 pages plus an appendix, figures and graphics are welcom).

The zip file with your project has to be send by e-mail to **nikita.veshchikov@ulb.ac.be** with subject "INFO-F-404 project2". The name of the folder and of the containing zip file : if Chuck Norris makes his project with Michael Jackson they should send a file named *norris-jackson-project1.zip*.

Good luck!