SE205: TP
Lock-Based List-Based Set Implementations

The goal of this exercise session is to get familiar with fine-grained locking schemes on the example of a sorted linked list used to implement a set abstraction.

We are going to use the `synchrobench` benchmark (`https://github.com/gramoli/synchrobench`) to study the performance of four lock-based set implementations:

- Coarse-grained locking (already present as `linkedlists.lockbased.CoarseGrainedListBasedSet.java`)
- Optimistic list with wait-free traversal and validation
- Lazy Linked List by Heller et al. (already present as `linkedlists.lockbased.LazyLinkedListSortedSet.java`)

For the optimistic algorithm refer to the slides or Chapter 9 of the book by Herlihy and Shavit (`https://www.dawsonera.com/abstract/9780123977953`).

In the simplest setting, `synchrobench` allows you to define a workload varying the following parameters: number of threads, size of the list, range of the list values, duration of the experiment, and fraction of updates (adds and removes, equally shared). In the simulated runs, every update picks a random value in the range uniformly at random.

The collected statistics contains the throughput (the number of complete operations per second), as well as the fractions of successful removes, adds and contains.

Check INSTALL for instructions on adding new algorithms to the `synchrobench` library (you may also use `eclipse` with `build.xml` to do this), and README.md for the parameters of the simulations.

A sample command to run `synchrobench` with a lock-based linked list algorithm ALG on 10 threads, update ratio 10, list size 1024, range 0-2047, and duration 3000ms is:

```
> java -cp bin contention.benchmark.Test -b linkedlists.lockbased.ALG -d 3000 -t 10 -u
    10 -i 1024 -r 2048
```

(1) Implement the optimistic list-based sets. You may take the coarse-grained locking algorithm as a basis. Make sure that your new classes implement `AbstractCompositionalIntSet` and export the right interface.

(2) Run the three algorithms: coarse-grained, optimistic and lazy, through `synchrobench` with the following parameters:

- Number of threads: 1, 4, 8, 12
- Update ratios: 0, 10, 100
- List sizes: 100, 1k, 10k (choose the ranges of the lists twice as big, to maintain the expected list size constant)
- For the other parameters, specify `-W 0 -d 2000` (no "warmup", duration 2000ms).

(3) Prepare a graph depicting the throughput as a function of the number of threads for the three algorithms, for some representative list size and update ratio. You may use `gnuplot` or any other plotting program of your choice.

Also, for each algorithm, prepare a graph depicting the throughput as the function of the number of threads, varying the update ratio, for the list size 1k.

Explain the forms of the curves and their relations to each other and prepare a short report with the plots and explanations.

Please include in your report, the system details of the machine you used for your experiments: in particular, the number of cores is important. (Ideally, use a machine with 12 or more cores).

The report is to be sent to `mailto:petr.kuznetsov@telecom-paristech.fr`