INF5171: Laboratoire #2
Parallélisme récursif de style *fork/join* et utilisation de `PRuby.pcall` et `PRuby.future`

21 septembre 2017
13h30–15h30
PK-S1565

Le but de ce laboratoire est de vous familiariser avec l'utilisation de la bibliothèque `PRuby` et ses constructions pour le parallélisme de type *fork/join*.

---

1 Obtenir le code à compléter

Obtenez une copie des fichiers pour le labo en exécutant la commande suivante sur `japet` :

```
$ git clone http://www.labunix.uqam.ca/~tremblay/git/Inversions.git
```

2 Ce que vous devez faire

Pour ce laboratoire, vous devez **compléter** trois versions d'une méthode, définie dans une **extension** de la classe `Array`, qui permet de trouver le nombre d'**inversions** dans un tableau.

Plus spécifiquement, il s'agit de compter combien **d'éléments adjacents** du tableau **sont dans le mauvais ordre**. Voici quelques exemples :

```
[10, 20].nb_inversions.must_equal 0
[20, 10].nb_inversions.must_equal 1
[10, 10, 30, 30].nb_inversions.must_equal 0
[10, 10, 3, 3].nb_inversions.must_equal 1
[2, 1, 3, 4, 6, 5, 7, 9, 8, 10].nb_inversions.must_equal 3
```

(Pour des exemples additionnels, voir le programme de test : `nb_inversions_spec.rb`.)

Vous devez ensuite comparer les performances de ces trois versions, et ce à l'aide du programme `nb_inversions_bm.rb` : voir le `makefile`.

3 Ce qui vous est fourni

Une méthode séquentielle itérative, `nb_inversions_seq`, vous est fournie. Vous devez définir, puis comparer, trois autres méthodes, une séquentielle, deux parallèles :

1. `nb_inversions_rec` : Méthode **récursive séquentielle** fondée sur une approche diviser-pour-régner dichotomique avec seuil — troncation de la récursion lorsque la taille du problème devient inférieure à un certain seuil.

2. `nb_inversions_rec_pcall` : Méthode **récursive parallèle** — dichotomique et avec seuil — qui utilise la construction `PRuby.pcall`.

3. `nb_inversions_rec_future` : Méthode **récursive parallèle** — dichotomique et avec seuil — qui utilise la construction `PRuby.future`.

Divers autres fichiers vous sont fournis :

- `nb_inversions.rb` : Fichier avec squelette de mise en oeuvre, qui inclut un «répartiteur» (*dispatcher*) pour appeller la version requise — voir plus bas — ainsi qu'une méthode séquentielle itérative `nb_inversions_seq`.

- `nb_inversions_spec.rb` : Programme de tests. Ce programme teste diverses versions avec divers seuils. Pour ne tester qu'une seule méthode, il suffit de spécifier la variable d'environnement `METHODE` : voir le fichier `makefile`.

  Remarque : Si vous désirez rendre temporairement *inactifs* un ou des cas de tests, deux possibilités :

  1. Remplacer l'appel de méthode «`it`» par «`_it`».
  2. Ajouter une instruction «`skip "Raison ..."`» à la $1^{ère}$ ligne du cas de test.

- `nb_inversions_bm.rb` : Programme de mesures des performances — temps et accélération. Lui aussi s'exécute pour un ensemble de combinaisons de versions et de seuils.

- `makefile` : Fichier pour automatiser le lancement des tests et autres tâches :
  - `$ make` : Exécute le programme de tests.
    Initialement : `DEFAUT=$(TESTS)` et `TESTS=tests_rec`.
  - `$ make tests_rec`, `$ make tests_pcall`, `$ make tests_future` : Lance les tests pour une des trois versions.
  - `$ make tests` : Lance le programme de tests pour les trois versions.
  - `$ make bm` : Lance le programme de mesures des performances.

4 Question additionnelle

Si vous avez complété la version séquentielle et les deux versions parallèles et fait l'analyse des performances, écrivez alors une version **parallèle récursive «trichotomique»** — donc qui décompose récursivement le problème en trois (3) sous-problèmes.