Karnaj

CC BY NC ND

Zeste de savoir

Séquences infinies en Ruby

4 mai 2019

Table des matières

1. Introduction 1

1. Introduction

Juste un petit billet pour parler rapidement des séquences « infinies » en Ruby qui se font notamment grâce à la classe `Enumerator`. Quand on parle de séquences infinies, en fait, il s'agit d'être paresseux et de ne calculer les choses que quand y en a besoin.

Un exemple est la classe `Prime` pour les nombres premiers. `p.next` nous donne le prochain nombre premier.

```
p = Prime.each
10.times { puts p.next }
puts p.take(10)
```

Avec la méthode `lazy`, on transforme un objet énumérable en `Enumerator` paresseux. Les éléments ne seront donc calculés que si nécessaires.

```
l = (1..Float::INFINITY).lazy.map { |x| 2 * x }
10.times { puts l.next }
```

Ici, le calcul du `map` ne sera fait que pour les 10 premiers éléments de la liste (car on en a besoin, on a demandé les valeurs).

Mais dans ce billet, je voulais surtout poster un petit test rigolo. Avec des lambdas, je définis la fonction exponentielle par sa somme

$$\exp(x) = \sum_{k=0}^{+\infty} \frac{x^k}{k!}.$$

```
def subexp
  (0..Float::INFINITY).lazy.map do |k|
    lambda { |x| x**k / Math.gamma(k + 1) }
  end
```

```
5 end
6
7 exp = lambda { |x| (subexp.map { |f| f.call(x) }).sum }
8
9 # puts exp(2)
```

Et voilà, un code qui mélange lambdas et `Enumerator`. Bien sûr, ça ne va pas trop fonctionner si on utilise la méthode `exp`... On pourrait être malin et lui donner un second argument (un k maximum) et dans ce cas, ça terminera, et ça donnera en fait un moyen de contrôler la précision de la méthode (mais on aura alors un code compliqué pour pas grand-chose).

PS : La flemme (celle de calculer la factorielle) explique la présence de `Math.gamma`...