# Introduction

# 1 Mission

## 1.1 Problème posé

L'exploration de l'arbre d'un semigroupe numérique n'est pas triviale et est apparentée au problème de Frobenius.

Le mathématicien Georg Frobenius s'est posé la question suivante : quel est le montant maximal que l'on peut pas rendre en fonction de pièces de monnaie données ?

Ce montant se nomme "nombre de Frobenius" et est plus formellement décrit par le plus grand entier qu'il est impossible de calculer à partir de coefficients donnés.

On peut aussi visualiser ce problème en considérant les scores au Rugby : quels sont les scores que l'on ne peut pas obtenir avec le but (3 points), l'essai (5 points) et l'essai transformé (7 points) ?

La résolution des scores au Rugby est simple, on ne peut pas avoir 1, 2 ou 4 points ; tous les autres scores sont possibles. En effet, il est aisé de calculer le nombre de Frobenius (ici, 4) pour $n \leqslant 3$ où $n$ est le nombre d'entiers possibles pour la combinaison. Ici, $n = 3$ soit $\{3, 5, 7\}$.

# 2 Déroulement

## 2.1 Jeudi 12 janvier 2017

Installation au LRI, explications :

- parcours de l'arbre de semi-groupes numériques
- structures de données utilisées, notamment dans `NumericMonoid`
- algorithme "naïf" : tableau de booléens (le nombre est-il dans le semi-groupe ?) de taille $B =$ genre, soit la profondeur dans l'arbre et le nombre de trous

  **pour** $g$ de dernierEnlevé + 1 à $B$ **faire**
  **si** $T[g]$ **alors**
  **pour** $i$ de 1 à $\lfloor \frac{g}{2} \rfloor$ **faire**
  **si** $T[i]$ et $T[g-i]$ **alors**
  $g$ est un nombre de décomposition
- algorithme optimisé facilitant la vectorisation : tableau d'entiers modélisant le nombre de paires ($i, j \in$ SN tel que $i + j = g$ et $i \leqslant j$), tous les nombres avec exactement une paire sont des générateurs

## 2.2 Autonomie

- première implémentation de l'algorithme naïf ; difficultés : initialisation, confusion décomposition/générateur
- première implémentation de l'algorithme optimisé
- organisation du dépôt Git

### 2.3 Lundi 16 janvier 2017

— mise au point et correction de l'algorithme naïf

— débogage et tests

### 2.4 Autonomie

— mise en place de tests unitaires

— amélioration de l'algorithme naïf ; difficulté : les tests unitaires sur des semigroupes connus passent, mais le résultat final (nombre de semigroupes calculés, par genre) n'est pas bon

— correction de l'algorithme optimisé : il fonctionne

## 3 Résultats

## Conclusion