Algorithme genetique

# Avant propos

Le projet consiste a trouver en utiilisant les technique d’algorithme génétique les différent point d’une ville en fonction de critères de recherche

(proche d’écoles primaires, de transports, de restaurant… etc)

En fonction de ces critères et de la proximité de ces derniers un score est attribué (**fitness**)

## Individu

Les individu de notre algorithme génétique disposent de coordonnées,   
Cette coordonnée est constituée de 2 axes (x et y) et existe uniquement dans une ville (par facilité) pour intégrer du contrôle concernant le dépassement de la ville ainsi programmée).

## Génération

La génération est un ensemble d’individu avec différents critères tel que un score ajouté lorsque l’on trouve des restaurants … etc

La génération peut se générer une première fois basée sur un ensemble de critères en générant des individus aléatoire,

Ou en se basant sur une génération précédente auquel cas la nouvelle génération va faire des **crossover** et des **mutation** entre les individus de la génération précédente (avec pour but finale d’améliorer le **fitness** de sa population).

## Ville

La ville est un carré de 4 km (une valeur de 4000 est attribué)

Cette ville contient des lieux (par facilité généré en brut dans le code avec une procédure Load)

## Coordonnée

Comme indiquée les coordonnées sont relatives à une ville ainsi elle regarde la taille de la ville pour ne pas dépasser,

Ce sont ses coordonnée qui vont avoir les méthodes pour générer un crossover et une mutation

### Crossover

Le crossover est un croisement « génétique » de 2 coordonnées pour en former une.

Comme nos coordonnée ne peuvent pas dépasser 4000 elles ne peuvent par conséquent pas dépasser plus de 12 bits (car 12 bits = [0 ; 4095])

On prends les valeurs binaires en X de nos deux parents P1 et P2

0 0 1 0 1 0 1 0 1 1 0 1 (685)

0 0 0 0 1 1 1 1 0 1 1 0 (246)

On place arbitrairement un bit aléatoire comme étant point de croisement ici le 5 ème bit de poids faible.  
  
On peut alors generer deux crossOver pour l’axe X

0 0 1 0 1 0 1 0 0 1 1 0 (678)

0 0 0 0 1 1 1 1 1 1 0 1 (253)

### Mutation

Maintenant supposons qu’on veuille générer des mutations pour cela on va indiquer un taux de probabilité **p**,

A chaque bit **b** de notre donnée ici l’axe X sur 12 bits,   
on va avec la probabilité de p inverser le niveau logique de b ( càd dire que ce bit b = !b)