Algorithmes génétiques

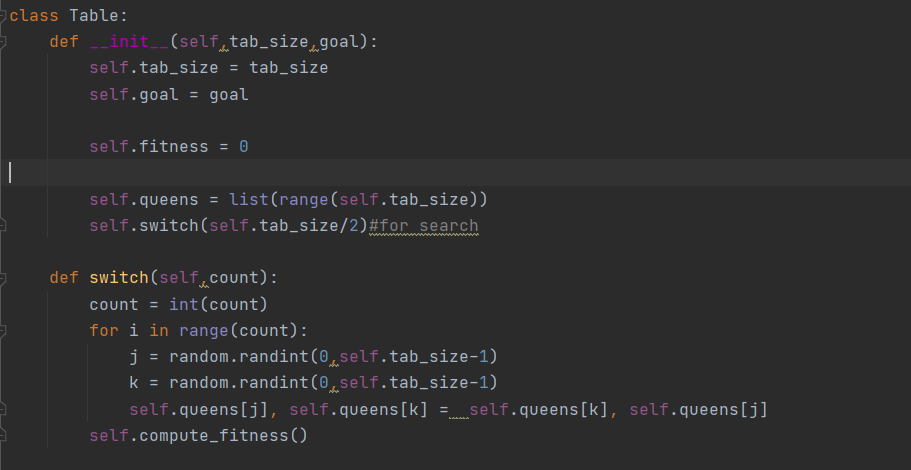
Dans ce partie on va utiliser genetic algorithm pour résoudre le probleme de N-REINS

Algorithmes génétiques sont des algorithmes de recherche heuristique adaptatifs qui appartiennent à la plus grande partie des algorithmes évolutifs. Les algorithmes génétiques sont basés sur les idées de sélection naturelle et de génétique. Il s'agit d'une exploitation intelligente de la recherche aléatoire fournie avec des données historiques pour diriger la recherche dans la région de meilleures performances dans l'espace de la solution. Ils sont couramment utilisés pour générer des solutions de haute qualité pour les problèmes d'optimisation et les problèmes de recherche.

Les algorithmes génétiques sont basés sur une analogie avec la structure génétique et le comportement du chromosome de la population. Voici le fondement des AG basées sur cette analogie

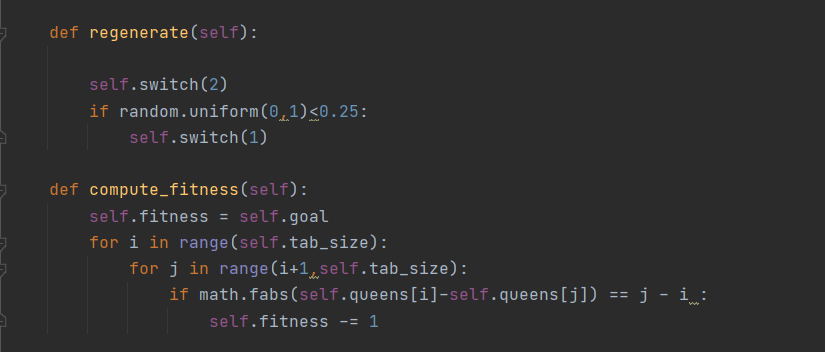
* Les individus de la population se disputent les ressources et s'accouplent
* Les individus qui réussissent (les plus en forme) s'accouplent alors pour créer plus de progéniture que les autres
* Les gènes du parent le plus «apte» se propagent tout au long de la génération, c'est-à-dire que parfois les parents créent une progéniture qui est meilleure que l'un ou l'autre des parents.
* Ainsi, chaque génération successive est plus adaptée à leur environnement.On va discuter le code utilser pour resoudre ce probleme :

-Le code utiliser est contient deux class :Table et GAqueens

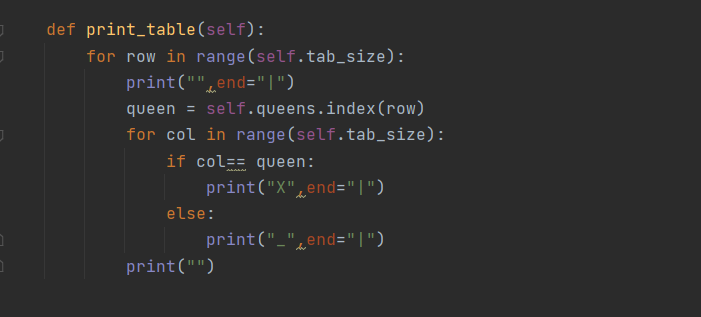


Dans la class Table on initialiser par un constructeur et defines les variable suivant:

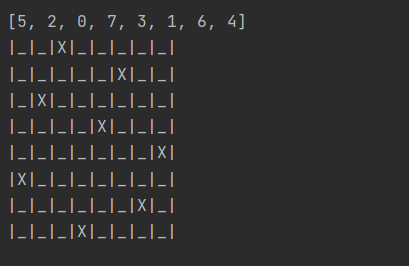
* Tab\_size: c’est la tail de table de chess aussi le nombre des chromosome(reins) dans ce probleme.
* Goal: c’est le target
* Fitness: initialiser la valeur de function fitness
* Queens: créer les premiers reins dans la table de chess
* Switch: est une function qui va changer les places des gene dans uns chromosomeet calculcler la valeur de fitness pour la nouveau combinaison des reins



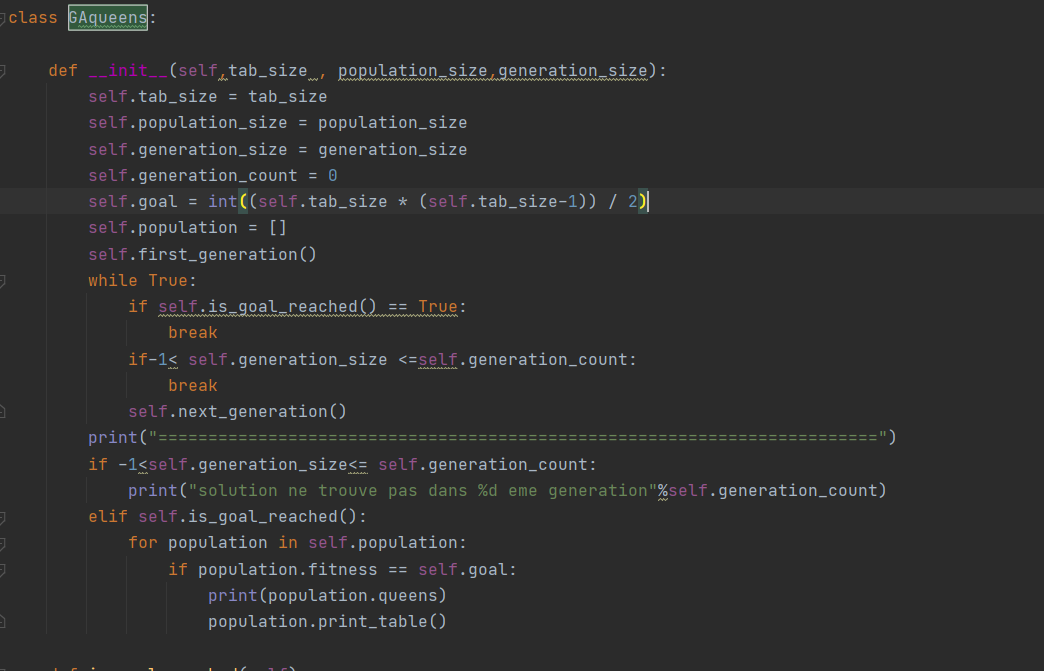
* Regenerate: cette function va déplacer au hasard 3 reines utilisé pour créer une nouvelle generation
* Compute\_fitness: dans cette function on v calculer la fitness de population donné par donner a fitness la valeur de target et lorsque on trouve une rein attacke une autre on va dicrémenter la valeur de fitness par 1



Cette function va design la table de chess au console et placer le reins au etat final



resultat



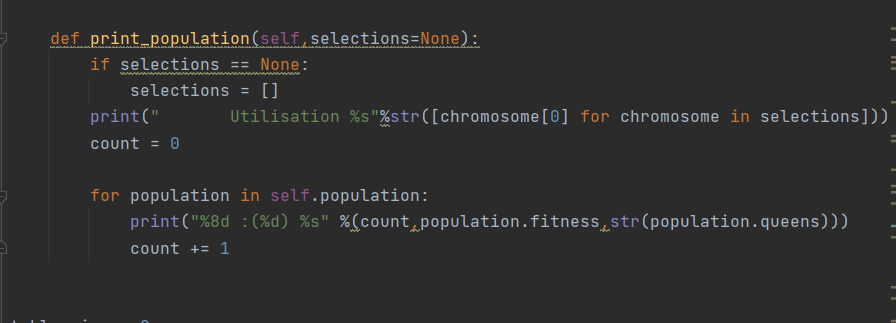
Dans le constructeur de deuxieme class on a :

* Donner une value a la fitness .
* Definit la tail de population
* Creer la premiere generation
* Tester si on a atteindre la solution par la function is-goal\_reached() qui va retourn un Boolean si nous ne atteignons pas notre objectif on va créer une autre generation par la fonction next\_generation()



* la function random\_selection() c’est l’une des caraterestique de gneetiques alghorithmes où on va

sélectionner certains éléments de la population actuelle pour la sélection de la prochaine génération sont les éléments avec la valeur d'ajustement la plus élevée renvoie une liste de selections



* print\_population() va afficher tout le de population donnée comme suivant

0 :(96) [5, 1, 2, 4, 7, 0, 6, 3]

1 :(99) [0, 4, 1, 5, 2, 6, 3, 7]

2 :(96) [2, 5, 3, 4, 0, 1, 6, 7]

3 :(87) [0, 5, 2, 1, 4, 3, 6, 7]

4 :(95) [5, 1, 4, 0, 3, 2, 6, 7]

5 :(91) [6, 1, 4, 7, 2, 3, 0, 5]

6 :(97) [0, 4, 5, 3, 1, 7, 2, 6]

7 :(93) [5, 4, 2, 1, 3, 0, 6, 7]

8 :(90) [0, 1, 7, 5, 4, 3, 6, 2]

9 :(92) [0, 2, 1, 6, 4, 5, 3, 7]

10 :(95) [4, 1, 2, 6, 5, 7, 3, 0]

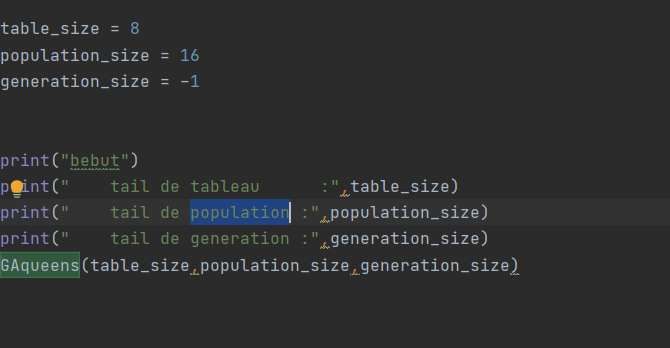
11 :(87) [0, 7, 2, 1, 4, 5, 6, 3]

12 :(93) [3, 7, 1, 0, 4, 5, 6, 2]

13 :(90) [0, 1, 2, 4, 3, 5, 7, 6]

14 :(92) [3, 4, 1, 0, 2, 7, 6, 5]

15 :(93) [0, 5, 1, 3, 2, 6, 4, 7]



on a affecter des valeurs constants pour executer notre code

On affect -1 à generationi\_size pour met l’lgorithme cherché au solution sans limitation

Finalement on initiliser la class GAqueens avec les constant données.

La resultat est suivant:

