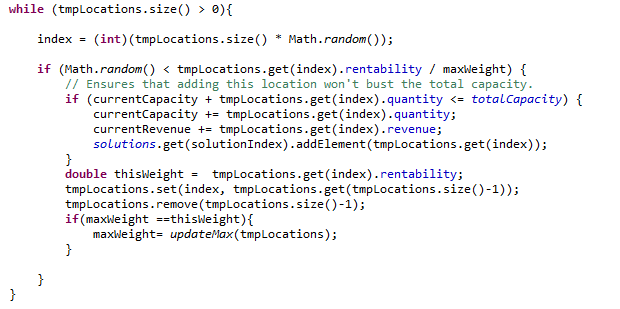
Analyse asymptotique de complexité :

Vorace :

Voici l’itération principale de l’algorithme



C’est une itération sur tmpLocation. //TODO SELECTION

À chaque itération, la taille de tmpLocation diminue de 1 menant à un nombre d’itération linéaire.

À l’intérieur de de la sélection, l’ajout d’élément dans une liste a une complexité constante, de même pour la comparaison et les additions. Enlever un élément à une liste a une complexité de N en moyenne, cependant nous avons contourné ce problème en échangeant l’élément à retirer et le dernier élément. Enlever le dernier élément d’une liste est seulement de complexité constante.

Toujours dans ce IF, nous devons mettre à jour le maxWeight si ce dernier a été modifié. Les probabilités que cela arrive est d’environ 1/n, et le coût n, car on doit itérer à travers la liste. En moyenne cela fait une complexité constante car 1/n\*n = 1.

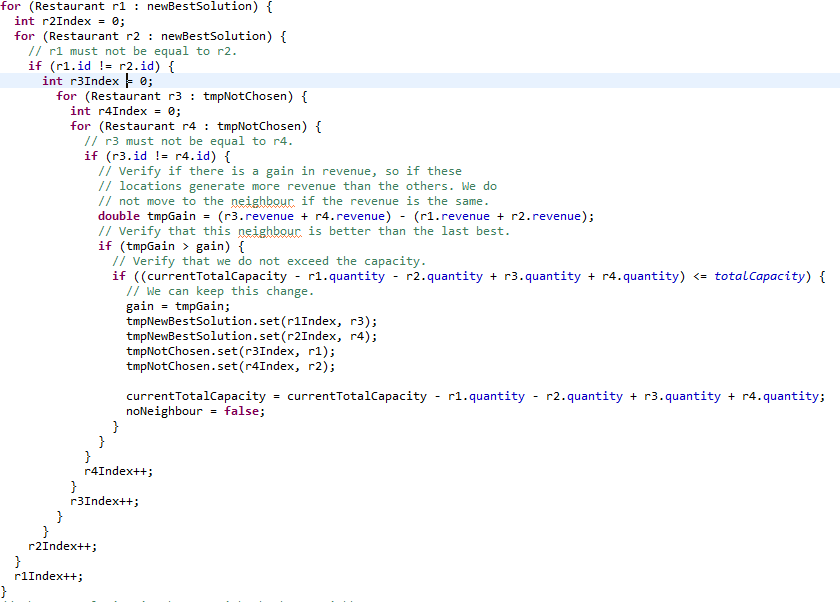
Comme toutes les opérations à chaque itération sont en temps constantes, on peut donc affirmer que la complexité de l’algorithme est de O(n).

Dynamique :

L’algorithme consiste à évaluer les éléments d’un tableau de deux dimensions de taille nombre d’éléments par capacité totale, la complexité est donc égale à O(nm) où n est le nombre d’éléments et m la capacité totale de la ville.

Amélioration locale.

Le morceau de code à analyser est les tests de comparaisons 2 par 2, parce qu’elle est d’évidence plus complexe que le 1 par 1.



La complexité serait en théorie où n est le nombre d’éléments, m le nombre d’éléments dans la solution et i le nombre d’itérations avant l’arrêt, car l’intérieur de la boucle se fait en temps constante.