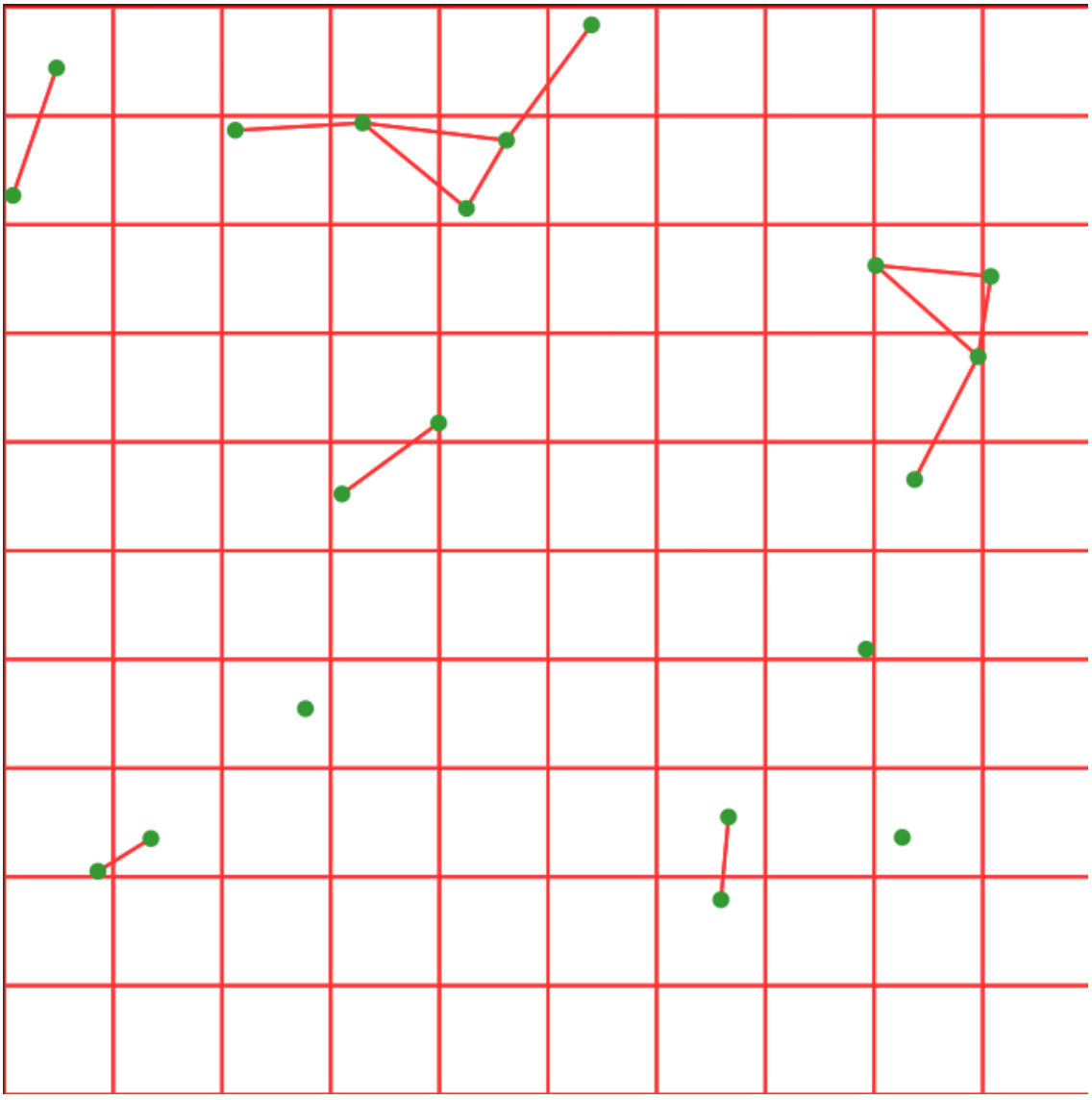
Algorithmique et structure de données

Rapport du projet

ENSIMAG 1A - Année 2020-2021

Mustapha ADIOUANE – Oussama AHOUZI



Explication de l'algorithme :

Classe Connexe :

On définit d'abord une classe appelée connexe qui représente un carré de coté (d/racine (2)) et donc les points à l'intérieur vont faire partie par conséquent à une même partie connexe. Les 4 attributs définis sont :

- Contenu : Liste des points à l'intérieur du carré.
- PASSED : égale à 1 si on a déjà traité le carré et 0 sinon.
- Indice : représente le numéro du carré (i, j) dans notre grille ([0,1] *[0,1]) sachant que le premier est celui dont un des sommets est l'origine.
- Size : C'est tout simplement la longueur de la liste contenue, elle sert à éviter d'ajouter les points d'un rectangle déjà traité.

Création des carrés et remplissage du dictionnaire :

On crée les carrés qui contiennent au moins un point de notre liste "points" et on les met dans un dictionnaire. On met à jour les attributs size et contenu à chaque fois qu'on retrouve un nouveau point.

Traitement des carrés :

On boucle sur les clés de notre dictionnaire et pour chaque carré qui n'est pas déjà parcouru :

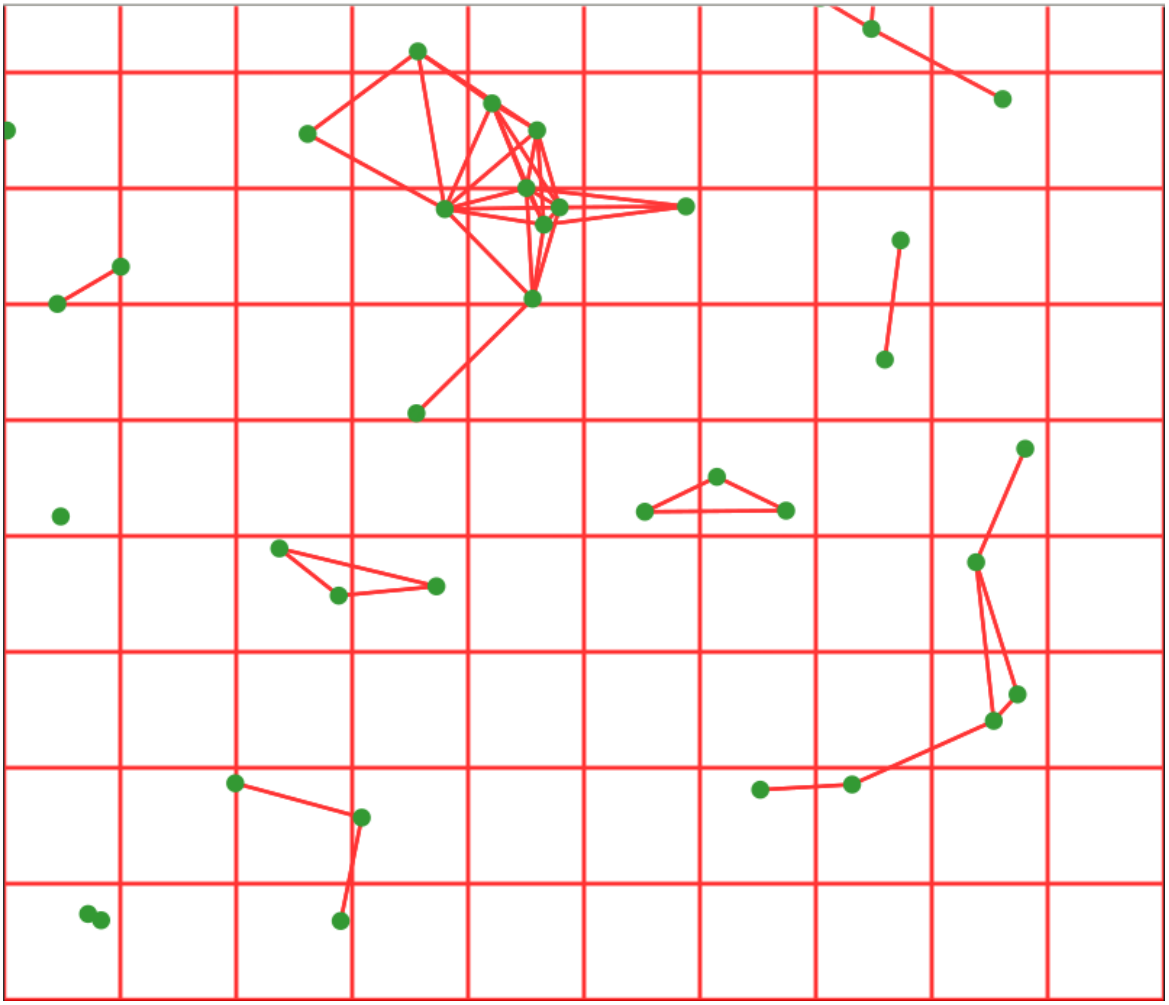
1-On crée une liste **L** qui va contenir les carrés à comparer (au début elle contient un seul) et une variable longueur qui va contenir le nombre de points de notre composant connexe.

2-On le compare avec ses adjacents (les carrés qui se retrouvent à 2 cases de notre carré dans n'importe quelle direction).

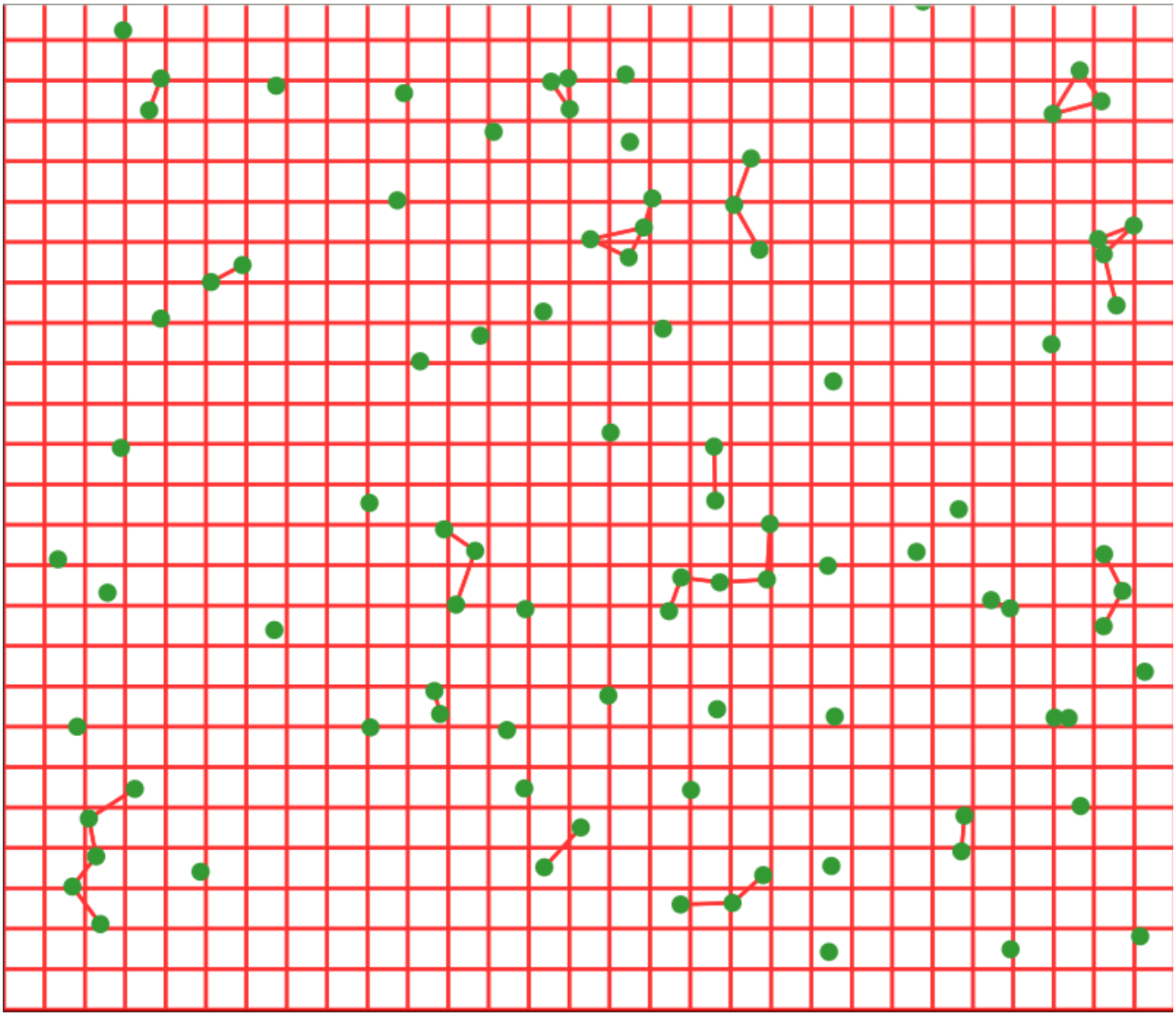
3-Si on retrouve un adjacent, on l'ajoute dans **L** tout en mettant à jour la longueur.

4-On répète jusqu'à ce qu'on retrouve une liste **L** vide.

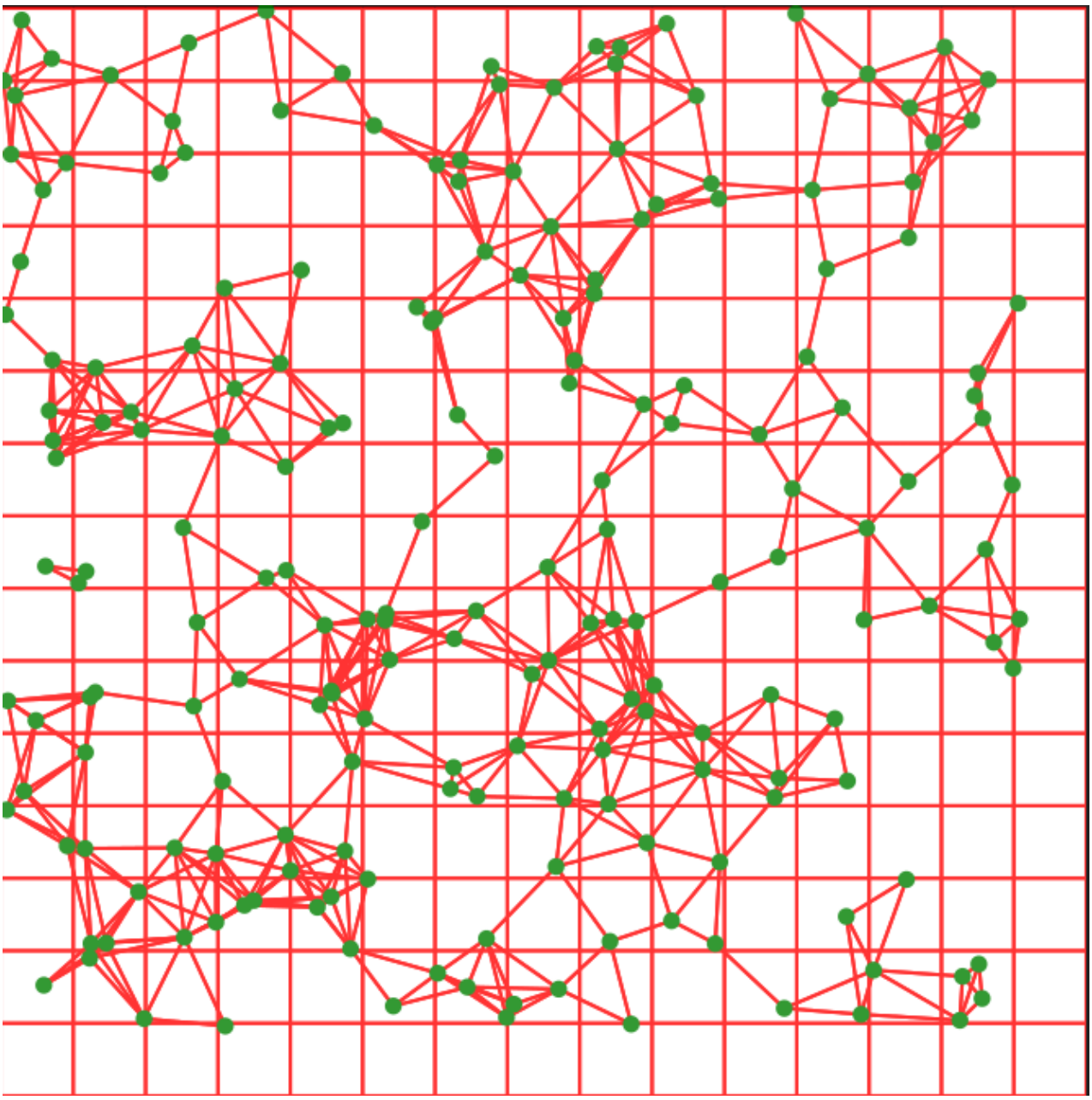
Résultat Graphique sur des exemples particuliers :



Test de 40 points



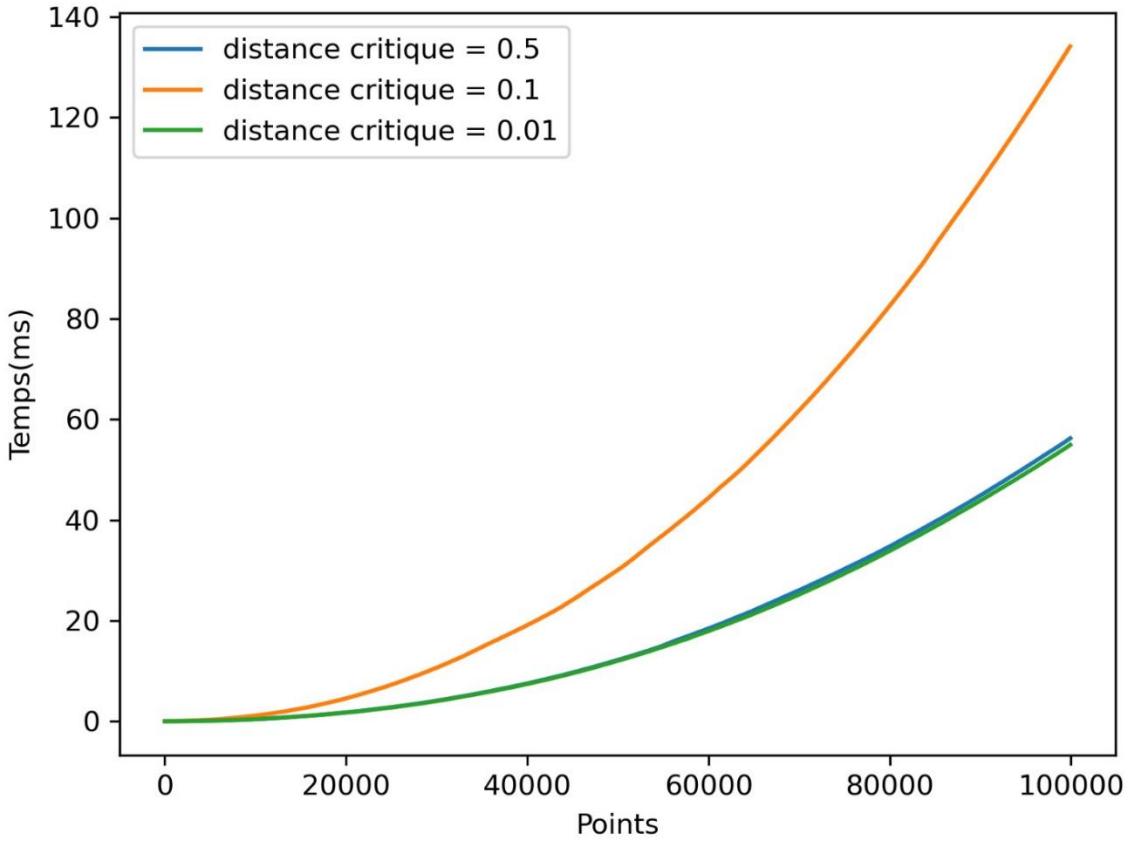
Test de 100 points



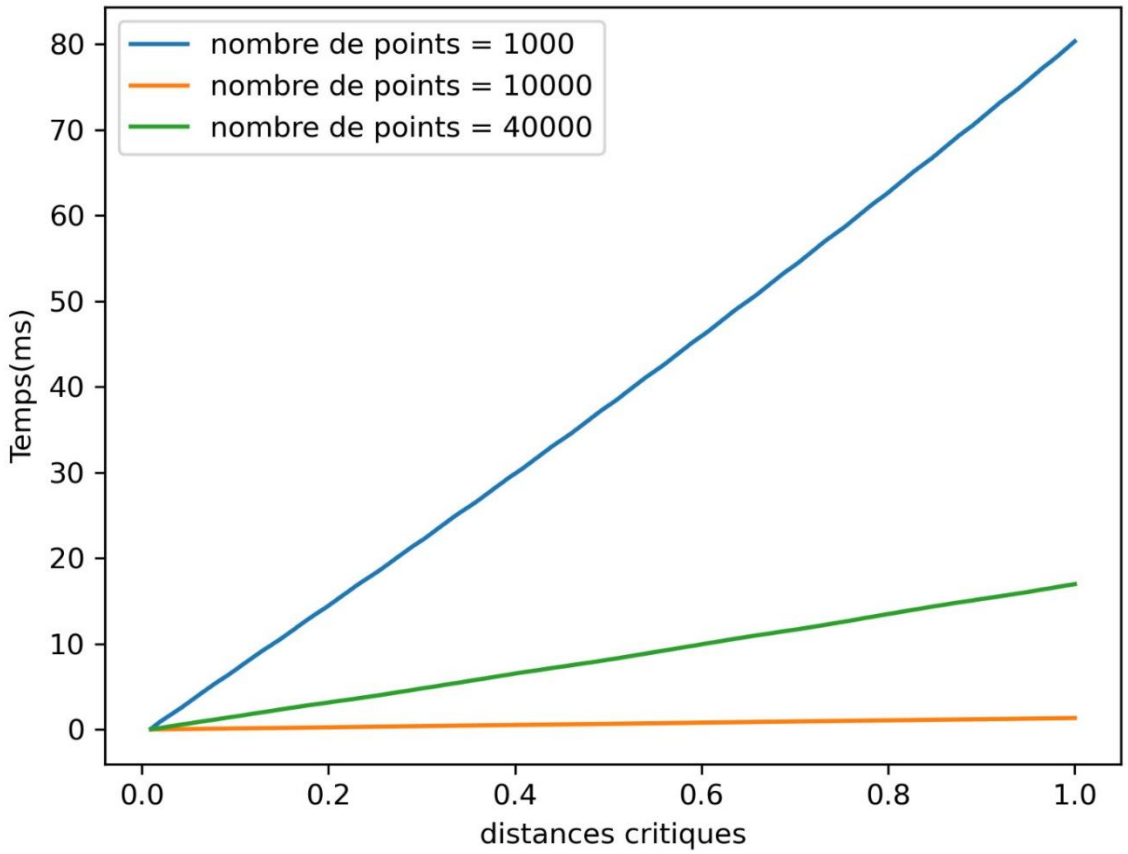
Test de 200 points

Performances :

On a exécuté les fichiers test sur un ordinateur de 8GO de RAM et un processeur i5-6300U.Les résultats obtenus sont les suivant :



Commentaire : On a créé environ 500 tests avec des nombres de points différents et on a obtenu ce diagramme qui représente l'influence de nombre de points sur la performance de notre algorithme. On remarque que si les distances critiques sont assez petites ou grandes, notre programme fonctionne rapidement.



Commentaire : Ainsi, on a créé environ de 500 fichiers tests de distances critiques différentes et on a obtenu ce graphe qui représente le temps en (ms) en fonction de la distance critique. On observe que la variation du temps par rapport à la distance critique est linéaire.