Taleb Rida n°: 3804264
Achraf Jday n° : 3802410
Professeur : Jean-Noël Vittaut

Flood IT : Contre-rendu

Introduction :

Le projet Flood-it est un jeu d'inondation de grilles aussi connu sous le nom de Mad Virus . Ce jeu est trouvable en ligne sur internet ou sur smartphone . L'objectif tout au long du projet a été de mettre en place différentes stratégies et de les étudier pour gagner au jeu du Flood-it .

Dans la partie 1 nous avons implémenter une séquence aléatoire pour le jeu . Plusieurs implémentation ont été comparées ainsi que leurs vitesse d'exécution .

Dans la partie 2 nous avons essayer de mettre en place différentes stratégies pour gagner au jeu d'inondation le plus rapidement possible en nombres d'itérations .

Nous allons effectuer des tests en fessons varier un des paramètres (Dimension , Nombre de couleur , Difficulté) et en fixant les autres et ainsi voir quelle implémentation de la séquence aléatoire entre celle de l'exercice 1 et 3 est la plus efficace .

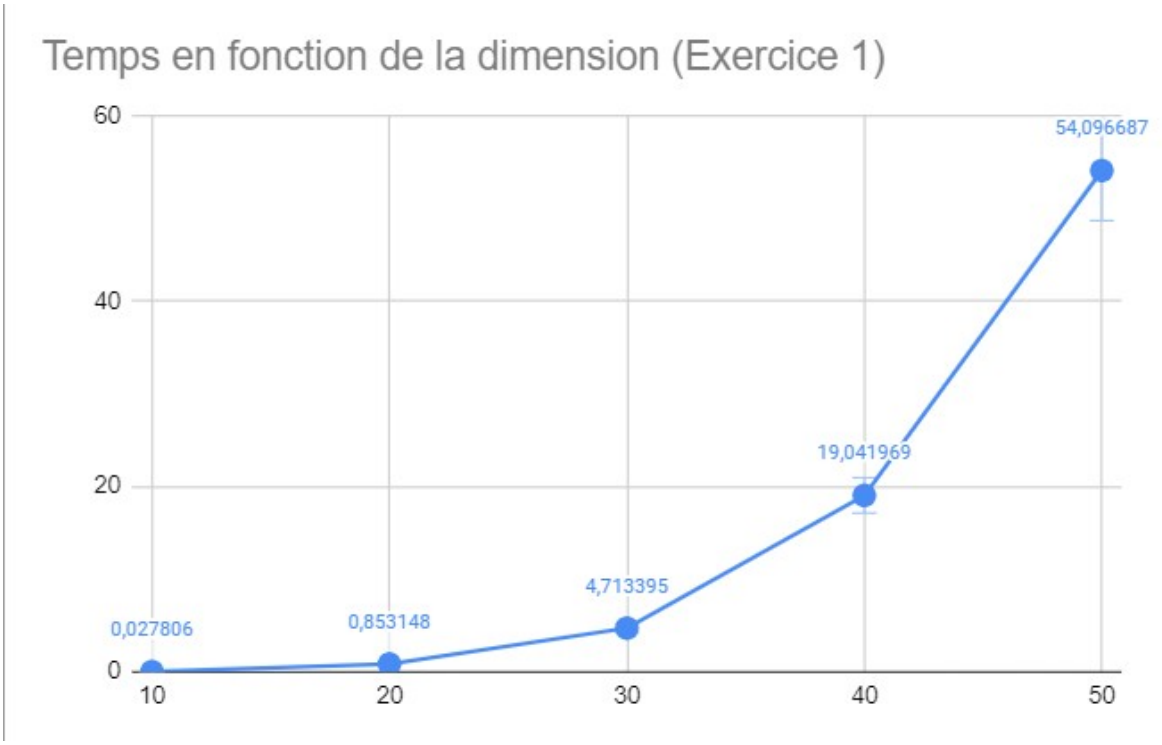
Pour les tests un script a été fait pour simplifier les tests de votre coté . Ce script s'appelle Test.sh et s'utilise de cette manière :
./Test.sh <paramètre>
Cela effectuera 10 test pour le paramètre donné .
Le .sh doit être modifié selon le test .

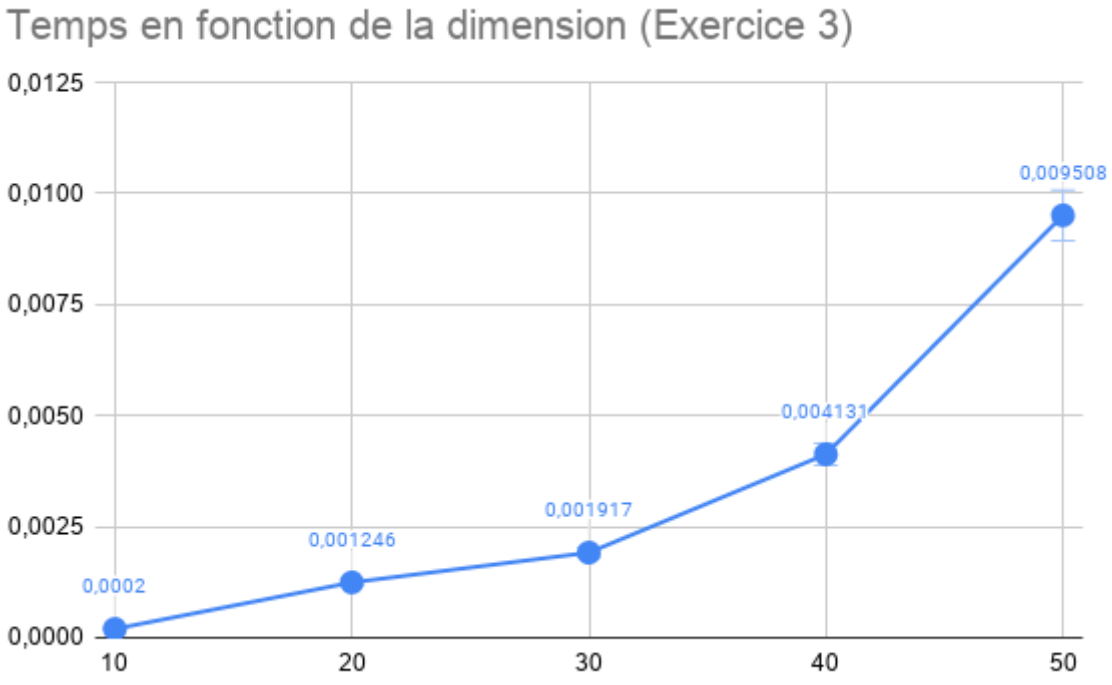
Test de dimension (nombre de couleur et difficulté fixe) :

Conditions du test :
Pour ce test nous ferons varier la valeur de la dimension de 10 à 50 de 10 en 10 et nous fixons la couleur à 25 et la difficulté à 1 . Pour chaque valeur

prise par la dimension 10 tests seront effectués et une moyenne des résultats sera effectué pour des raisons de précision .

Les résultats sous forme de graphe :





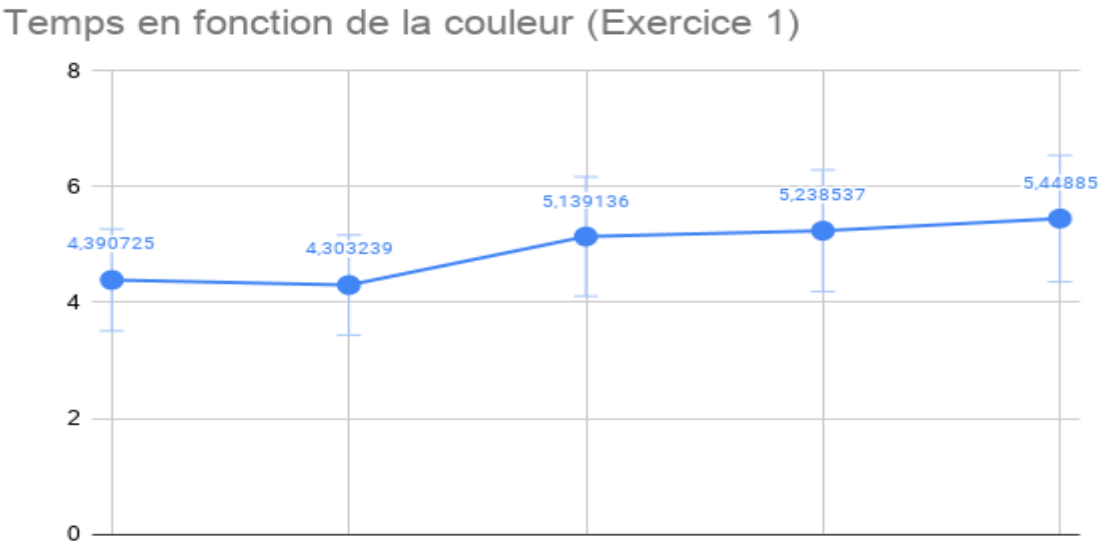
Remarque :

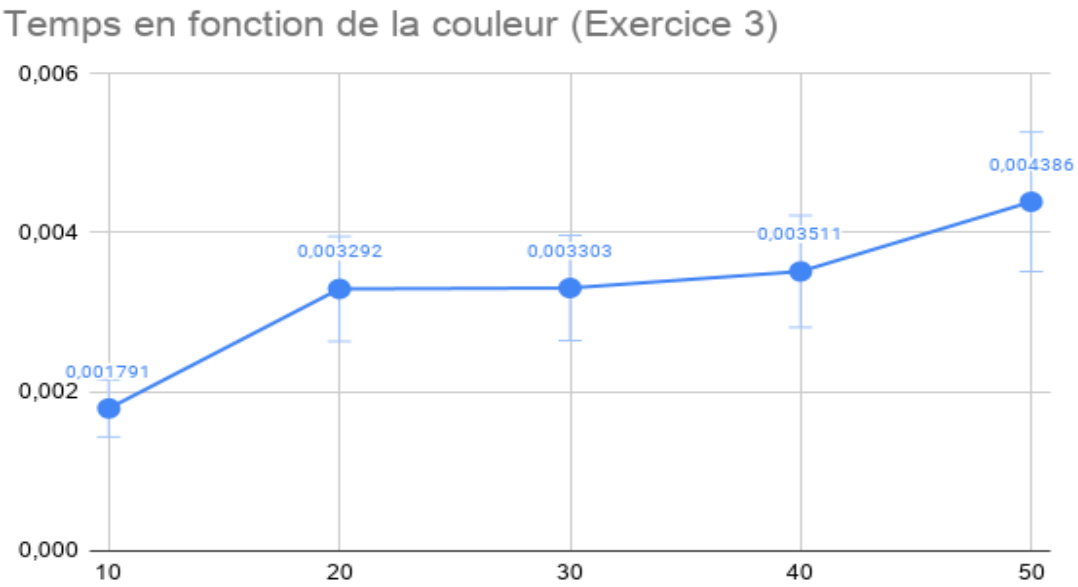
Nous pouvons observer une augmentation du temps d'exécution dans les 2 cas . Cela est lié à la dimension de la grille qui change . Par contre nous remarquons clairement que le temps d'exécution de l'exercice 3 est largement plus court que celui de l'exercice 1 dans les même conditions . Cela montre que l'implémentation de la séquence aléatoire en version itérative dans l'exercice 3 réduit considérablement le temps d'exécution et semble plus efficace car celui-ci conserve la liste_Zsg .

<u>Test avec le nombre de couleur</u> (dimension et difficulté fixe) :

Conditions du test :
Pour ce test nous ferons varier la valeur de la dimension de 10 à 50 de 10 en 10 et nous fixons la dimension à 30 et la difficulté à 1 . Pour chaque valeur prise par le nombre de couleur 10 tests seront effectués et une moyenne des résultats sera effectué pour des raisons de précision .

Les résultats sous forme de graphe :



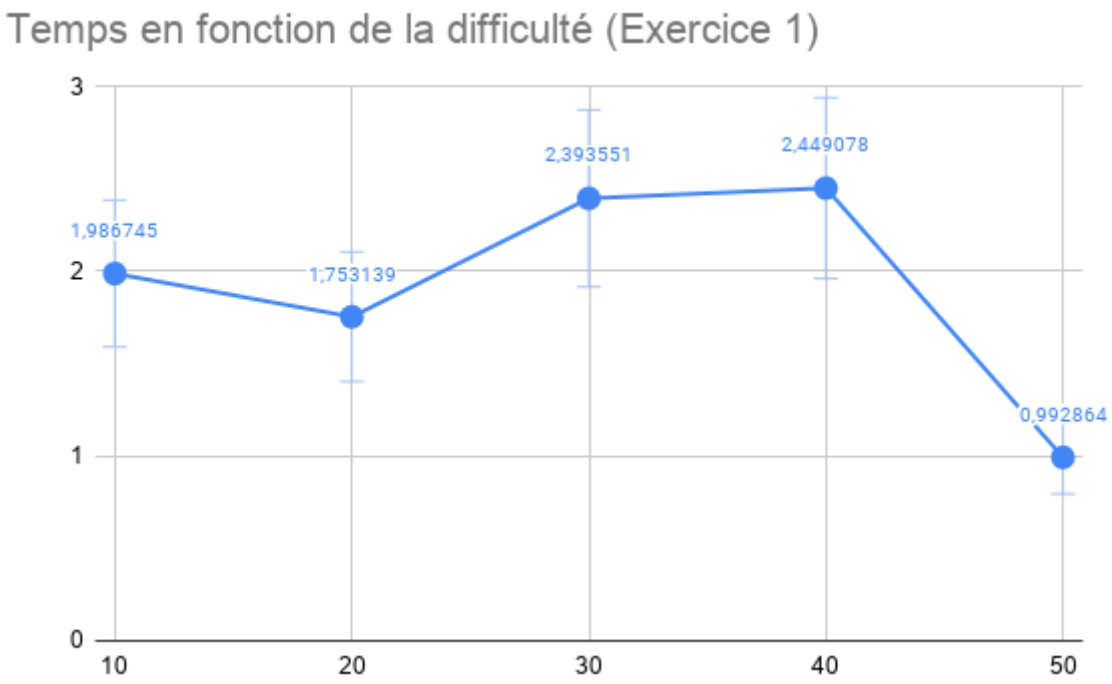


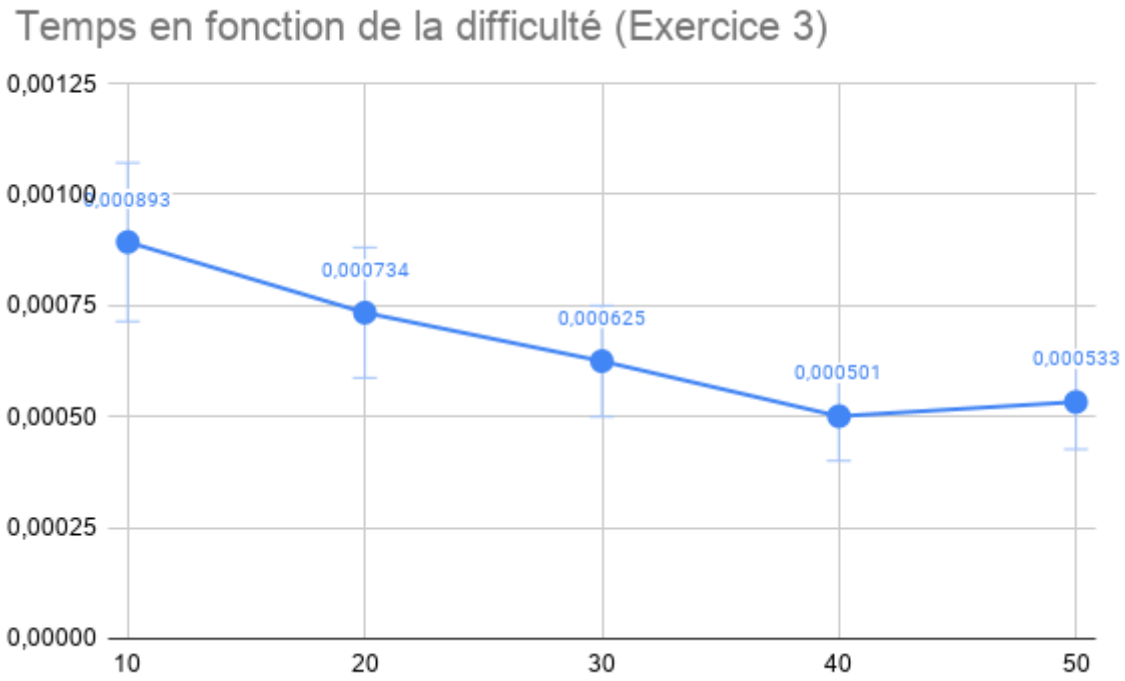
Remarque :
Dans le cas où nous fixons la dimension et la difficulté et que nous fessons augmenter le nombre de couleur nous pouvons remarquer que l'exercice 1 à un temps d'exécution qui reste plus au moins le même malgré de légère variation . La couleur n'est donc pas un facteur de lenteur pour l'exercice 1 . Pour l'exercice 3 l'évolution de la courbe montre bien que celui -ci augmente avec la variation du nombre de couleur mais reste comme même très rapide comparé à l'exercice 1 .

<u>Test avec la difficulté</u> (dimension et nombre de couleur fixe) :

Conditions du test :
Pour ce test nous ferons varier la valeur de la dimension de 10 à 50 de 10 en 10 et nous fixons la dimension à 30 et le nombre de couleur à 25 . Pour chaque valeur prise par la difficulté 10 tests seront effectués et une moyenne des résultats sera effectué pour des raisons de précision .

Les résultats sous forme de graphe :





Remarque :
La variation de la difficulté réduit le temps d'exécution dans les 2 exercices . Ce qui est normal car augmenter le paramètre de difficulté revient à permettre l'apparition de plus grande zone de même couleur . Mais nous pouvons remarquer que certaines valeurs dans le graphe de résultat de l'exercice 1 sont élevées et n'épouse pas l'allure de la courbe . Cela peut être lié au faite que parmi les valeurs obtenus lors des 10 exécutions de l'exercice 1 avec le paramètre difficulté à 30 et puis 40 nous ayons eux des valeurs plutôt élevées (sûrement à cause de l'environnement dans lequel à été lancer l'algo ou d'un ralentissement matériel) qui ont augmenter la valeur de la moyenne .

Conclusion global des tests :

Dans l'ensemble l'exercice 3 a un temps d'exécution record comparé à l'exercice 1 . Cela peut s'expliquer par le faite que implémentation de la séquence aléatoire itérative conserve la liste_Zsg grace à la structure S_Zsg tandis que dans l'exercice 1 nous utilisions une liste de case qui se reset à chaque tour ce qui a tendance à ralentir l'algorithme et à augmenter le temps d'exécution . Pour le cas de l'exercice 5 la création du temps des met énormément de temps à créer tout les sommets ce qui a pour

conséquence que le temps d’exécution est trop long donc nous l’avons pas mis dans les test .

Difficultés rencontrées :

En cette période difficile moi et mon binôme avons travaillé à distance malgré que celui-ci n’a pas eux un accès continu à internet . Ce fut compliqué à cause du faite que la communication entre nous était simplement issue des message partagé entre nous ce qui n’est pas le mieux pour travailler sur un projet mais malgré tout nous avons essayer . Ce fut un plaisir d’apprendre avec vous .