|  |
| --- |
| AMMOUR amine, QUEST Alexis, MISTRY darshan |
| Projet Flood-it |
| LU2IN006 |

|  |
| --- |
|  |

**Question 1.3 :**

On a fait plusieurs tests, avec ce premier algorithme :

|  |  |  |
| --- | --- | --- |
| Nombre de couleurs | Nombre d’essais | temps d’exécution |
| 2 | 3 | 0,000041 |
| 4 | 13 | 0,000089 |
| 8 | 19 | 0,00007 |
| 15 | erreur seg | erreur seg |
| 20 | erreur seg | erreur seg |

***Test n°1 : temps d’exécution et nombre d’essais de l’algorithme en fonction du nombre de couleurs pour une dimension de 5***

|  |  |  |
| --- | --- | --- |
| Nombre de couleurs | Nombre d’essais | temps d’exécution |
| 2 | 6 | 0,000288 |
| 4 | 21 | 0,000883 |
| 8 | 38 | 0,000899 |
| 15 | erreur seg | erreur seg |
| 20 | erreur seg | erreur seg |

***Test n°2 : temps d’exécution et nombre d’essais de l’algorithme en fonction du nombre de couleurs pour une dimension de 10***

|  |  |  |
| --- | --- | --- |
| Nombre de couleurs | Nombre d’essais | temps d’exécution |
| 2 | 6 | 0,000606 |
| 4 | 24 | 0,003342 |
| 8 | 68 | 0,007684 |
| 15 | 151 | 0,010325 |
| 20 | 164 | 0,013747 |

***Test n°3 : temps d’exécution et nombre d’essais de l’algorithme en fonction du nombre de couleurs pour une dimension de 15***

|  |  |  |
| --- | --- | --- |
| Nombre de couleurs | Nombre d’essais | temps d’exécution |
| 2 | 7 | 0,002519 |
| 4 | 32 | 0,010278 |
| 8 | 94 | 0,016768 |
| 15 | erreur seg | erreur seg |
| 20 | erreur seg | erreur seg |

***Test n°4 : temps d’exécution et nombre d’essais de l’algorithme en fonction du nombre de couleurs pour une dimension de 20***

On remarque une certaine régularité et logique dans les résultats c’est-à-dire que plus il y’a de couleurs et/ou de case plus le temps d’exécution ainsi que le nombre d’essais est important cependant on remarque que pour 2 couleurs que ce soit sur un plateau de 10 cases ou 15 cases l’algorithme résout le problème en un même nombre d’essais, 6 en l’occurrence

On s’aperçoit également des faiblesses de ce modèle « récursif » puisque lorsque l’on utilise de grand nombre pour tester le programme on remarque qu’à partir d’un moment une Erreur de segmentation survient.

Une image contenant dessin

Description générée automatiquement

***Screenshot pris pendant l’exécution de flood-it***

**Question 3.4 :**

On remarque une nette améliorations des performances du programme par rapport au modèle récursif concernant le temps d’exécution, cependant on remarque que le nombre d’itérations est toujours le même

**Question 3.5 :**

On a fait plusieurs test, avec ce second algorithme :

|  |  |  |
| --- | --- | --- |
| Nombre de couleurs | Nombre d’essais | temps d’exécution |
| 2 | 2 | 0,000024 |
| 4 | 12 | 0,000033 |
| 8 | 18 | 0,000025 |
| 15 | 109 | 0,000041 |
| 20 | 143 | 0,000041 |

***Test n°5 : temps d’exécution et nombre d’essais de l’algorithme en fonction du nombre de couleurs pour une dimension de 5***

|  |  |  |
| --- | --- | --- |
| Nombre de couleurs | Nombre d’essais | temps d’exécution |
| 2 | 5 | 0,000051 |
| 4 | 20 | 0,000052 |
| 8 | 37 | 0,000054 |
| 15 | 117 | 0,000072 |
| 20 | 168 | 0,0006 |

***Test n°6 : temps d’exécution et nombre d’essais de l’algorithme en fonction du nombre de couleurs pour une dimension de 10***

|  |  |  |
| --- | --- | --- |
| Nombre de couleurs | Nombre d’essais | temps d’exécution |
| 2 | 5 | 0,000055 |
| 4 | 23 | 0,000081 |
| 8 | 67 | 0,000092 |
| 15 | 150 | 0,000151 |
| 20 | 163 | 0,000157 |

***Test n°7 : temps d’exécution et nombre d’essais de l’algorithme en fonction du nombre de couleurs pour une dimension de 15***

|  |  |  |
| --- | --- | --- |
| Nombre de couleurs | Nombre d’essais | temps d’exécution |
| 2 | 6 | 0,000102 |
| 4 | 31 | 0,000137 |
| 8 | 94 | 0,000183 |
| 15 | 219 | 0,000294 |
| 20 | 327 | 0,000312 |

***Test n°8 : temps d’exécution et nombre d’essais de l’algorithme en fonction du nombre de couleurs pour une dimension de 20***

***Comparaison du temps d’exécution entres les deux algorithmes pour un plateau de dimension variable avec 2 couleurs différentes***

On remarque aisément la différence de rapidité entre l’algorithme de l’exercice 1 et celui de l’exercice 3 en utilisant seulement le graphique précèdent. On peut donc en conclure que la structure acyclique est plus intéressantes dans ce cas que la version récursive.

***Partie 2 :***

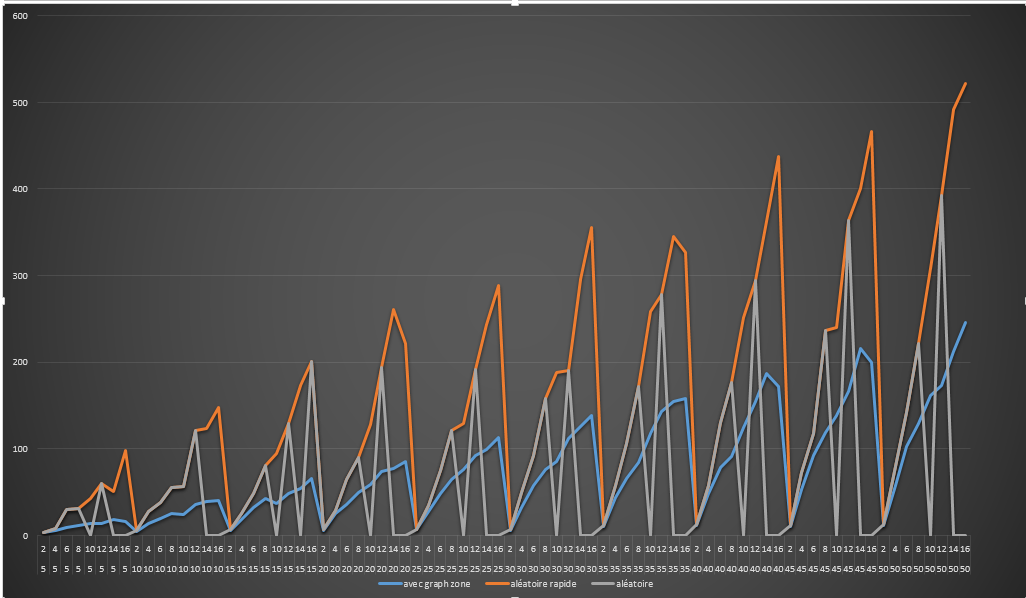
**Question 6.1 :**

Dans notre cas le graphe zone G est composé d’un ensemble de sommets. Chaque sommet possède une couleur ainsi qu’une liste de sommets adjacents Le but est de trouver un parcours en largeur de notre graphe

**Question 8.1 :**

Afin de faire une comparaison détaillée entre les différentes stratégies, nous avons créé un script en BASH. Ce script permet d’exécuter chacune des trois stratégies en faisant varier, la dimension, le nombre de couleur ainsi que la difficulté. Nous avons donc exécuté 800 configurations et nous avons récupéré les données dans un tableau. Pour chaque stratégie nous avons le temps d’exécution et le nombre d’essai.

Nous avons dans un premier temps fait un graph comparant le nombre d’essai en fonction de la dimension et du nombre de couleurs.

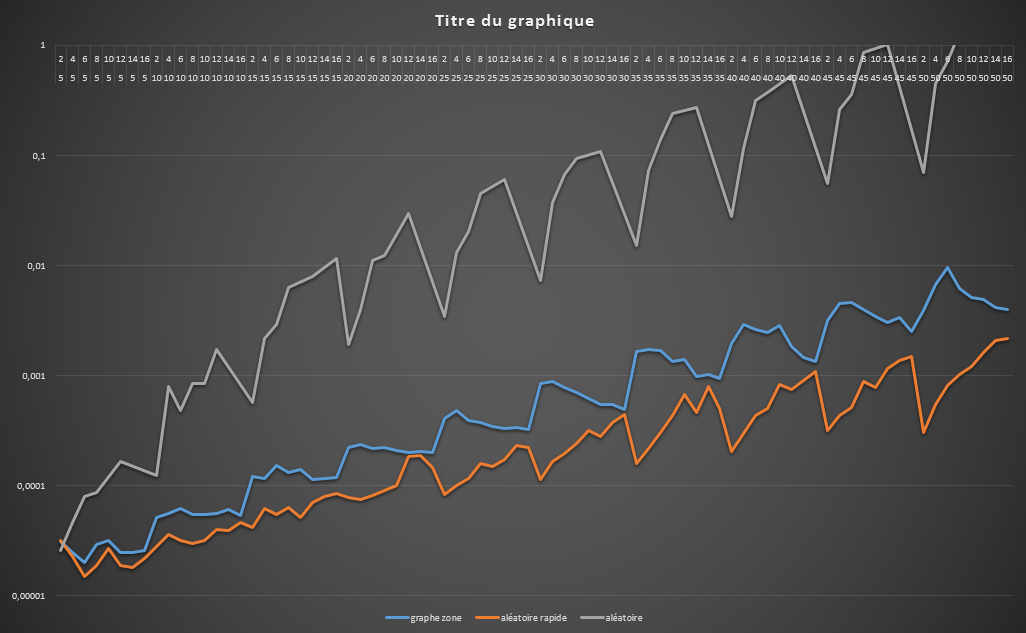


**Nombre d’essai en fonction de la dimension et du nombre de couleur**

On peut voir qu’il y a des cycles pour les trois courbes. Chaque fin de cycle représente une augmentation de la dimension et une réinitialisation du nombre de couleur. Qu’importe la dimension, quand il y a un nombre de couleur faible, le nombre d’essai est bas pour les trois stratégies. Néanmoins plus le nombre de couleur augmente, plus le nombre d’essai augmente. Et plus la dimension augmente, plus le nombre d’essai augmente.

Les deux stratégies aléatoires ont le même nombre d’essai pour chaque configuration sauf pour certain cas où la stratégie aléatoire non rapide n’a pas de valeur du à ……….. La stratégie graphe zone prend toujours moins d’essai que les deux autres, environ deux fois moins.

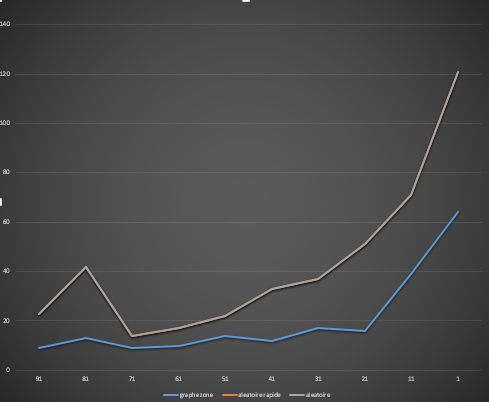
Nous avons ensuite refait le même graphique mais cette fois ci en comparant les temps d’exécution.



**Temps d’exécution en fonction de la dimension et du nombre de couleur**

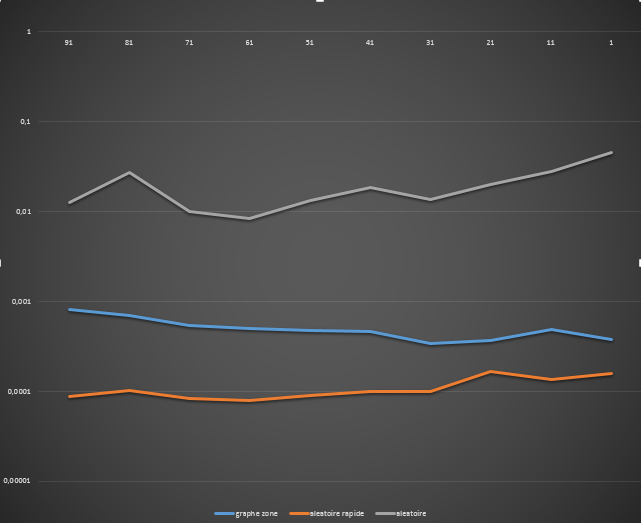
L’échelle du temps est logarithmique de base 10. On peut voir que la stratégie aléatoire est toujours la plus longue. La stratégie avec graph zone et la stratégie aléatoire rapide ont à peu près le même temps d’exécution, légèrement plus long pour le graphe zone.

Pour finir, nous avons fixé une dimension (25) et un nombre de couleur (8) et nous avons fait varier la difficulté



**Nombre d’essai en fonction de la difficulté**

Sans surprise, plus la difficulté est élevée, plus le nombre d’essai est important. Le nombre d’essais entre les deux versions aléatoires est toujours égal. C’est donc pour cela qu’on ne distingue qu’une seule droite.



**Temps d’exécution en fonction de la difficulté**

Le temps d’exécution, pour les trois stratégies, varie très peu selon la difficulté. Il augmente légèrement pour la version aléatoire non rapide sur les difficultés élevées. Il baisse légèrement pour la version graphe zone sur les difficultés élevées.