|  |
| --- |
|  |
| DM deMaths - 1 |
| Sujet : Tri Fusion |
|  |
| **Binôme B3302** |
| Robin Gicquel  Arnaud Mery de Montigny |
| **Le 9 décembre 2011** |

|  |
| --- |
|  |

1. **Protocole de validation**

**Test1 :** Nous vérifions le bon fonctionnement de la méthode separation

On crée une liste d’éléments non triée prédéfinie, donc on peut facilement créer manuellement les deux listes issues de la séparation de la première liste non triée. On exécute la méthode separation sur la liste non triée et on vérifie que les listes issues de la méthode sont identiques avec les listes séparées manuellement. Si c’est le cas on affiche sur la sortie standard le message suivant : *Test 1 validé*. Sinon on affiche le message suivant : *Test 1 faux*.

**Test2 :** Nous vérifions le bon fonctionnement de la méthode fusion

On crée deux listes d’éléments triées prédéfinies, donc on peut facilement créer manuellement la liste fusionnée. On exécute la méthode fusion sur les deux listes et on vérifie qu’elle est identique à la liste triée fusionnée manuellement. Si c’est le cas, on affiche sur la sortie standard le message suivant : *Test 2 validé*. Sinon on affiche le message suivant : *Test 2 faux*.

**Test3 :** Nous vérifions le bon fonctionnement de la méthode tri\_fusion

On crée une liste d’éléments non triée prédéfinie, donc on peut facilement créer manuellement la liste triée. On exécute la méthode tri\_fusion sur la liste non triée et on vérifie qu’elle est identique à la liste triée manuellement. Si c’est le cas on affiche sur la sortie standard le message suivant : *Test 3 validé*. Sinon on affiche le message suivant : *Test 3 faux*.

**Test4 :** Nous vérifions le bon fonctionnement de la méthode tri\_fusion\_inverse

On crée une liste d’éléments non triée prédéfinie, donc on peut facilement créer manuellement la liste triée en sens inverse. On exécute la méthode tri\_fusion\_inverse sur la liste non triée et on vérifie qu’elle est identique à la liste triée manuellement. Si c’est le cas on affiche sur la sortie standard le message suivant : *Test 4 validé*. Sinon on affiche le message suivant : *Test 4 faux*.

1. **Protocole de Mesure de Performance**

* Nous vérifions la performance du tri en fonction de différents paramètres.

Nous effectuons une boucle où l’on applique la méthode tri\_fusion. Au début de chaque boucle, on crée une liste aléatoire grâce à une méthode spécifique. La liste est créée avec un nombre d’éléments qui s’incrémente à chaque boucle.

A chaque boucle, on mesure le temps de l’exécution de la méthode tri\_fusion. On note aussi le nombre de permutations et de comparaisons pendant l’exécution de la méthode tri\_fusion.

Toutes ces valeurs sont stockées dans un fichier CSV qui peut être directement traité par un tableur. La réalisation de graphiques à partir d’un grand nombre de valeurs est ainsi facilitée. Une meilleure interprétation peut ainsi être faite.

* Nous vérifions la performance sur différentes configurations de listes

Nous mesurons le temps d’exécution de la méthode tri\_fusion pour des listes contenant les mêmes éléments, une qui est déjà triée, une autre qui est inversement triée et une dernière qui est agencée aléatoirement.

Pour obtenir des temps d’exécution représentatifs, on réalise ce test sur un très grand nombre de listes ayant un très grand nombre d’éléments. On retient le temps d’exécution total.

Ce protocole nous permet d’identifier dans quelle disposition ce tri est le plus efficace.

1. **Résultats**

* Les tests de validation sont tous positifs.
* Voici les résultats obtenus lors de la mesure de performances du tri fusion sur des listes de taille croissante :

On s’aperçoit que les courbes sont linéaires même si l’incertitude augmente avec le nombre d’éléments.

* En ce qui concerne les mesures de performances pour différentes configurations de listes, on obtient les valeurs suivantes :

Temps d'exécution pour 1000 tris de listes aléatoires de 20000 éléments : 10.046000 secs

Temps d'exécution pour 1000 tris de listes déjà triées de 20000 éléments : 5.585000 secs

Temps d'exécution pour 1000 tris de listes triées en sens inverse de 20000 éléments : 5.569000 secs

On se rend compte que le tri fusion est plus efficace sur des listes triées que sur des listes agencées aléatoirement. On remarque également qu’on obtient des temps d’exécution sensiblement similaires pour des listes triées en ordre croissant et des listes triées en ordre décroissant.

1. **Conclusions**

Il faut tout d’abord signaler que le fait que cet algorithme soit récursif limite les possibilités de test. En effet, la taille de la pile d’exécution est relativement limitée et arrive très rapidement à saturation dans le cas de mise en œuvre de fonctions récursives sur un grand nombre d’éléments. Il est donc impératif de « dérécursiver » ce genre d’algorithmes pour pouvoir les utiliser efficacement en pratique.

On a remarqué que cet algorithme est plus efficace pour une liste déjà triée, en sens inverse ou non et que son temps d’exécution est linéaire en fonction du nombre d’éléments à trier.