

Rapport du TME2-3-4

Partie 5 :

1) Afin de mettre en avant la rapidité et l'efficacité de chaque algorithme de recherche, nous allons, pour chacun d'entre eux, procéder à une série de recherche de morceaux par numéro, puis par titre et par artiste.

On travaille sur une base de 10000 morceaux afin d'effectuer ses tests.

On tire aléatoirement 5 numéros à chaque fois, qui permettront de définir les morceaux à rechercher. On considère que les morceaux recherchés existent tous dans les 10000 premières lignes de la base.

Les temps affichés par la suite sont en secondes et représentent le temps total nécessaire pour retrouver les 5 morceaux.

Recherche par numéro :

- Liste : 0.000294
- Arbre lexical : 0.010191
- Tableau dynamique : 0.000051
- Table de hachage : 0.001731

Recherche par titre :

- Liste : 0.001478
- Arbre lexical : 0.008405
- Tableau dynamique : 0.000311
- Table de hachage : 0.002705

Recherche par artiste :

- Liste : 0.005128
- Arbre lexical : 0.000034
- Tableau dynamique : 0.000850
- Table de hachage : 0.000020

On remarque que pour la recherche par numéro, la Liste et le Tableau Dynamique se démarquent du reste en terme de rapidité. Cela s'explique par le fait que la recherche se fait de manière linéaire. On retourne le morceau correspondant une fois trouvé.

Pour la recherche par titre, on peut effectuer le même constat que précédemment. La recherche étant linéaire sur la Liste et le Tableau Dynamique, on retrouve plus facilement les morceaux que dans les structures complexes comme l'Arbre Lexical ou la Table de hachage.

En revanche, lorsque la recherche est effectuée sur le nom de l'artiste, on remarque que les recherches s'effectuent très vite pour deux structures en particulier. En effet, comme l'Arbre Lexical est basé sur le nom des artistes afin d'effectuer ses liaisons et que la Table de hachage utilise comme clé la somme du code ASCII des caractères du nom de l'artiste, on arrive plus vite au résultat que dans les structures de recherche linéaire.

2) On modifie alors la taille de la Table de hachage afin de comparer les résultats obtenus. On effectue la compression sur la table de hachage pour les 100000 premières entrées de la base de données.

Les affichés par la suite sont en secondes.

Tailles de la Table de hachage :

- 100 => 2.912192
- 500 => 0.888030
- 1000 => 0.670746
- 5000 => 0.515367
- 10000 => 0.561136
- 30000 => 0.533017
- 60000 => 0.529126

On remarque donc que plus la taille de la table de hachage est importante, plus les temps de compression sont courts. Cela s'explique du fait qu'il y a moins de collisions entre éléments, et de ce fait, des accès aux listes de morceaux plus rapides et efficaces. On constate également que les temps de compression convergent vers une limite de temps (à partir de $T = 5000$ dans ce cas, on converge vers 0.5 secondes). On pourrait en conclure qu'il existe une taille optimale de table de hachage pour chaque situation.

3) On modifie le code du main afin de pouvoir récupérer sur des fichiers les temps nécessaires afin d'effectuer la recherche de morceaux uniques pour des tailles de base de données croissantes, en fonction de la structure.

On travaille sur les tailles suivantes :

10000, 30000, 50000, 100000, 150000, 200000, 250000, 300000.

Pour la structure en Liste, on obtient :

taille - temps

10000 - 1.248028

30000 - 22.033916

50000 - 57.462882

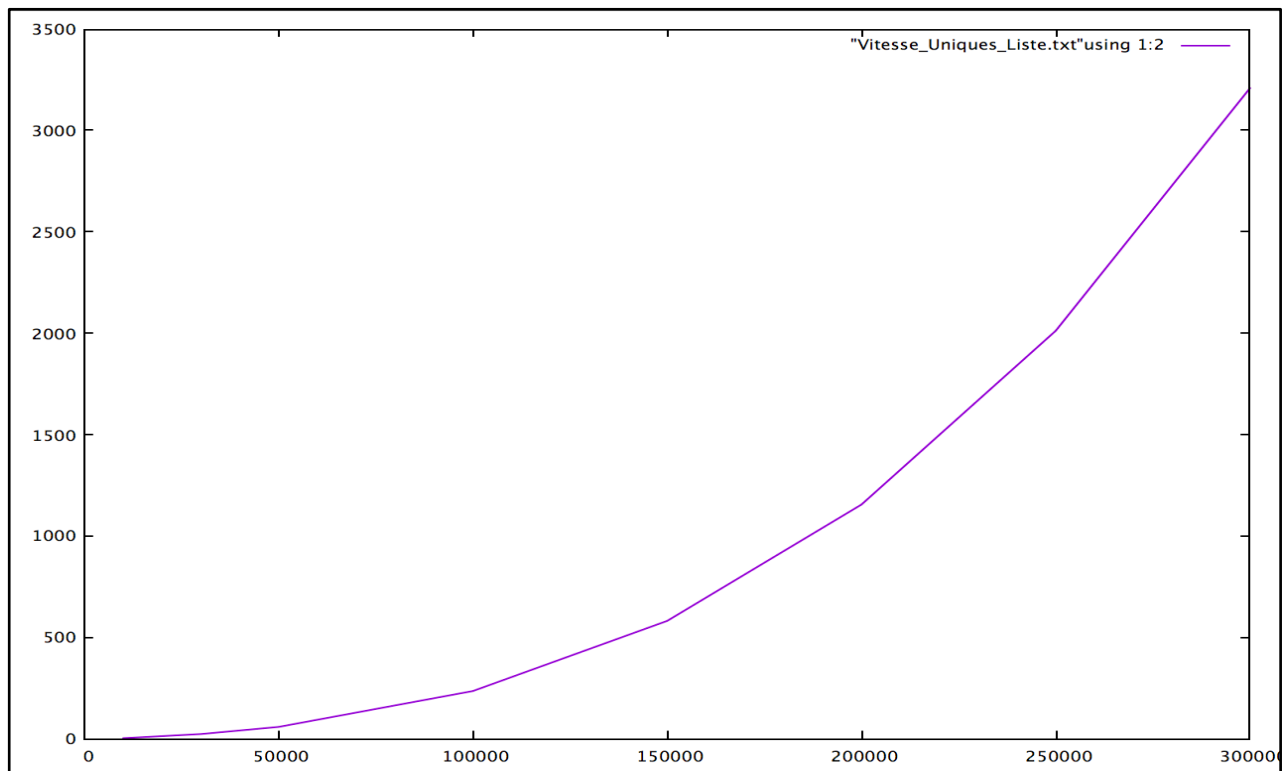
100000 - 233.809242

150000 - 580.70297

200000 - 1154.6

250000 - 2011.9

300000 - 3209.26



Titre – Représentation du temps de recherche de la bibliothèque des morceaux uniques d'une base de données en fonction de sa taille et de la structure de recherche en Liste.

Pour la structure en Arbre Lexical, on obtient :

taille - temps

10000 - 0.014928

30000 - 0.036684

50000 - 0.045866

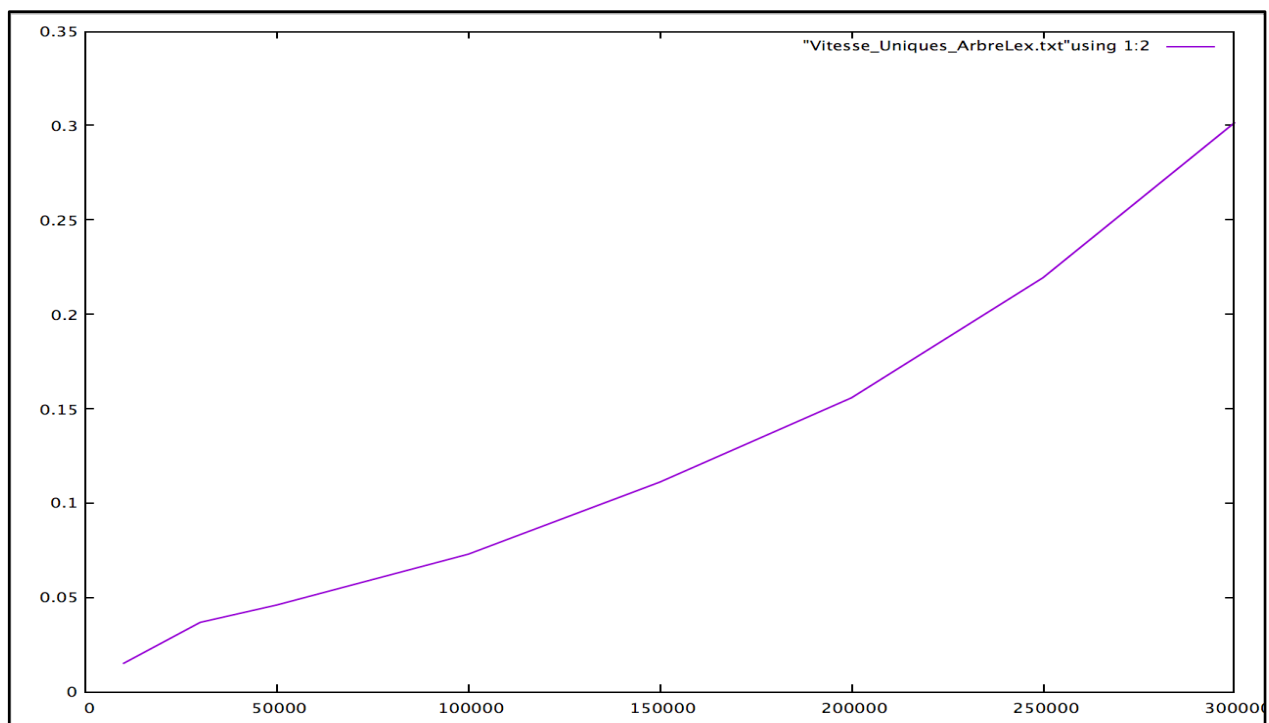
100000 - 0.072803

150000 - 0.111026

200000 - 0.155718

250000 - 0.219372

300000 - 0.301528



Titre – Représentation du temps de recherche de la bibliothèque des morceaux uniques d'une base de données en fonction de sa taille et de la structure de recherche en Arbre Lexical.

Pour la structure Tableau Dynamique, on obtient :

taille - temps

10000 - 0.91

30000 - 19.84

50000 - 51.45

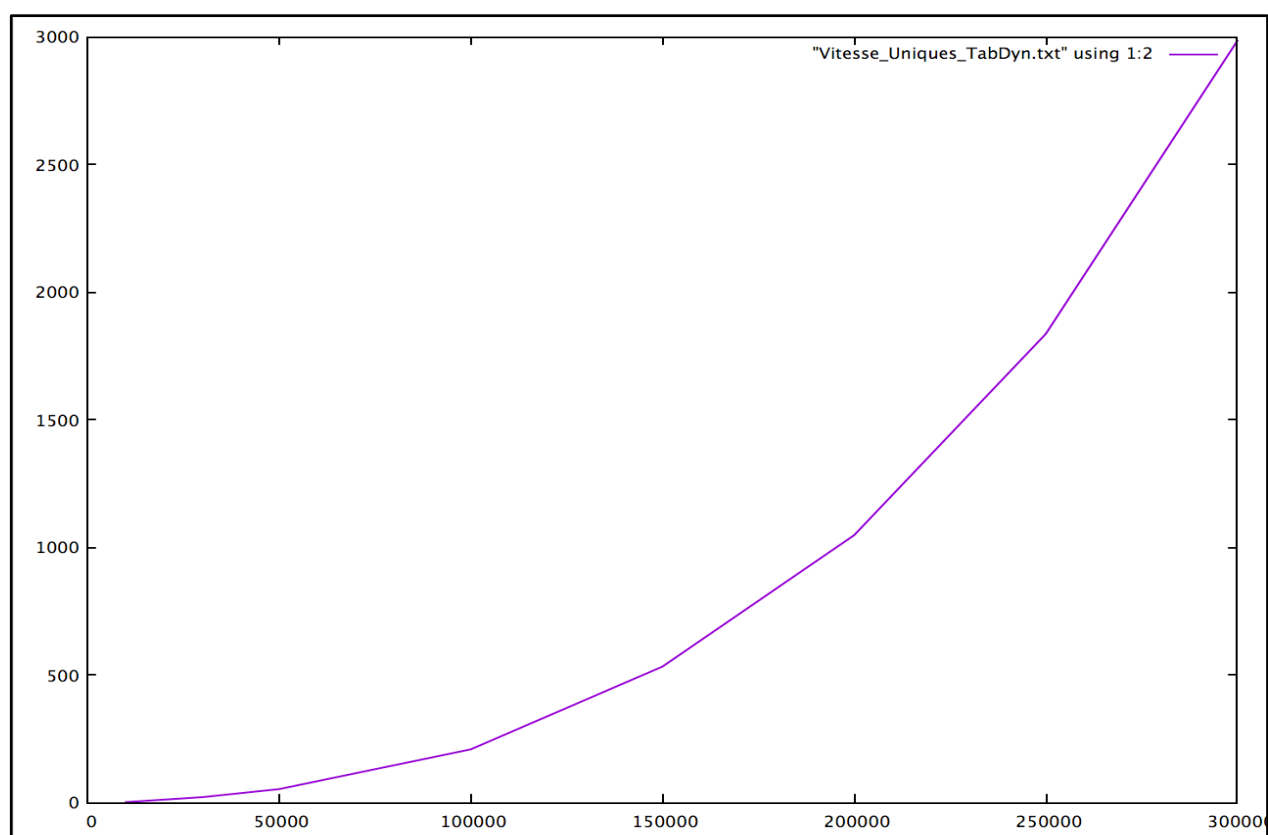
100000 - 207.37

150000 - 531.81

200000 - 1046.71

250000 - 1835.28

300000 - 2984.23



Titre – Représentation du temps de recherche de la bibliothèque des morceaux uniques d'une base de données en fonction de sa taille et de la structure de recherche en Tableau Dynamique.

Pour la structure Table de hachage, on obtient :

taille - temps

10000 - 0.005305

30000 - 0.024843

50000 - 0.045693

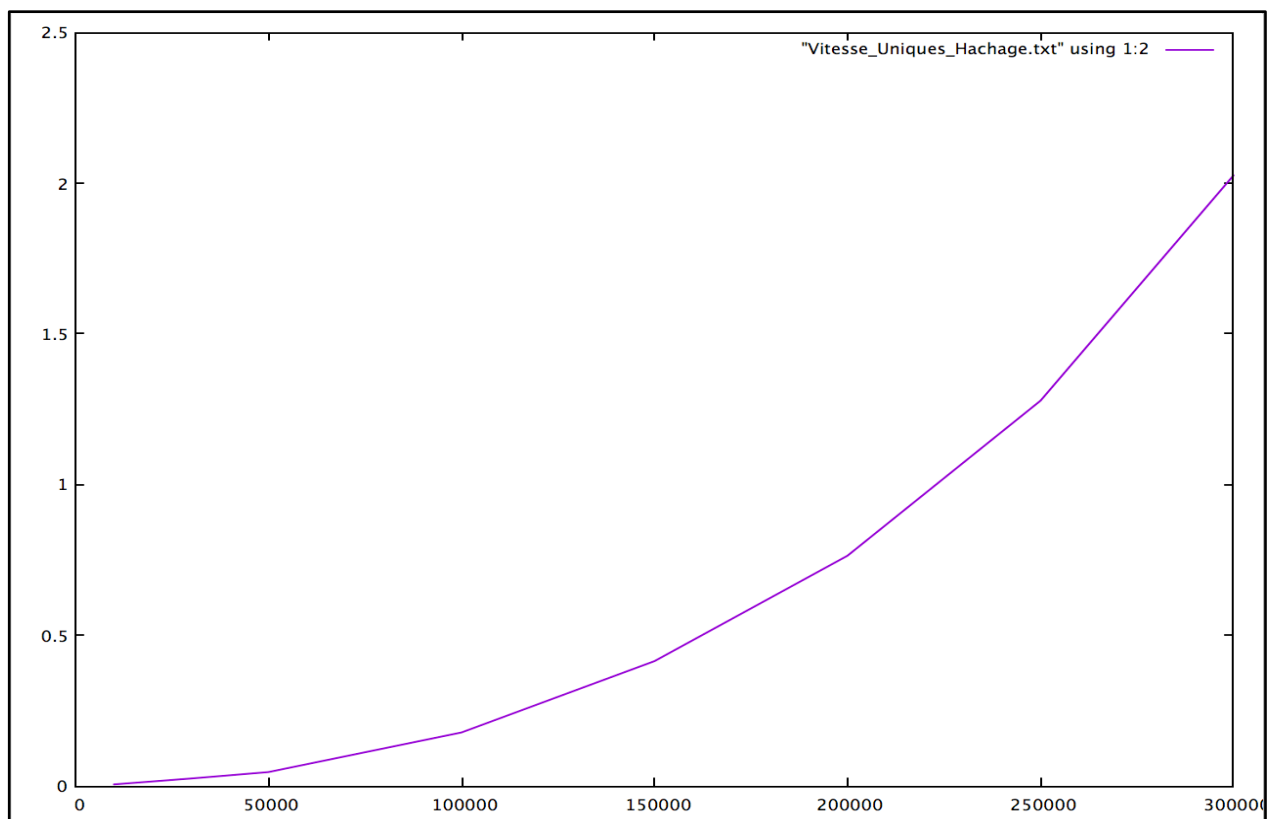
100000 - 0.177498

150000 - 0.413852

200000 - 0.763556

250000 - 1.278074

300000 - 2.025608



Titre – Représentation du temps de recherche de la bibliothèque des morceaux uniques d'une base de données en fonction de sa taille et de la structure de recherche en Table de Hachage.

(On note que la table a été utilisée pour une taille de 60000 éléments. Peut-être qu'en utilisant une table moins grande, on aurait pu obtenir de meilleurs résultats.)

4) On justifie les courbes obtenues avec les complexités pire-cas obtenues pour chaque structure de recherche.

Structure en Liste – La méthode de recherche est linéaire et on effectue N fois N comparaisons entre morceaux, avec N la taille de la liste. Les comparaisons prennent également un certain temps à s'effectuer. La complexité pire-cas est donc en $\Theta(n^2)$, soit quadratique polynomiale.

Structure en Arbre Lexical – La méthode de recherche est effectuée de manière FILO (First In Last Out) sur l'arbre. Lorsque l'on parcourt un noeud vide, aucune opération supplémentaire n'est effectuée. De plus, chaque nœud ne possède pas forcément une liste de morceaux et si elle existe, les comparaisons effectuées ne prennent en compte que le titre du morceau, puisqu'ils sont groupés par artistes. Dans ce contexte de recherche, la complexité pire-cas est donc en $\Theta(n * \log(n))$, soit quasi-linéaire.

Structure en Tableau Dynamique - Même chose que pour la structure en Liste, on effectue N fois N comparaisons entre morceaux. On note que les temps sont plus rapides que pour la structure en Liste puisque les opérations sur tableau sont (d'après mes essais), plus rapides que celles sur listes chaînées. La complexité pire-cas est donc en $\Theta(n^2)$, soit quadratique polynomiale.

Structure en Table de Hachage – En utilisant une structure en table de hachage, on effectue parcourt l'entièreté de la table, mais en raison de la grandeur de la table (60000 cases), il n'y a pratiquement aucune collision entre différents artistes, ce qui permet d'effectuer des comparaisons uniquement sur les titres. En raison de la taille importante, il est également que certaines cases ne référencent vers aucun artiste, donc aucune opération de comparaison supplémentaire devant être effectuée. La complexité pire-cas est donc en $\Theta(n * \log(n))$, soit quasi-linéaire.