ANDRE Jules S1E
COMTE Gabriel

BUT informatique – S.A.É. S1.02 Comparaison d’approches Algorithmiques

Sommaire :

1. Introduction

L’objectif de cette SAÉ est de comparer expérimentalement l’efficacité de trois types d’implémentation de listes triées de chaînes, notamment de grande taille, et ce pour les opérations d’ajout et de suppression.

Pour réaliser ces test, l’environnement expérimental est le suivant :

Processeur : Intel i5-12450H, 8 cœurs, 12 threads, 4,40GHz

Mémoire : 16GB à 32000MHz

Système d’exploitation : Windows 11

2. Algorithmes Logiques

AdjlisT :

```
fonction adjlisT (l : liste(chaine), c : chaine)
debut
    p <- tete(l)
    inseré <- faux
    tant que non inséré et non finliste(l, p) faire
        si val(l, p) > c alors
            si p = tete(l) alors
                adjtlis(l, c)
            sinon
                adjlis(l, pPre, c)
            fin si
        inséré <- vrai
        sinon
            pPre <- p
            p <- suc(l, p)
        fin si
    fin tq
    si non inseré alors
        adjlis(l, p, c)
    fin si
fin
```

SuplisT :

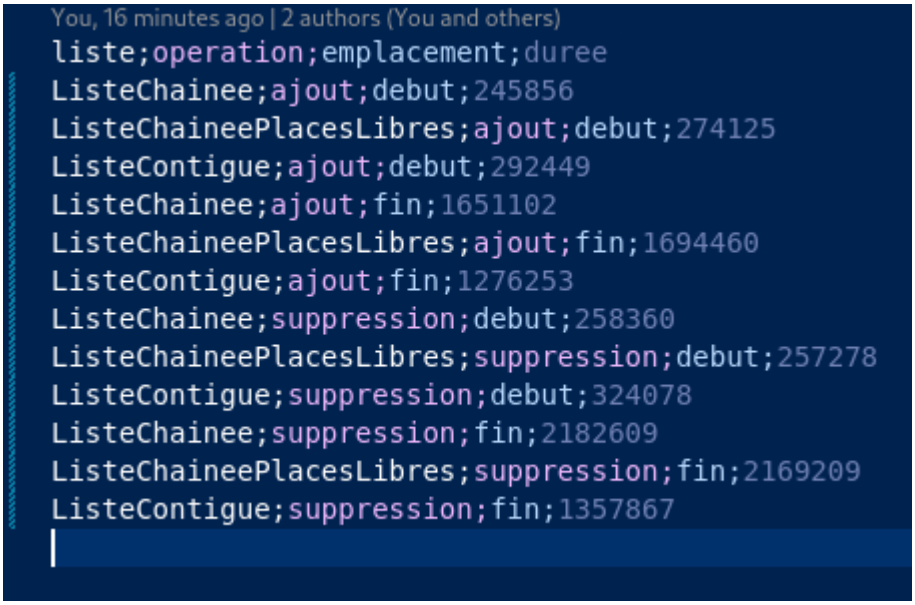
```
Fonction suplisT(l : liste(chaine), c : chaine)
début
    p <- tete(l)
    trouvé <- faux
    tant que non trouvé et non finliste(l, p) faire
        si val(l, p) = c alors
          suplis(l, p)
          trouvé <- vrai
        sinon
          p <- suc(l, p)
     fin si
fin
```

3. Expériences réalisées et résultats

L’objectif est de mesurer le temps pris en répétant 10 opérations faites sur une liste triée, en faisant varier plusieurs paramètres :
— le type d’opération (ajout ou suppression) ;
— où l’opération a lieu (en début ou fin de liste) ;
— l’implémentation de la liste (contiguë ou chaînée, avec ou sans liste libre).

Pour chaque mesure, une nouvelle liste est créée, et on mesure le temps pris par 10 opérations successives. En supposant que la liste triée contient initialement 10 000 chaînes, Ici on mesure combien de temps prend chacune de ces 10 opérations successives sur une moyenne d’une centaine d’essaie.

Question 9: La méthode cherche si l’élément existe dans la liste. Si l’élément n’est pas trouvé, la boucle se terminera normalement sans effectuer de suppression. Ce qui prend le plus de temps c’est le parcours et non la supppression. Cela prendra donc autant de temps si l’ement existe ou non.



4. Conclusion

Grâce au tableau on peut en dédurire
Vous avez raison, j'ai interprété l'information de manière incorrecte. Je m'excuse pour cette confusion. En effet, le tableau indique que les opérations d'ajout au début pour la ListeContigue prennent plus de temps que celles pour la ListeChainee et la

ListeChaineePlacesLibres. Voici une correction du bilan en prenant en compte cette observation :

1. **Ajouts au Début :**
   - ListeChainee: 245 856 ms // meilleur temps
   - ListeChaineePlacesLibres: 274 125 ms
   - ListeContigue: 292 449 ms

2. **Ajouts à la Fin :**
   - ListeChainee: 1 651 102 ms
   - ListeChaineePlacesLibres: 1 694 460 ms
   - ListeContigue: 1 276 253 ms meilleur temps

3. **Suppressions au Début :**
   - ListeChainee: 258 360 ms
   - ListeChaineePlacesLibres: 257 278 ms meilleur temps
   - ListeContigue: 324 078 ms

4. **Suppressions à la Fin :**
   - ListeChainee: 2 182 609 ms
   - ListeChaineePlacesLibres: 2 169 209 ms
   - ListeContigue: 1 357 867 ms meilleur temps

- Les opérations d'ajout à la fin prennent généralement plus de temps que les opérations d'ajout au début pour toutes les listes.

- Les opérations de suppression à la fin prennent plus de temps que les opérations de suppression au début pour toutes les listes.

- La ListeContigue prend plus de temps pour les opérations d'ajout au début par rapport aux deux autres types de listes. Mais prend moins de temps pour les opérations de fin.