

TP Listes

! Attention

Pour toute comparaison de performance entre deux algorithmes, pensez à compiler avec l'option -O3.

Partie I: Liste chaînée vs vecteur

1. Simplement chaînée

Le fichier vector.c contient l'implémentation de plusieurs fonctions opérant sur une liste dynamique et affiche les performances de recherche d'éléments ainsi que la quantité de mémoire utilisée.

- Modifiez ce code pour qu'il ajoute 50 caractères de plus dans le tableau au moment de l'initialisation, comparez les résultats.
- Implémentez les mêmes méthodes pour la liste chaînée suivante utilisant deux noeuds sentinelle.

```
typedef struct Node {
    struct Node* next;
    char elem;
} Node;

typedef struct LinkedList {
    Node* start;
    Node* end;
} LinkedList;

LinkedList* llist_new();
void llist_add_elem_start(LinkedList* ll, char elem);
LinkedList* llist_from_buffer(char* buffer, int size);
int llist_used_memory(LinkedList* ll);
Node* llist_rechercher(LinkedList* ll, Node* starting_at, char elem);
void llist_free(LinkedList* ll);
```

- Modifiez le programme afin qu'il affiche aussi les performances de cette liste:

Par exemple, le programme pourra afficher ceci:

```
===== Test tableau
Tableau chargé en ...
Mémoire utilisée pour 4058504 elements: ...
Mémoire utilisée après ajout de 50 caractères (4058554 elements): ...
26 parcours de tableau et 3356198 appels à la fonction rechercher en ...
===== Test liste
Liste chargée en ...
Mémoire utilisée pour 4058504 elements: ...
Mémoire utilisée après ajout de 50 caractères (4058554 elements): ...
26 parcours de liste et 3356198 appels à la fonction rechercher en ...
```

2. Doublement chaînée

- Copiez le code ou modifiez le afin d'utiliser une liste doublement chaînée. Plusieurs fonctions devront être modifiées.

Partie II Edition

3. Insertion

1. Créez les fonctions suivantes pour le vecteur:

- `vec_insere_caractere`: permet d'insérer un caractère au milieu de la liste (par exemple `vec_insere_char(v, 5, 'k')` insère le caractère 'k' à la position 5).

Note: Cette fonction augmente donc la taille de la liste.

- `insere_chaine`: permet d'insérer une chaîne de caractères au milieu de la liste (par exemple `vec_insere_chaine(v, 5, "coucou")` insère la chaîne "coucou" à la position 5).

Note: Cette fonction pourra réutiliser la fonction précédente.

2. Modifiez votre programme pour qu'il insère la chaîne "onjour" après toutes les occurrences de la lettre 'b' dans le texte en utilisant la fonction précédemment créée.
3. Affichez le temps d'exécution de cette opération ainsi que la taille utilisée par le tableau après modification.

Par exemple, le programme pourra afficher ceci:

```
===== Test tableau
Tableau chargé en ...
Memoire utilisée pour 4058504 elements: ...
Insertion effectué en ...
Memoire utilisée pour ... elements: ...
26 parcours de tableau et ... appels à la fonction rechercher en ...
===== Test liste
...
```

4. Liste chaînée

- Créez les mêmes fonctions pour la liste chaînée, comparez les .

Partie II Entre les deux

5. LinkedListVector

Afin de concilier les avantages de la liste chaînée et du vecteur, nous utiliserons une liste chaînée de petit vecteurs (ou unrolled list):

1. Complétez le code donné du fichier .c.
2. Comparez les performances et l'usage mémoire de cette structure de donnée avec celles utilisées précédemment lorsqu'on remplace 'b' par "bonjour".

```
===== Test tableau
...
===== Test liste
...
===== Test liste vecteur
...
```

#emph()[Il existe de nombreux exemples réels de listes déroulées plus optimisées, par exemple la quicklist de redis: <https://github.com/redis/redis/blob/unstable/src/quicklist.h>]