Trier une liste

Trier une liste chaînée

Voici l'algorithme que nous allons expliciter :

```
function sortLinkedList(ll) {
  let current = ll.head;
  if (!current || !current.next) {
    return;
  } else {
    const sortedList = new LinkedList();
    sortedList.addFirst(current.value);
    current = current.next;
    while (current) {
      let sortedHead = sortedList.head;
      while (sortedHead.next && sortedHead.next.value < current.value) {
        sortedHead = sortedHead.next;
      }
      if (sortedHead.value > current.value) {
        sortedList.head = new Node(current.value, sortedHead);
      } else {
        sortedHead.next = new Node(current.value, sortedHead.next);
      }
      current = current.next;
    }
    ll.head = sortedList.head;
  }
}
```

Copier

Nous commençons par vérifier que la liste a au moins deux éléments sinon nous n'avons pas à la trier.

Nous utilisons une nouvelle liste chaînée, la complexité mémoire sera donc de `O(n)` mais cela permet de faciliter la compréhension.

Nous commençons par ajouter un élément à la liste, peu importe sa valeur car nous nous en servirons comme premier point de comparaison pour l’insertion suivante.

Ensuite, nous sélectionnons le second élément de la liste à trier comme élément courant.

Nous itérons sur les éléments de la liste non triée avec une première boucle.

Pour chaque itération sur la liste non triée, nous allons considérer les éléments de la liste triée (boucle imbriquée).

Nous décalons le point d'insertion vers la droite tant que l'élément à insérer est supérieur à l'élément considéré dans la liste triée :

```
while (sortedHead.next && sortedHead.next.value < current.value) {
  sortedHead = sortedHead.next;
```

Copier

Autrement dit, à chaque fois que l'élément suivant de la liste trié est inférieur, nous regardons si l'élément encore d'après est également inférieur. Si c'est le cas nous décalons le point d'insertion.

Si il n'y a plus d'élément dans la liste trié ou si la valeur de l'élément d'après dans la liste trié est supérieure à la valeur de l'élément considéré, nous sortons de la boucle.

Nous sommes donc certain que le point d'insertion de l'élément considéré est bien immédiatement après le dernier élément strictement inférieur dans la liste triée, sauf si le second élément de la liste est supérieur ou qu'il n'y a qu'un seul élément dans la liste trié.

Dans ce cas, nous n'avons pas itéré sur la liste trié et nous sommes toujours avec :

```
let sortedHead = sortedList.head;
```

Copier

Ensuite, nous devons considérer deux cas, soit la tête de la liste a une valeur supérieur et donc l'élément considéré doit être la nouvelle tête de la liste trié :

```
if (sortedHead.value > current.value) {
  sortedList.head = new Node(current.value, sortedHead);
```

Copier

Sinon, nous l'insérons au point d'insertion qui est le dernier élément qui est inférieur à l'élément considéré.

A la fin nous modifions simplement la propriété `head` de la liste actuelle et passons la référence de la nouvelle liste.