N.Richard ALG3

NRI

October 8, 2021

Contents

1 ex 1.1

données :

- ListeChainée<T> liste
- T val

résultat : (néant) il faut introduire "val" dans "liste"

void insérerValeur(ListeChainée<T> liste, T val) liste.insérerTête(val) fin algo

2 ex 1.2

données :

- ListeChainée<T> liste
- T val

```
résultat : ElementListe<T>

ElementListe<T> recherche(ListeChainée<T> liste, T val)
        ElementListe<T> cur = liste.getPremier();
        while (cur != null && cur.getValeur() != val)
                cur = cur.getSuivant()
        fin-while
        // 2 possibilités:
        // - cur est null
        // - cur.getValeur() == val
```

```
        return cur
fin algo
```

3 ex 1.3

```
boolean contient(ListeChainée<T> liste, T val)
        return recherche(liste, val) != null
fin algo
```

4 1.4

4.1 version 1

```
boolean supprimerValeur(ListeChainée<T> liste, T val)
        /*
        Attention ceci ne convient pas car parcours inutile :
        if (!contient(liste, val))
                return false
        else
                // supprimer la valeur
        */
        // idée générale :
        // SI la valeur cherchée est en début de liste,
        // on supprime et on retourne vrai
        // SINON on cherche un ElementListe qui précède la valeur cherché
        ElementListe<T> prec = liste.getPremier()
        while (prec != null
                        && prec.getSuivant() != null
                        && prec.getSuivant().getValeur() != val) {
                prec = prec.getSuivant()
        }
        // REMARQUE: si prec == null, c'est que la liste est vide
        // 3 possibilités:
        // 1. prec == null, càd la liste vide (si prec==null)
        // 2. prec.getSuivant() == null  càd la valeur n'est pas trouvée
        // 3. sinon, c'est qu'on a trouvé la valeur
        if (prec == null)
                return false
        else if (liste.getPremier().getValeur() == val)
                liste.supprimerTête()
                return true
        else if (prec.getSuivant() != null)
                liste.supprimerAprès(prec)
                return true
        else
                return false
        fin if
fin algo
```

4.2 version 2

```
boolean supprimerValeur(ListeChainée<T> liste, T val)
        ElementListe<T> prec = liste.getPremier()
        if (prec == null)
                return false
        else if (prec.getValeur() == val)
```

```
                liste.supprimerTête()
                return true
        fin if

        while (prec.getSuivant() != null
                        && prec.getSuivant().getValeur() != val)
                prec = prec.getSuivant()
        fin while

        if (prec.getSuivant() != null)
                liste.supprimerAprès(prec)
                return true
        else
                return false
        fin if

fin algo
```

5 1.5

5.1 v1

```
int supprimerToutes(ListeChainée<T> liste, T val) {
    int cnt = 0;
    while (supprimerValeur(liste, val)){
        cnt++;
    }
    return cnt;
}
```

très lisible mais ça reparcourt la liste du début à chaque itération

5.2 v2

```
int supprimerToutes(ListeChaînée<T> liste, T val) {
    ElementListe<T> prec = null;
    ElementListe<T> cur = liste.getPremier();
    int cnt = 0;
    // contrainte: "prec.getSuivant() == cur" (sauf en tête de liste, prec = null)
    while (cur != null) {
        if (cur.getValeur() == val) {
            cnt++
            if (prec == null) {
                liste.supprimerTête();
                cur = liste.getPremier();
            } else {
                liste.supprimerAprès(prec);
                cur = prec.getSuivant();
            }
        } else {
            prec = cur;
            cur = cur.getSuivant();
        }
    }
    return cnt;
}
```

6 exo 3

6.1 3.1

```
void ajouterTrié(ListeChaînée<T> l, T val) {
    // trouver le bon endroit
    ElementListe<T> prec = null;
    ElementListe<T> cur = l.getPremier();
    while (cur != null && cur.getValeur() < val) {
        prec = cur;
        cur = cur.getSuivant();
    }
    // insérer : insérerTête ou insérerAprès(elt, val)
    if (prec == null) {
        l.insérerTête(val);
    } else {
        l.insérerAprès(prec, val);
    }
}
```

6.2 3.2

```
ElementListe<T> rechercheTriée(ListeChaînée<T> l, T val) {
    ElementListe<T> cur = l.getPremier();
    while (cur != null && cur.getValeur() < val) {
        cur = cur.getSuivant();
    }
    //3 possibilités:
    // cur == null, c'est-à-dire val ne se trouve pas là
    // cur.getValeur() > val : val ne s'y trouve pas non  plus
    // cur.getValeur() == val : val s'y trouve (dans cur)

    if (cur == null || cur.getValeur() > val)
        return null;
    else
        return cur;
}
```

6.3 3.3

```
boolean contientTriée(ListeChaînée<T> l, T val) {
    return rechercheTriée(l, val) != null;
}
```

6.4 3.4

```
boolean supprimerTriée(ListeChaînée<T> l, T val) {
    ElementListe<T> prec = null;
    ElementListe<T> cur = l.getPremier();
    while (cur != null && cur.getValeur() < val) {
        prec = cur;
        cur = cur.getSuivant();
    }
    if (cur == null || cur.getValeur() > val) {
        return false;
    }
    // ici on sait cur.getValeur() == val
```

```
    if (prec == null) {
        l.supprimerTête();
    } else {
        l.supprimerAprès(prec);
    }
    return true;
}
```

6.5 3.5

```
  int supprimerToutesTriées_v1(ListeChainée<T> l, T val) {
      ElementListe<T> prec = null;
      ElementListe<T> cur = l.getPremier();
      int cnt = 0;
      while (cur != null && cur.getValeur() == val) {
          l.supprimerTête();
          cur = l.getPremier();
          cnt++;
      }
      while (cur != null && cur.getValeur() < val) {
          prec = cur;
          cur = cur.getSuivant();
      }
      while (cur != null && cur.getValeur() == val) {
          l.supprimerAprès(prec);
          cur = prec.getSuivant();
          cnt++;
      }
      return cnt;
  }

//Autre version

  int supprimerToutesTriées_v2(ListeChainée<T> l, T val) {
      ElementListe<T> prec = null;
      ElementListe<T> cur = l.getPremier();
      int cnt = 0;
      while (cur != null && cur.getValeur() < val) {
          prec = cur;
          cur = cur.getSuivant();
      }
      while (cur != null && cur.getValeur() == val) {
          cur = cur.getSuivant();
          if (prec == null) {
              l.supprimerTête()
          }
          else {
              l.supprimerAprès(prec);
          }
          cnt++;
      }
      return cnt;
  }
```