Liste

cliquot

March 20, 2021

Contents

1 EXO 1

1.1 La cellule

1.1.1 Implémentation

Une cellule sera implémentée de cette façon :

- Une variable de type T qui sera notre valeur à stocker.
- Une cellule qui représente l'index de notre cellule suivante.

1.1.2 Les méthodes de la cellule

1. créerCellule O(1)

```
1 Cellule creerCellule()
2 {
3   Cellule c = new Cellule();
4   c.val = null;
```

```
5   c.suiv = null;
6   return c;
7 }
```

2. valeurCellule O(1)

```
1 T valeurCellule(Cellule c)
2 {
3   return c.val;
4 }
```

3. suivantCellule O(1)

```
1 Cellule suivantCellule(Cellule c)
2 {
3   return c.suiv;
4 }
```

4. insererSuivantCellule O(1)

```
1 Cellule insererSuivantCellule(Cellule c, Cellule s)
2 {
3   c.suiv = s;
4   return s;
5 }
```

5. dernierCellule O(1)

```
1 Cellule dernierCellule(Cellule c)
2 {
3   return suivantCellule(c) == null;
4 }
```

1.2 QUESTION 1

1.2.1 Implémentation

La liste sera implémentée de cette façon :

- Un tableau pour stocker nos cellules.
- Une pile pour connaître les cellules encore vides.
- un entier qui nous indiquera l'index de notre première cellule.

1.2.2 Les méthodes de la liste

1. créerListe O(1)

```
1 Liste creerListe(int n)
2 {
3   Liste l = new Liste();
4   l.tab = Cellule[n];
5   l.pile = creerPile(n);
6   for (int i = 0; i < n; ++i)
7     {
```

```
      empile(l.pile,tab[i]);
    }
  l.start = null;
  return l;
}
```

2. estVideListe O(1)

```
bool estVideListe(Liste l)
{
  return l.start = null;
}
```

3. teteList O(1)

```
Cellule teteListe(Liste l)
{
  return l.start;
}
```

4. insererTeteList O(1)

```
Liste insererTeteListe(Liste l,T val)
{
  if(!estVide(l.pile))
    {
      Cellule nouveauStart = sommet(l.pile);
      nouveauStart.val = val;
      insererSuivantCellule(nouveauStart,l.start);
      l.start = nouveauStart;
      depiler(l.pile);
    }
  return l;
}
```

5. supprimerTeteListe O(1)

```
Liste supprimerTeteListe(Liste l)
{
  if(!estPlein(l.pile))
    {
      empiler(l.pile,l.start);
      l.start = suivantCellule(l.start);
    }
  return l;
}
```

6. tailleListe O(n)

```
int tailleListe(Liste l)
{
  int size = 0;
  Cellule celluleActuelle = teteListe(l);
  while (celluleActuelle != null)
    {
```

```
      size++;
    }

  return size;
}
```

7. queue O(1)

```
Liste queue(Liste l)
{
  Liste retour = creerListe(n-1);
  retour.start = suivantCellule(teteListe(l));
  return retour;
}
```

8. obtenirElement O(i)

```
Cellule obtenirElement(Liste l,int i)
{
  Cellule celluleActuelle = teteListe(l);
  while(celluleAtuelle != null && i > 0)
    {
      i--;
      celluleActuelle = suivantCellule(celluleActuelle);
    }
  return valeurCellule(celluleActuelle);

}
```

9. insererElement O(i)

```
Liste insererElement(Liste l,T val,int index)
{
  if(index == 0)
    {
      insererTeteListe(l,val);
    }
  else {
    if(!estVide(l.pile))
      {
        Cellule celluleActuelle = teteListe(l);
        while (dernierCellule(celluleActuelle) && index > 1) {
          index--;
          celluleActuelle = suivantCellule(celluleActuelle);
        }
        Cellule celluleSuivante = suivantCellule(celluleActuelle);
        insererSuivantCellule(celluleActuelle,sommet(l.pile));
        depiler(l.pile);
        suivantCellule(celluleActuelle).val = val;
        insererSuivantCellule(suivantCellule(celluleActuelle),
   celluleSuivante);
      }
  }
  return l;
}
```

10. supprimerElement O(i)

```
Liste supprimerElement(Liste l,int index)
{
  if(index == 0)
    {
      return supprimerTeteListe(l);
    }
  else {
    Cellule celluleActuelle = teteListe(l);
    while(dernierCellule(celluleActuelle) != null && i > 1)
      {
        i--;
        celluleActuelle = suivantCellule(celluleActuelle);
      }
    if (index == 1 && !estPlein(l.pile)) {
      Cellule celluleASupprimer = suivantCellule(celluleActuelle);
      insererSuivantCellule(celluleActuelle,suivantCellule(
   celluleASupprimer));
      empiler(celluleASupprimer);
    }
    return l;
  }
}
```

1.3 QUESTION 2

1.3.1 Implémentation

La liste sera implémentée de cette façon :

- Une seule cellule qui représentera la tête de notre tableau.

1.3.2 Les méthodes de la liste (même complexité en temps que la QUESTION 1)

1. créerListe()

```
Liste creerListe()
{
  Liste l = new Liste();
  l.tete = null;
  return l;
}
```

2. estVideListe

```
bool estVideListe(Liste l)
{
  return teteListe(l) == null;
}
```

3. teteListe

```
Cellule teteListe(Liste l)
{
  return l.tete;
}
```

4. insererTeteListe

```
Liste insererTeteListe(Liste l,T val)
{
  Cellule ancienneTete = teteListe(l);
  l.tete = creerCellule();
  l.tete.val = val;
  insererSuivantCellule(teteListe(l),ancienneTete);
  return l;
}
```

5. supprimerTeteListe

```
Liste supprimerTeteListe(Liste l)
{
  Cellule nouvelleTete;
  if (teteListe(l) != null) {
    nouvelleTete = suivantCellule(teteListe(l));
  }
  else
    {
      nouvelleTete = null;
    }
  free(teteListe(l));
  l.tete = nouvelleTete;
  return l;
}
```

6. tailleListe

```
int tailleListe(Liste l)
{
  int size = 0;
  Cellule celluleActuelle = teteListe(l);
  while (celluleActuelle != null)
    {
      size++;
    }

  return size;
}
```

7. queue

```
Liste queue(Liste l)
{
  Liste retour = creerListe();
  retour.tete = suivantCellule(teteListe(l));
  return retour;
}
```

8. obtenirElement

```
Cellule obtenirElement(Liste l,int i)
{
```

```
  Cellule celluleActuelle = teteListe(l);
  while(celluleAtuelle != null && i > 0)
    {
      i--;
      celluleActuelle = suivantCellule(celluleActuelle);
    }
  return valeurCellule(celluleActuelle);

}
```

9. insererElement

```
Liste insererElement(Liste l,T val,int index)
{
  if(index == 0)
    {
      insererTeteListe(l,val);
    }
  else {
    Cellule celluleActuelle = teteListe(l);
    while (dernierCellule(celluleActuelle) && index > 1) {
      index--;
      celluleActuelle = suivantCellule(celluleActuelle);
    }
    Cellule celluleSuivante = suivantCellule(celluleActuelle);
    insererSuivantCellule(celluleActuelle,creerCellule());
    suivantCellule(celluleActuelle).val = val;
    insererSuivantCellule(suivantCellule(celluleActuelle),
   celluleSuivante)
  }
  return l;
}
```

10. supprimerElement

```
Liste supprimerElement(Liste l,int index)
{
  if(index == 0)
    {
      return supprimerTeteListe(l);
    }
  else {
    Cellule celluleActuelle = teteListe(l);
    while(dernierCellule(celluleActuelle) != null && i > 1)
      {
        i--;
        celluleActuelle = suivantCellule(celluleActuelle);
      }
    if (index == 1) {
      Cellule celluleASupprimer = suivantCellule(celluleActuelle);
      insererSuivantCellule(celluleActuelle,suivantCellule(
   celluleASupprimer));
      free(celluleASupprimer);
    }
```

```
19      return l;
20    }
21  }
```

1.4 QUESTION 3

Pour avoir une liste chainée bi-directionelle, il faudrait rajouter dans la cellule une variable prec pour stocker la position de la cellule précédente (de la même façon que pour la variable suiv) ainsi que des méthodes precedenteCellule() et insertionPrecedenteCellule() qui vont modifier de la même façon que suiv la variable prec.

De plus partout ou on a modifié notre variable suiv, il faudrait modifier la variable prec de la cellule qui suit (sauf si le suivant est null) (ex insertionSuivantCellule(c1,c2) -> insertionPrecedentCellule(c2,c1)).

Pour la liste chainée circulaire (pas bi-directionelle) il faudra remplacer tous nos null par la valeur de la tete de liste (par exemple lorsqu'on on regardait si notre cellule et la dernière, il faudra faire suivantCellule(celluleActuelle) == teteListe(l)). Il faudra aussi bien penser lors de l'ajout d'un element qui se retrouve être le dernier ou le premier de la liste qu'on doit avoir la variable suiv du dernier bien modifié, de même lorqu'on supprime la tête (ce qui va consister à modifier inserer/supprimerTeteListe et rajouter une condition lorsque on ajoute/supprime le dernier element si on a défini les méthodes obtenir/supprimerElement comme dans la question 2 ainsi que la méthode queue) .

2 Exo 2

cette algorithme va renverser une liste.

3 Exo 3

Tout en O(1);

3.1 QUESTION 1

Le seul moyen pour avoir une concaténation de deux liste en temps O(1) et d'avoir des listes circulaires. En effet on va faire : (on suppose que insererPrecedentCellule ˜= insererSuivantCellule et precedentCellule ˜= suivantCellule. la seule différence étant qu'il modifie la valeur qui représente la cellule précédente).

```
1   void concatenation(Liste l1,Liste l2)
2 {
3   insererSuivantCellule(precedentCellule(teteListe(l1)),teteListe(l2));
4   insererSuivantCellule(precedentCellule(teteListe(l2)),teteListe(l1));
5
6   Cellule ancienDernierL1 = precedentCellule(teteListe(l1));
7
8   insererPrecedentCellule(teteListe(l1),precedentCellule(teteListe(l2))
     );
9   insererPrecedentCellule(teteListe(l2),ancienDernierL1);
10 }
```

3.2 QUESTION 2

3.3 QUESTION 3

3.3.1 PILE (est donc composée d'une liste l);

1. creerPile

```
1   Pile creerPile()
2 {
3   Pile p = new Pile();
4   p.l = creerListe();
5   return p;
6 }
```

2. estVide

```
1 bool estVide(Pile p)
2 {
3   return estVideListe(p.l);
4 }
```

3. empiler

```
1 Pile empiler(Pile p,T val)
2 {
3   insererTeteListe(p.l,val);
4   return p;
5 }
```

4. depiler

```
1 Pile depiler(Pile p)
2 {
3   supprimerTeteListe(p.l);
4   return p;
5 }
```

5. sommet

```
1 T sommet(Pile p)
2 {
3   return valeurCellule(teteListe(p.l));
4 }
```

3.3.2 FILE (est donc composée d'une liste l);

1. creerFile

```
1   Pile creerFile()
2 {
3   File f = new File();
4   f.l = creerListe();
5   return f;
6 }
```

2. estVide

```
1 bool estVide(File f)
2 {
3   return estVideListe(f.l);
4 }
```

3. emfiler

```
File empiler(File f,T val)
{
  insererTeteListe(f.l,val);
  return f;
}
```

4. defiler

```
File defiler(File f)
{
  Cellule derniereCellule = precedentCellule(teteListe(f.l));
  inserersuivantCellule(precedentCellule(precedentCellule(teteListe
   (f.l))),teteListe(f.l));
  free(derniereCellule);
  return f;
}
```

5. teteFile

```
T teteFile(File f)
{
  return valeurCellule(precedentCellule(teteListe(f.l)));
}
```

4 Exo 4

4.0.1 Fonction

```
  Liste inverser(Liste l)
{
  Cellule precedente;
  Cellule actuelle = teteListe(l);
  Cellule suivante = suivantCellule(actuelle);
while(suivant != null)
  {
    precedente = actuelle;
    actuelle = suivante;
    suivante = suivantCellule(suivante);
    insererCelluleSuivante(actuelle,suivante);
  }
 l.tete = actuelle;
 return l;
}
```