Algorithmique Avancée
Structures de données linéaires

Youness LAGHOUAOUTA

Institut National des postes et télécommunications
laghouaouta@inpt.ac.ma

Février 2022

Objectifs du cours

- Comprendre l'utilité des outils d'analyse des algorithmes
- Choisir les bonnes structures de données pour concevoir un algorithme
- Concevoir de nouvelles structures de données efficaces
- Comprendre certaines stratégies et les utiliser pour concevoir des algorithmes efficaces

Organisation du cours

1. Analyse des algorithmes + TD1
2. Algorithmes de tri + TD2
3. **Structures de données linéaires 1 + TD3**
4. Structures de données linéaires 2 + TD4
5. Dictionnaires +TD5
6. Arbres + TD6
7. Paradigmes et stratégies algorithmiques + TD7

Structure de données

- Une structure de données est une manière d'organiser et de conserver des données :
  - Traitement texte
  - Impression documents
  - Jeu des échecs
  - ...
- Une structure de données a une interface qui spécifie les opérations agissant sur les données (ajout, accès, suppression ...).
- Une structure de données conserve des données des méta-données.

Rappel - Tableau

Définitions

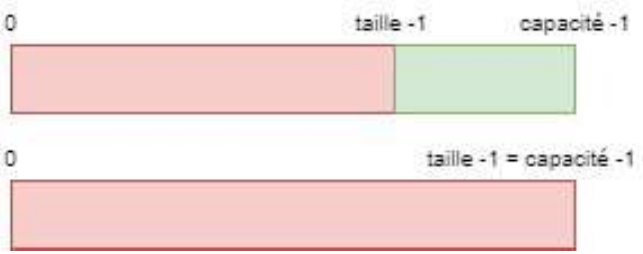
- Un tableau est une structure de données qui permet de stocker un certain nombre d'éléments repérés par leurs indices.
- Un tableau est un ensemble d'emplacements mémoire contigus en nombre fixé contenant le même type de donnée.

Figure: Tableau T de taille n

Variabilité de la taille

- Simulation d'un tableau de taille variable :
  - Réserver une quantité donnée en mémoire (capacité)
  - Utiliser une variable qui joue le rôle de la taille du tableau (méta-donnée)
  - Ranger les valeurs au début du tableau (mise à jour de la taille)
- Encapsulation du tableau dans une structure qui supporte la variation de la taille :
  - C : Pointeur
  - Java : ArrayList

Dépacement taille maximale

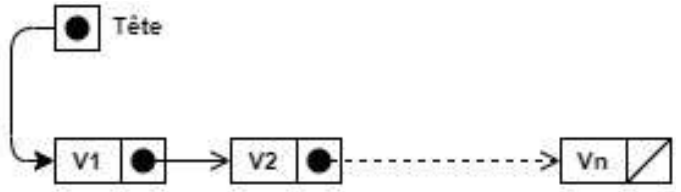


- Erreur (ArrayIndexOutOfBoundsException)
- Réallocation du tableau par recopie (System.arraycopy(...))

Liste chainée

Définitions

- Une liste chainée est une structure linéaire composée de maillons.
- Un maillon comprend un champ *valeur* et un champ *suivant* qui pointe vers le maillon suivant.
- Le dernier maillon pointe sur une valeur spéciale qui représente la notion de l'absence d'information (NIL, référence null en Java...)



Exemples opérations

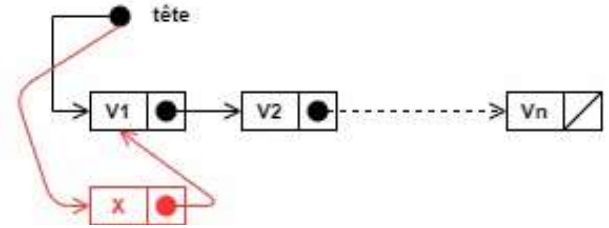


Figure: Insertion en tête

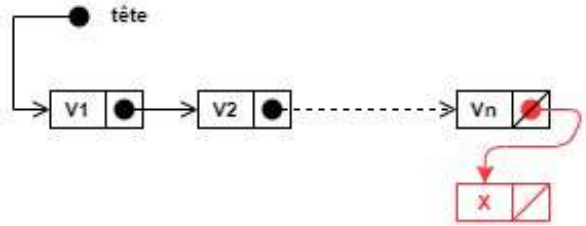


Figure: Insertion en queue

Exemples opérations

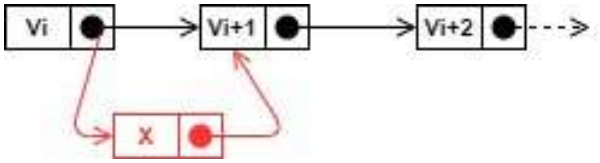


Figure: Insertion

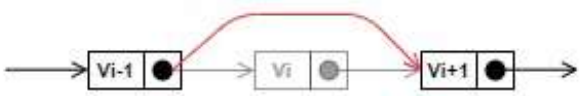


Figure: Suppression

Mode itératif Vs Mode récursif

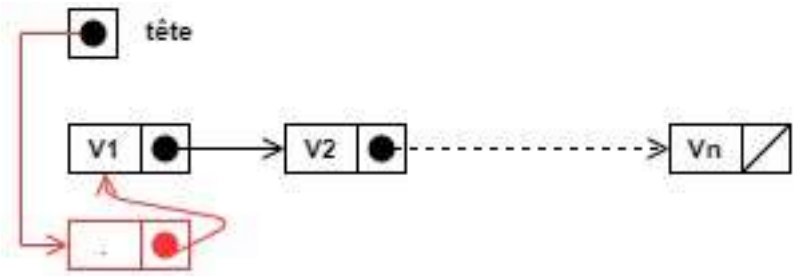
Mode itératif

```
int longueur(list lst) {
    int c = 0;
    list l = lst;
    while (estVide(l)){
        l = l.suivant;
        c++;
    }
    return c;
}
```

Mode récursif

```
int longueur(list lst) {
    if (estVide(lst)) return 0;
    else return 1 + longueur(lst.suivant);
}
```

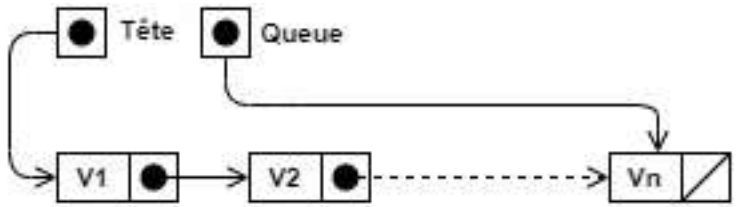

Liste chainée avec fausse tête



Intérêts

- Traiter indifféremment le premier maillon pour les opérations d'insertion et suppression.

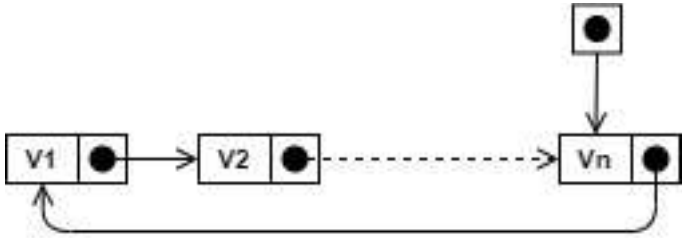
Liste chainée avec pointeurs de tête et de queue



Intérêts

- Ajout en queue sans parcours.
- Concaténation sans parcours.

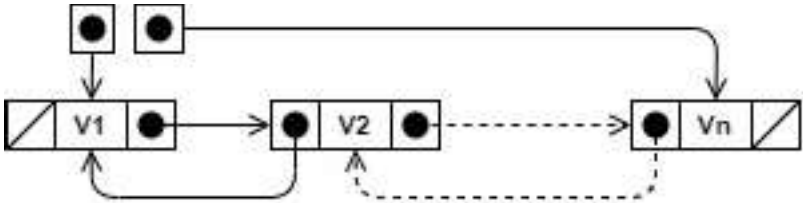
Liste chainée circulaire



Intérêts

- Ajout en queue sans parcours.
- Concaténation sans parcours.

Liste doublement chainées



Intérêts

- Parcours avant-arrière.
- Ajout en queue sans parcours.
- Suppression en queue sans parcours.
- Concaténation sans parcours.

Implémentation en Java

```
public class LinkedList<Type> {
        static class Maillon<Type>{
                public Maillon<Type> suivant;
                public Type valeur;
                public Maillon(Type valeur){
                        this.suivant=null;
                        this.valeur=valeur;
                }
        }
        public Maillon<Type> tete;
}
```

Type de données abstrait TDA

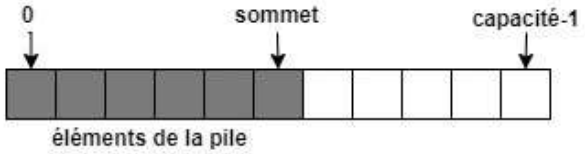
- Un type de données abstrait définit les propriétés d'une structure et son interface (contrat)
  - Pile : collection d'objets accessible selon une politique LIFO
  - File : collection d'objets accessible selon une politique FIFO
  - Liste : collection d'objets ordonnés accessible à partir de leur position.
  - Dictionnaire
  - Arbre
  - Graphe
  - ...
- Plusieurs implémentations sont possibles pour le même type de données abstrait.
- En POO les TDAs sont implémentés par des classes.
- La complexité des opérations dépend de l'implémentation.

Pile

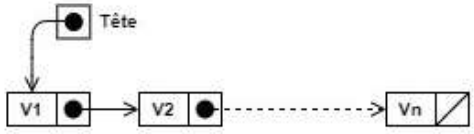
- Une pile est une structure de données où les insertions et les suppressions se font toujours du même côté (en haut).
- Interface :
  - empiler(pile,x) : Insérer l'élément x en haut de la pile.
  - dépiler(pile) : Supprimer la valeur sur le sommet de la pile.
  - estVide(pile) : Vérifier si la pile est vide.

Pile - mise en oeuvre

- Avec un tableau



- Avec une liste chainée

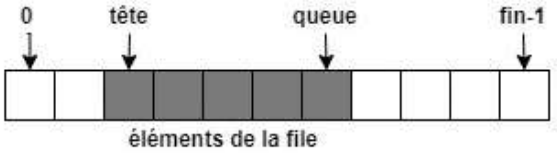


File

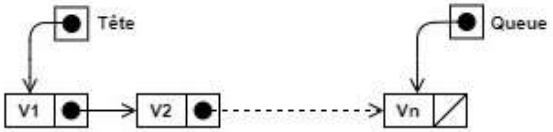
- Une file est une structure de données où les insertions se font d'un côté (queue) et les suppressions se font de l'autre côté (tête)
- Interface :
  - enfiler(file,x) : insérer l'élément x à la fin de la fille (queue)
  - defiler(file) : retirer l'élément à la tête de la file

File - mise en oeuvre

- Avec un tableau



- Avec une liste chainée (pointeurs de tête et queue)

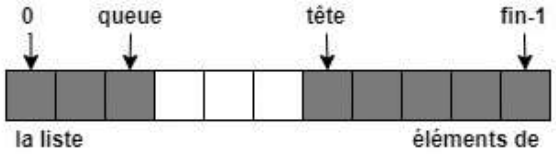


Liste

- Une suite finie d'éléments de même type accessibles selon leur position (relative)
- Une généralisation des piles et files.
- Interface :
  - insererDebut(liste,x) : insérer l'élément x au début de la liste.
  - insererFin(liste,x) : insérer l'élément x à la fin de la liste.
  - supprimerDebut(liste) : retirer l'élément au début de la liste.
  - supprimerFin(liste) : retirer l'élément à la fin de la liste.
  - inserer(liste,pos,x) : insérer l'élément x à la position pos.
  - supprimer(liste,pos) : supprimer l'élément à la position pos.
  - ...

Liste - mise en oeuvre

- Avec un tableau (peu pratique)



- Avec une liste doublement chainée (avec sentinelles)