

Révision Intra

1) Analyse et complexité des algorithmes:

- On classe les algorithmes selon leur complexité en temps dans le pire des cas, en moyenne ou dans le meilleur des cas
 - Dans le cours, on s'est intéressé à la **complexité dans le pire des cas**
- Analyse théorique
 - Pour calculer la complexité en temps d'un algorithme, on compte le **nombre d'opérations élémentaires** qu'on doit exécuter dans le pire des cas
 - On utilise la **notation asymptotique** pour exprimer la complexité d'un algorithme

Étant donné des fonctions $f(n)$ et $g(n)$, on dit que $f(n)$ est $O(g(n))$

s'il existe des constantes $c > 0$ et $n_0 \geq 1$ telles que $f(n) \leq cg(n) \quad \forall \text{entier } n \geq n_0$

2) Révision mathématiques:

- Méthodes de preuves
 - Par contradiction
 - Par induction
 - Directe et indirecte

- Sommations

- Par exemple, on a $\sum_{i=1}^n i = \frac{n(n+1)}{2}$ $\sum_{i=0}^h 2^i = 2^{h+1} - 1$

3) Fonctions récursives:

- Récursion: lorsqu'une fonction s'appelle elle-même
- Un exemple classique: La **fonction factorielle** $n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot (n-1) \cdot n$
- Définition récursive: $f(n) = \begin{cases} 1 & \text{si } n = 0 \\ n \cdot f(n-1) & \text{sinon} \end{cases}$
- Éléments essentiels:
 - Un ou plusieurs cas de base
 - Les appels récursifs doivent se terminer sur un cas de base
- Récursion linéaire, récursion binaire
- Complexité en temps d'un algorithme récursif:
 - Nombre d'appels récursifs X complexité cas de base

4) Piles:

● “Dernier arrivé, premier servi”

● Opérations principales”

- empiler(objet)
- dépiler()

● Opérations auxiliaires”

- haut()
- taille()
- estVide()

● ExceptionPileVide si on essaie de dépiler une pile vide

● Implémentations

■ Tableau de taille prédéfinie N

- ▲ ExceptionPilePleine liée a l'implémentation
- ▲ Complexité en espace $O(N)$
- ▲ Complexité en temps $O(1)$

■ Liste chaînée

- ▲ Pas de nouvelle exception liée a l'implémentation
- ▲ Complexité en espace $O(n)$
- ▲ Complexité en temps $O(1)$

5) Files:

● “Premier arrivé, premier servi”

● Opérations principales”

- ajouter(objet)
- enlever()

● Opérations auxiliaires”

- devant()
- taille()
- estVide()

● ExceptionFileVide si on essaie de d'enlever un élément et la file est vide

● Implémentations

■ Tableau circulaire de taille prédéfinie N

- ▲ ExceptionFilePleine liée a l'implémentation
- ▲ Complexité en espace $O(N)$
- ▲ Complexité en temps $O(1)$

■ Liste chaînée

- ▲ Pas de nouvelle exception liée a l'implémentation
- ▲ Complexité en espace $O(n)$
- ▲ Complexité en temps $O(1)$

6) Arbres et arbres binaires

● Définition et terminologie

● Profondeur d'un noeud et hauteur d'un arbre

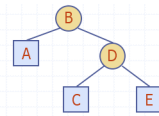
● Parcours d'arbres

- En profondeur: Parcours préfixe → B, A, D, C, E
Parcours suffixe → A, C, E, D, B
Parcours symétrique → A, B, C, D, E

- En largeur: Parcours hiérarchique → B, A, D, C, E

● Arbres binaires

- Définition, TAD arbres binaires
- Hauteur
- Code de Huffman



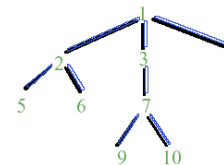
6) Arbres et arbres binaires (suite)

● Implémentations:

1) table des parents: on met dans la cellule i, le parent du noeud i

2) tableau: On met la valeur de la racine dans la cellule i. Si v est le fils gauche de u, on met sa valeur dans la cellule $2 * \text{index de } v$, si fils droit dans la cellule $2 * \text{index } v + 1$

3) structures chaînées



Nombre de cellule du tableau: $2^{h+1} \stackrel{\text{ici}}{=} 2^3$

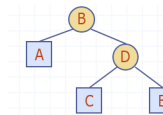
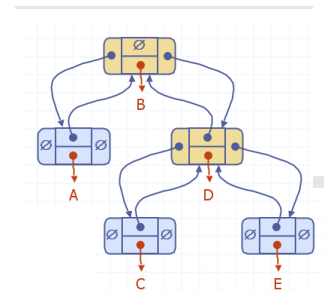


table des parents P

-	1	1	1	2	2	3	4	7	7	
0	1	2	3	4	5	6	7	8	9	10

∅	B	A	D	∅	∅	C	E
0	1	2	3	4	5	6	7



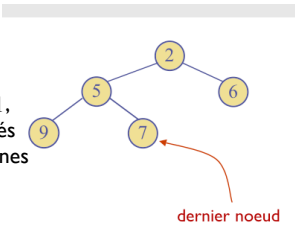
7) Files avec priorités

- Définition, TAD Files avec priorités
- Opérations principales:
 - insérer(k,v)
 - enleverMin()
- Relation d'ordre total sur les clés
- Implémentations:

■ Séquence ordonnée	■ Séquence non-ordonnée	■ Monceau
▲ insérer $O(n)$	▲ insérer $O(1)$	▲ insérer $O(\log n)$
▲ enleverMin $O(1)$	▲ enleverMin $O(n)$	▲ enleverMin $O(\log n)$
▲ Tri $O(n^2)$	▲ Tri $O(n^2)$	▲ Tri $O(n \log n)$
- Comparateur
- Trier avec une file avec priorités:
 - On insère les éléments à trier un à un dans la file
 - On tri avec une suite d'opération enleverMin()

8) Monceaux

- Un monceau est un arbre binaire ayant les deux propriétés suivantes:
 - **Ordre de monceau:** $clé(v) \leq clé(fil(v))$
 ↑
 Prioritaire
 - **Arbre binaire complet:** soit h la hauteur du monceau, alors
 - ▬ Pour $i = 0, \dots, h-1$, il y a 2^i noeuds de profondeur i
 - ▬ Au niveau (profondeur) $h-1$, les noeuds internes sont situés à la gauche des noeuds externes
- Hauteur d'un monceau: $O(\log n)$
- Insertion dans un monceau: $O(\log n)$
- Suppression dans un monceau: $O(\log n)$
- Construction de bas en haut: $O(n)$ au lieu de $O(n \log n)$

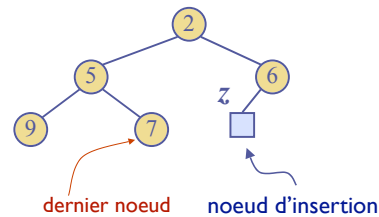


Insertion d'éléments dans un monceau

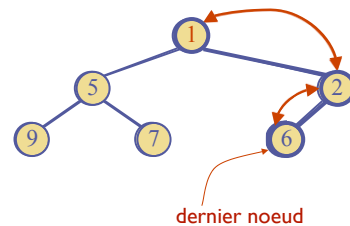
L'opération **insérer(k,x)** du TAD files avec priorités correspond à l'insertion d'un item **(k,x)** dans le monceau

- L'algorithme d'insertion consiste en trois étapes:

- 1) Trouver le noeud d'insertion z (qui deviendra le nouveau dernier noeud)
- 2) Insérer **(k,x)** dans le noeud z
- 3) Restorer l'ordre dans le monceau



© adapté de Goodrich, Tamassia, 2004

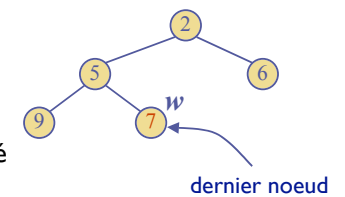


Suppression d'éléments dans un monceau

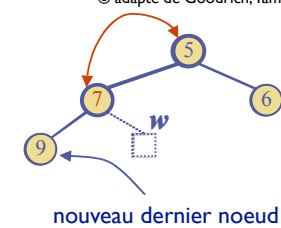
- L'opération **enleverMin()** du TAD files avec priorités correspond à la suppression de la racine d'un monceau

- L'algorithme de suppression est constitué de trois étapes:

- 1) Remplacer la clé de la racine par la clé du dernier noeud
- 2) Enlever le dernier noeud
- 3) Restorer l'ordre dans le monceau



© adapté de Goodrich, Tamassia, 2004



8) Monceaux: Implémentation

- On peut représenter un monceau contenant n clés par un vecteur de longueur $n+1$
- Pour le noeud au rang i du vecteur:
 - son fils gauche est au rang $2i$
 - son fils droit est au rang $2i + 1$
- La cellule de rang 0 n'est pas utilisée
- L'opération insérer(k,e) correspond à insérer au rang $n+1$
- L'opération enleverMin retourne l'élément au rang 1

