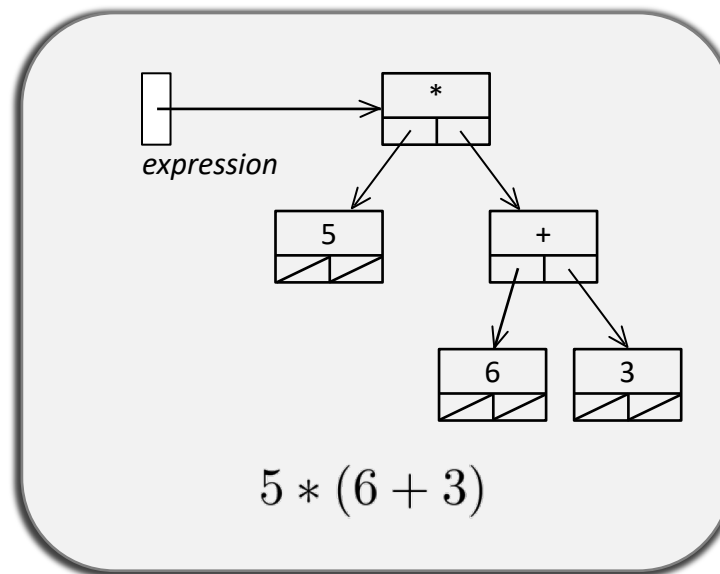




Algorithmique

Briques de base – Arbres



Introduction

Présentation

Parcours

Tas binaire

- Les listes chaînées sont adaptées à des contenus séquentiels
- Quand les liens entre les informations sont hiérarchiques ?
- Solution : arbre

Structure

Présentation

Parcours

Tas binaire

- Vocabulaire
 - On parle de relation de parenté
 - Cellules filles / cellule mère
 - Cellule qui a des filles : nœud
 - Cellule qui n'a pas de filles : feuille
- Nœud principal appelé racine
- Arbre binaire : 2 cellules filles

Propriétés

Présentation

Parcours

Tas binaire

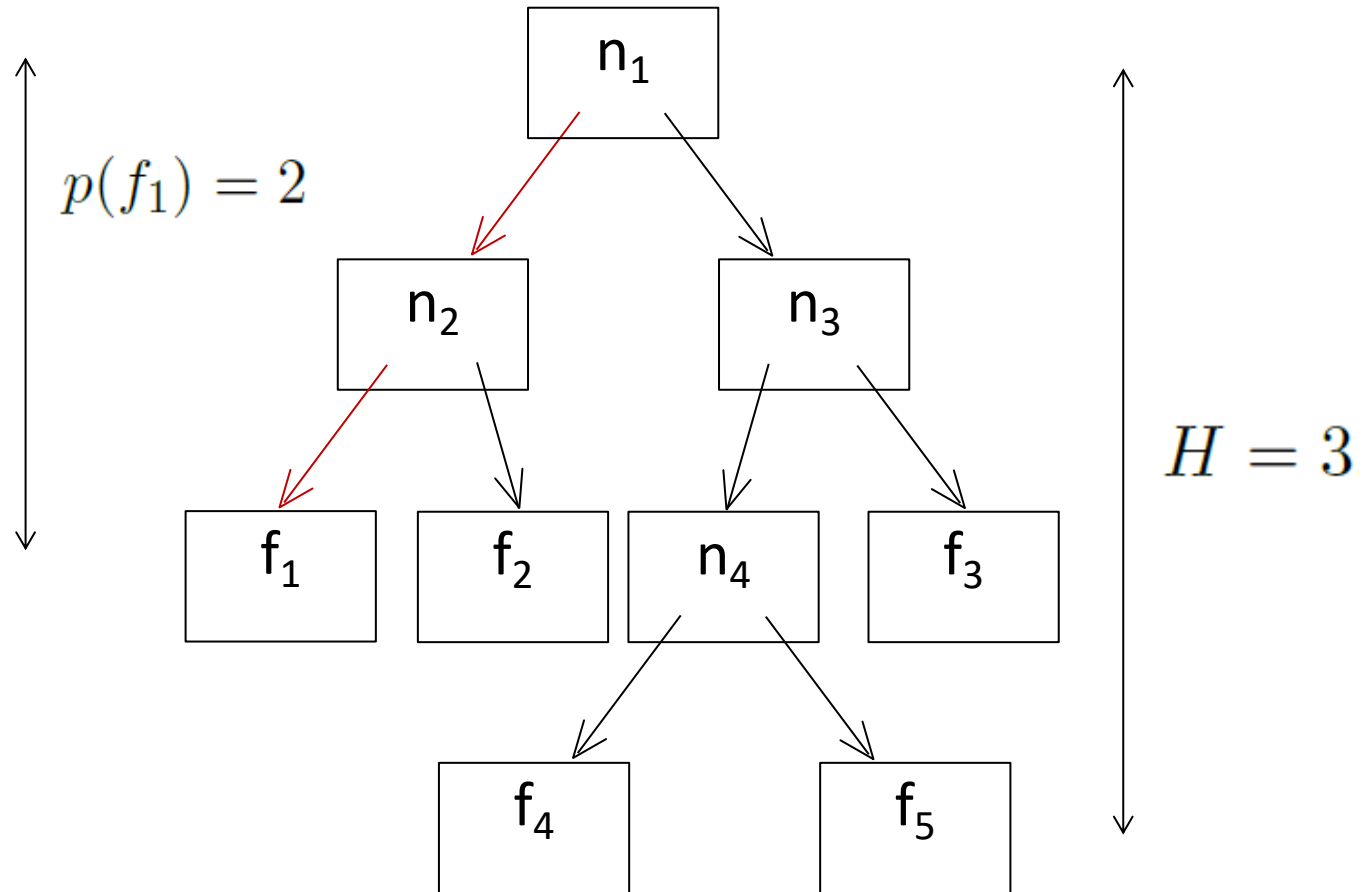
- Profondeur d'un nœud : nombre d'arrêtes qu'il faut parcourir pour atteindre la racine
- Hauteur d'un arbre : $\max(\text{profondeur})$
- Arbre équilibré : les profondeurs des sous-arbres gauche et droit ont une différence d'au plus 1
- Arbre parfait ou complet : profondeur des feuilles identique partout (=hauteur)

Propriétés

Présentation

Parcours

Tas binaire



Structure

Présentation

Parcours

Tas binaire

➤ Structure pour un arbre binaire

Structure *Arbre*

entier *valeur*

Arbre * *gauche*

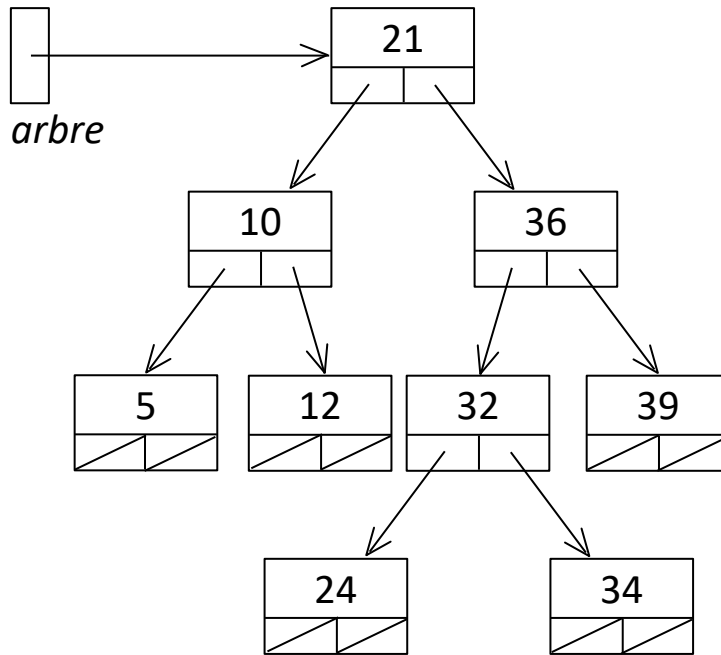
└─ Arbre * *droit*

Exemples

Présentation

Parcours

Tas binaire



- Arbre binaire de recherche
- Valeurs à gauche < valeur de la cellule mère
- Valeurs à droite > valeur de la cellule mère
- Propriété récursive

Exemples

Présentation

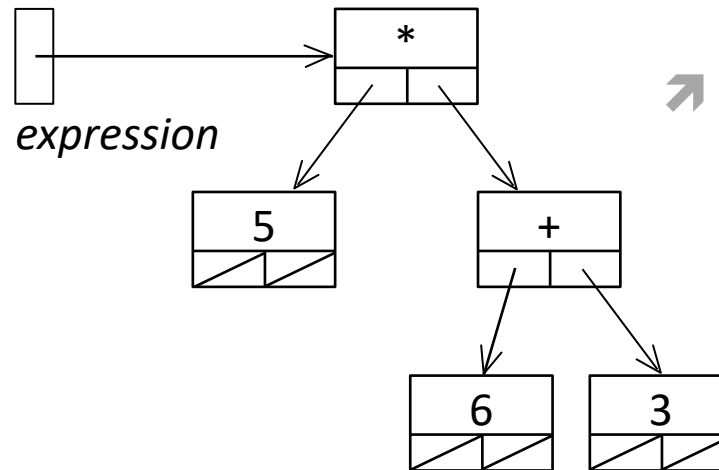
Parcours

Tas binaire

➤ Expression arithmétique

➤ Nœud : opérateur

➤ Feuilles : valeurs



$$5 * (6 + 3)$$

Adressage

Présentation

Parcours

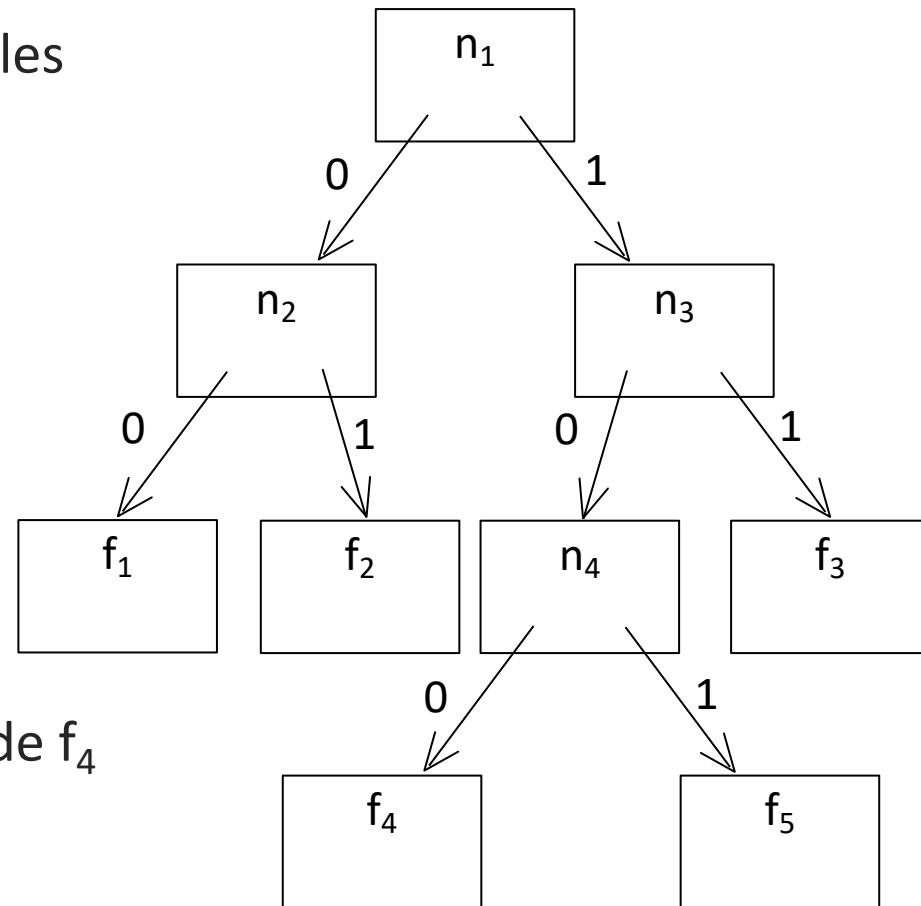
Tas binaire

➤ Manière de repérer les cellules dans l'arbre

➤ A gauche, 0

➤ A droite, 1

➤ Exemple : l'adresse de f_4 est 100



Aplatissement

Présentation

Parcours

Tas binaire

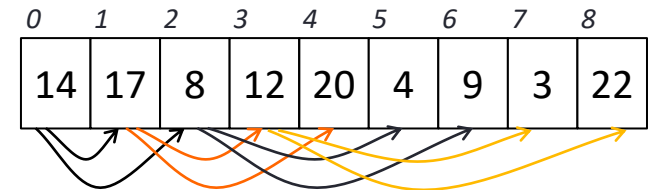
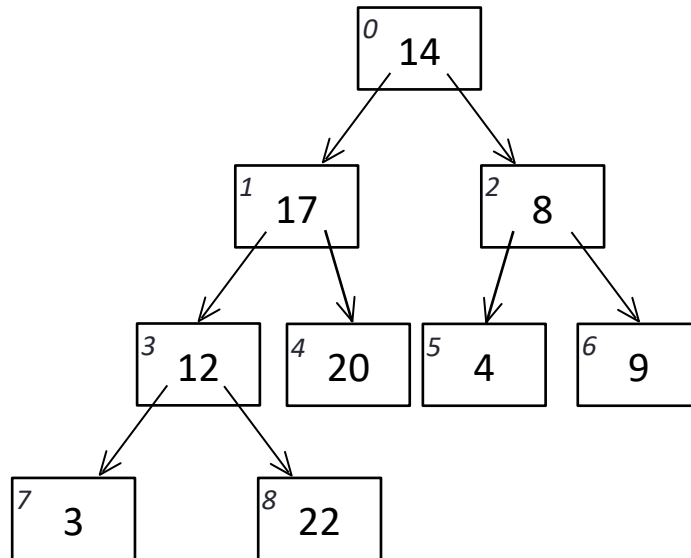
- L'arbre doit être quasi-complet pour une représentation optimale
- Seul le dernier niveau n'est pas complet
- Stockage dans un tableau avec
 - Indice de la racine 0
 - Indice du père $\lfloor (i - 1) / 2 \rfloor$
 - Indice du fils gauche $2i + 1$
 - Indice du fils droit $2i + 2$
- Si l'arbre n'est pas complet, alors il y a des trous dans le tableau

Aplatissement

Présentation

Parcours

Tas binaire



Parcours

Présentation

Parcours

Tas binaire

- Utilisation de fonctions récursives
- Permet un parcours en profondeur
- Suivant l'ordre dans lequel on effectue les opérations, le résultat est différent

Parcours

Présentation

Parcours

Tas binaire

- **Parcours préfixe**
 - On traite d'abord le nœud père puis le sous-arbre gauche, puis le droit
- **Parcours infixé**
 - On traite d'abord le sous-arbre gauche, puis le nœud père puis le sous-arbre droit
- **Parcours post-fixe**
 - On traite d'abord le sous-arbre gauche, puis le droit puis en dernier le nœud père

Parcours

Présentation

Parcours

Tas binaire

➤ Parcours préfixe

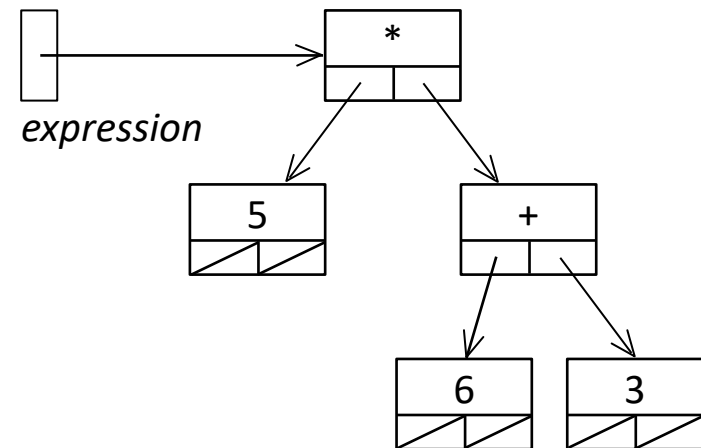
$* 5 + 6 3$

➤ Parcours infixe

$5 * 6 + 3$

➤ Parcours post-fixe

$5 6 3 + *$



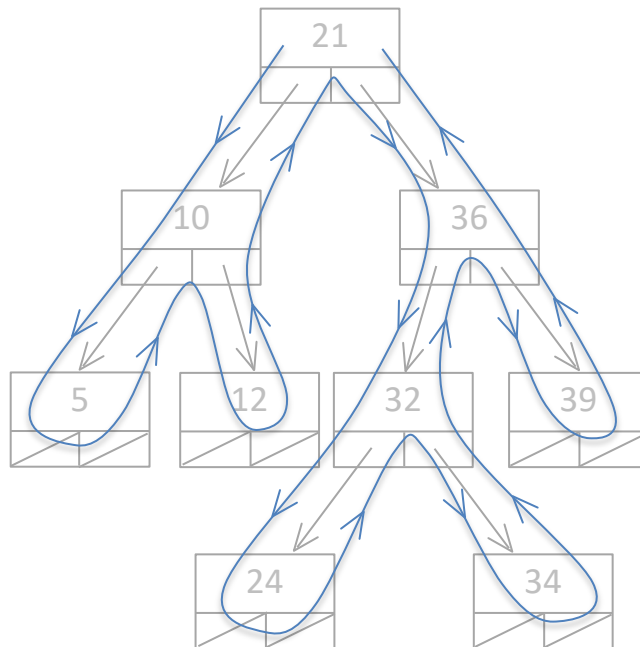
$5 * (6 + 3)$

Parcours

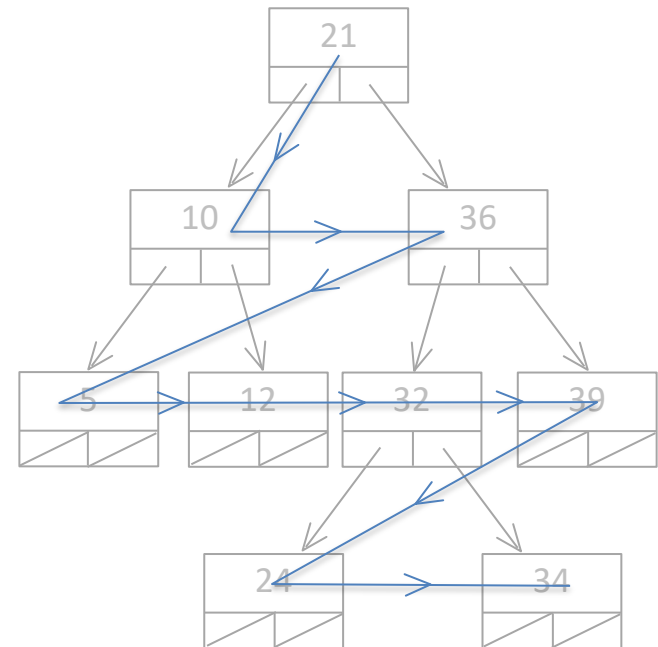
Présentation

Parcours

Tas binaire



Parcours en profondeur



Parcours en largeur

Parcours

Présentation

Parcours

Tas binaire

- Parcours en largeur
 - Algorithme itératif
 - Utilisation d'une structure de données annexe
- File
 - Principe FIFO
 - *First In First Out*
 - Le premier élément ajouté (enfilé) sera le premier élément à être retiré (défilé)

Parcours

Fonction `ParcoursArbreLargeur(racine)`

Entrée : Arbre * *racine*

Précondition : L'arbre est bien construit

Postcondition : Effectue l'opération `Operation()` sur chaque cellule de l'arbre dans l'ordre d'un parcours en largeur (par niveau)

Déclaration : Arbre * *courant*
File *file*

file.Enfiler(*racine*)

tant que *non file.Vide()* **faire**

courant \leftarrow *file*.Defiler()

`Operation(courant)`

si *courant* \rightarrow *gauche* $\neq \emptyset$ **alors**

file.Enfiler(*courant* \rightarrow *gauche*)

si *courant* \rightarrow *droit* $\neq \emptyset$ **alors**

file.Enfiler(*courant* \rightarrow *droit*)

Structure *Arbre*

entier *valeur*

Arbre * *gauche*

Arbre * *droit*

Exercices

Présentation

Parcours

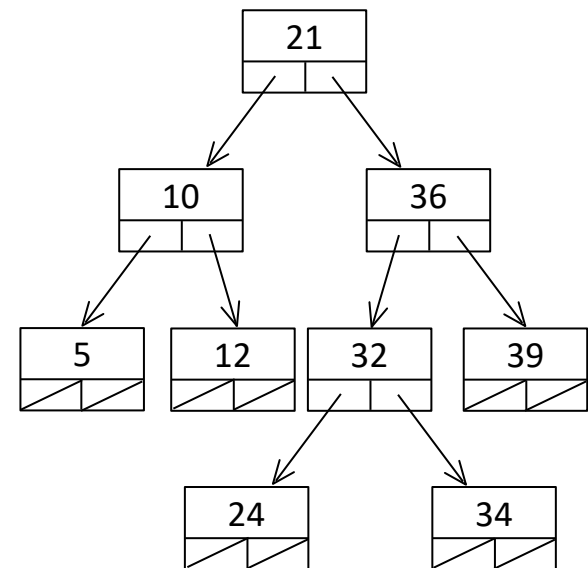
Tas binaire

➤ Dérouler l'algorithme sur l'arbre suivant

➤ Donner l'état de la file à chaque étape

```

file.Enfiler(racine)
tant que non file.Vide() faire
    courant ← file.Defiler()
    Operation(courant)
    si courant → gauche ≠ ∅ alors
        | file.Enfiler(courant → gauche)
    si courant → droit ≠ ∅ alors
        | file.Enfiler(courant → droit)
  
```



Exercices

Présentation

Parcours

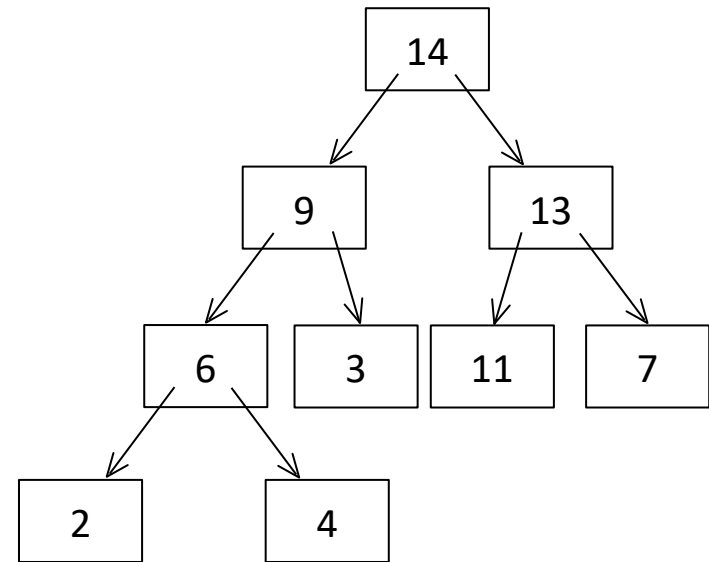
Tas binaire

- Taille maximum de la file ?
- Modifier l'algorithme pour qu'il fasse un parcours en profondeur

Tas binaire

Présentation
Parcours
Tas binaire

- Cas particulier d'un arbre binaire
 - Quasi-complet
 - Valeur père plus grande que la valeur des fils
- La plus grande valeur est donc la racine

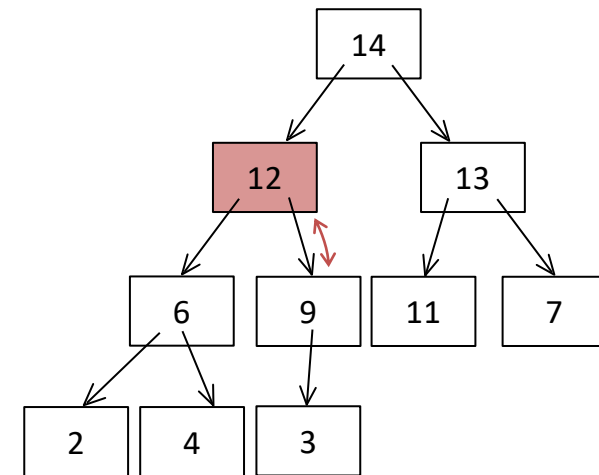
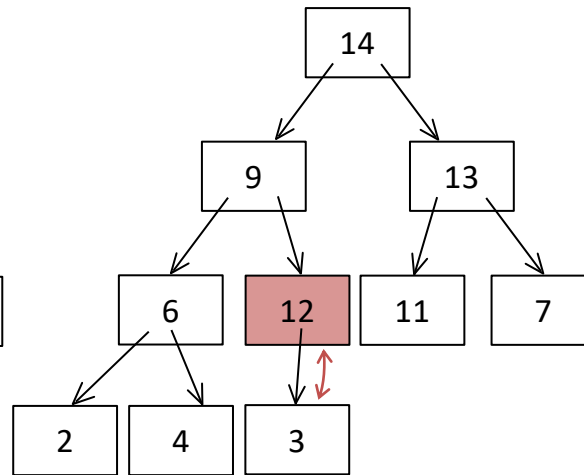
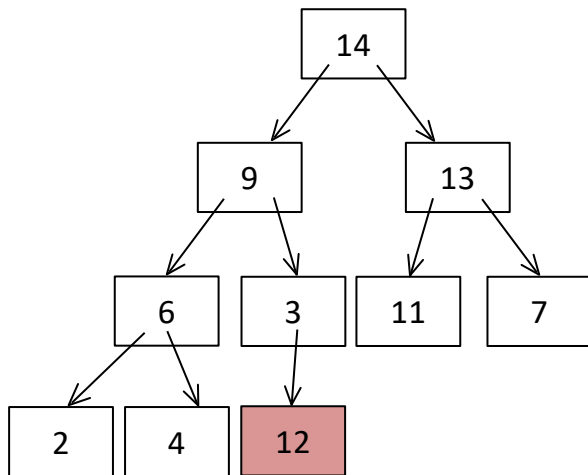


Tas binaire

Présentation
Parcours
Tas binaire

➤ Insertion

- Placement à la « fin » de l'arbre (première place libre)
- Permutations récursives jusqu'à la racine si besoin
 - Tant que la propriété n'est pas satisfaite



Tas binaire

Présentation

Parcours

Tas binaire

➤ Exercices

➤ Ecrire l'algorithme d'insertion

➤ Prouver qu'il est correct

➤ Il faut prouver que la propriété est bien respectée

➤ Bien penser aux pré et post-conditions

Tas binaire

Présentation
Parcours
Tas binaire

Structure *Tas*

entier *valeur*
Tas * *gauche*
Tas * *droite*
Tas * *pere*

Fonction *InsererTasArbre(cellule)*

Entrée/Sortie : Tas * *cellule*

Précondition : *cellule* contient un pointeur vers une cellule d'un arbre binaire bien formé dont la relation de tas binaire n'est en revanche pas respectée.

Postcondition : Les valeurs des cellules de l'arbre sont inversées (mais pas les pointeurs) récursivement jusqu'à la racine de l'arbre si nécessaire pour satisfaire la relation de tas binaire.

Déclaration : entier *tmp*

si *cellule* → *pere* ≠ ∅ **alors**

si *cellule* → *valeur* > *cellule* → *pere* → *valeur* **alors**

tmp ← *cellule* → *valeur*

cellule → *valeur* ← *cellule* → *pere* → *valeur*

cellule → *pere* → *valeur* ← *tmp* *InsererTasArbre*(*cellule* → *pere*)

Tas binaire

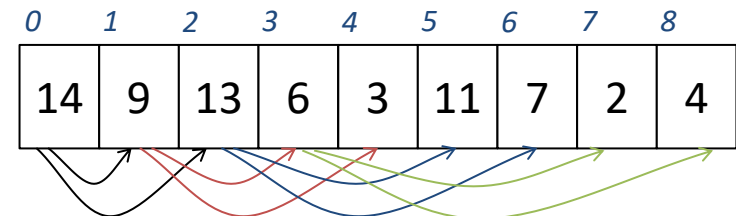
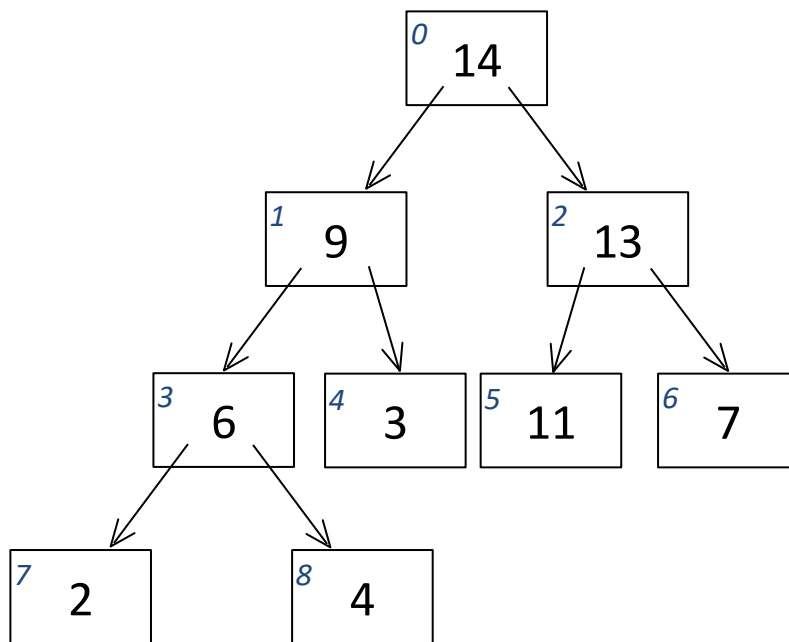
Présentation

Parcours

Tas binaire

➤ Avec un tableau (aplatissement)

➤ Très adapté !




Tas binaire

Présentation
Parcours
Tas binaire


➤ Insertion

0	1	2	3	4	5	6	7	8	9
14	9	13	6	3	11	7	2	4	12

0	1	2	3	4	5	6	7	8	9
14	9	13	6	12	11	7	2	4	3



0	1	2	3	4	5	6	7	8	9
14	12	13	6	9	11	7	2	4	3



Tas binaire

Présentation

Parcours

Tas binaire

Structure *Tas*

entier *taille*entier *taillemax*entier * *tab*

Fonction InsertionTasBinaire(*tas*, *valeur*)

Entrée/Sortie : Tas *tas*entier *valeur***Précondition** : Le tas est bien construit.

Postcondition : Effectue l'insertion de la valeur *valeur* dans le tas. La fonction renvoie *vrai* en cas de succès, *faux* en cas d'échec (taille max dépassée).

Déclaration : entier *indice*entier *pere*entier *tmp*

Tas binaire

Présentation
Parcours
Tas binaire

si $tas.taille = tas.taillemax$ **alors**

retourne *faux*

$tas.tab[tas.taille] \leftarrow valeur$

$indice \leftarrow tas.taille$

$tas.taille \leftarrow tas.taille + 1$

$pere \leftarrow \text{ArrondiInf}((indice - 1)/2)$

tant que $indice > 0$ **et** $tas.tab[pere] < tas.tab[indice]$ **faire**

$tmp \leftarrow tas.tab[pere]$

$tas.tab[pere] \leftarrow tas.tab[indice]$

$tas.tab[indice] \leftarrow tmp$

$indice \leftarrow pere$

$pere \leftarrow \text{ArrondiInf}((indice - 1)/2)$

retourne *vrai*

Tas binaire

Présentation

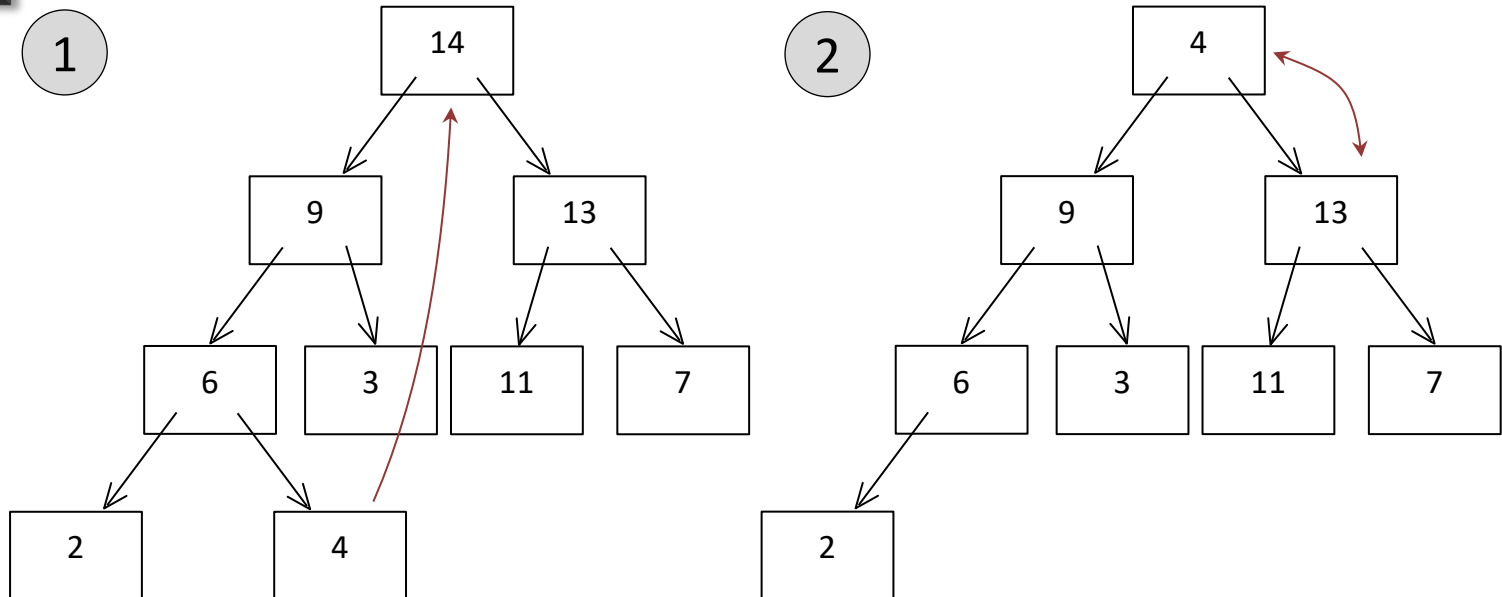
Parcours

Tas binaire

- Suppression
- On commence par permuter l'élément à supprimer avec le dernier
- Ensuite, récursivement, on permute l'élément avec le fils le plus grand lorsque la propriété n'est pas respectée

Tas binaire

Présentation
Parcours
Tas binaire



Tas binaire

Présentation
Parcours
Tas binaire

