Module EA4 – Éléments d'Algorithmique II

Outils pour l'analyse des algorithmes

Dominique Poulalhon
dominique.poulalhon@irif.fr

Université Paris Diderot
L2 Informatique & Math-Info
Année universitaire 2019-2020

Arbres Binaires de Recherche

IV. Hauteur moyenne

On a obtenu pour le moment deux types de résultats sur les ABR. D'abord :

Théorème

la liste triée des éléments d'un ABR de taille n peut être obtenue en temps $\Theta(n)$ par un parcours en profondeur infixe.

C'est un résultat de complexité *dans tous les cas*, et il est tout à fait satisfaisant : on ne peut pas espérer produire une liste de longueur n en moins que $\Theta(n)$ opérations, donc cette méthode est optimale.

En revanche, les opérations de recherche et de modifications ne donnent pas une complexité optimale dans tous les cas :

Théorème

la recherche, l'ajout et la suppression d'un élément dans un ABR de hauteur h se font en temps $\Theta(h)$ au pire.

Ce résultat fait intervenir la hauteur de l'ABR, qui n'est pas une constante pour une taille n donnée. Pour comprendre si ce résultat est intéressant, il faut étudier comment la hauteur d'un ABR est reliée à sa taille – dans le meilleur cas, dans le pire cas, et surtout, en moyenne.

Hauteur d'un ABR

Nous avons déjà vu que :

la hauteur $h(A)$ d'un arbre binaire A à n sommets vérifie :

$$\log n \leqslant h(A) \leqslant n - 1$$

et ces bornes sont atteintes :

- la borne inférieure, par les arbres binaires *parfaits*, c'est-à-dire dont toutes les feuilles sont au même niveau (ce qui n'est possible que si $n + 1$ est une puissance de 2), ou, pour n quelconque, par les arbres *quasi-parfaits*, dont toutes les feuilles sont sur les deux derniers niveaux
- la borne supérieure, par les arbres *filiformes*, qui n'ont qu'une feuille et des nœuds à un seul enfant ; il ne s'agit en fait pas vraiment d'arbres binaires, mais de listes chaînées un peu tordues...

Le premier cas est très satisfaisant : la recherche ressemble à une recherche dichotomique, les modifications se font avec le même coût... ces ABR ont les avantages du tableau trié sans les inconvénients, c'est parfait.
Le deuxième, c'est la cata totale : une structure (assez) compliquée, qui marche aussi mal qu'une liste chaînée non triée.
La question qui se pose alors naturellement est :

À quoi ressemble un ABR « typique » ? À un arbre binaire parfait ? À un arbre filiforme ? Ou ni l'un ni l'autre ?

Hauteur moyenne des ABR

À quoi ressemble un ABR « typique » ? À un arbre binaire parfait ? À un arbre filiforme ? Ou ni l'un ni l'autre ?

Une première (non-)réponse vient de l'observation suivante :

chaque arbre binaire à n sommets admet un unique étiquetage par $\{1, \dots, n\}$ respectant les contraintes d'un ABR

(forcément, puisque le parcours infixe d'un ABR doit parcourir ses sommets dans l'ordre de leurs clés)

On pourrait donc se dire qu'il « suffit » d'étudier les arbres binaires, et non les ABR. Ce qui est en fait difficile, et donne le résultat suivant :

Théorème (admis)

la hauteur moyenne d'un arbre binaire choisi uniformément parmi les arbres binaires à n nœuds est en $\Theta(\sqrt{n})$.

Bref, presqu'une catastrophe : l'arbre binaire moyen n'est, heureusement, pas filiforme, mais il n'est pas du tout parfait non plus.

Théorème (admis)

la hauteur moyenne d'un arbre binaire choisi uniformément parmi les arbres binaires à n nœuds est en $\Theta(\sqrt{n})$.

Mais la distribution uniforme est-elle pertinente ? cette distribution de probabilité correspond-elle à la manière dont les ABR sont obtenus ?

A priori non : on ne construit pas un ABR en choisissant sa forme au hasard, puis en le remplissant – et si on choisissait la forme d'abord, tant qu'à faire on le choisirait (quasi-)parfait, pas au hasard.

Non, un ABR est construit par une succession d'insertions et de suppressions, et ça n'a aucune raison de donner la distribution uniforme sur les formes d'arbres binaires.

Il faudrait donc construire un modèle probabiliste tenant compte à la fois des insertions et des suppressions – mais, d'une part, c'est difficile, et d'autre part un tel modèle dépendrait fortement de l'ensemble considéré. Par exemple, dans une bibliothèque, il y a *a priori* peu de suppressions (vols, pertes) une bibliothèque de n livres a donc certainement été obtenue après au plus $2n$ achats et n pertes.
À l'inverse, chaque année le nombre d'étudiants qui quittent une université est à peu près égal au nombre de nouveaux inscrits, donc le nombre total d'insertions et de suppressions dans l'ensemble n'a aucune raison d'être en $\Theta(n)$.

Hauteur moyenne des ABR

Bref, considérer les insertions *et les suppressions*, c'est trop compliqué. On va se contenter des insertions. Cela donne un modèle probabiliste à la fois représentatif d'une manière de construire des ABR, et suffisamment simple pour qu'on puisse l'étudier.

On considère donc, non pas la distribution uniforme sur les formes d'arbres binaires, mais la distribution uniforme sur les manières de créer un ABR de taille n par insertions successives de n éléments donnés – les entiers de 1 à n pour se fixer les idées. C'est cela qu'on appelle la *distibution de probabilité des ABR*

Et on va obtenir :

Théorème

la hauteur moyenne d'un ABR construit par l'insertion des entiers $1, \dots, n$ *dans un ordre aléatoire choisi uniformément est en* $\Theta(\log n)$.

OUF !!!

Suite du cours :

- comprendre, sur quelques exemples, pourquoi cette hauteur moyenne est très inférieure à celle des arbres binaires (selon la distribution uniforme)
- prouver ce théorème... c'est un peu dur, mais faisable
- remarquer que cela prouve au passage la complexité moyenne de QuickSort.

Hauteur moyenne des ABR

On note :

- $\mathcal{B}_n$ l'ensemble des arbres binaires de taille n,
- $\mathcal{A}(\sigma)$ l'ABR obtenu par insertion successive de $\sigma(1), \sigma(2), \ldots$
- $h(a)$ la hauteur d'un arbre a.

On veut comparer deux notions de hauteur moyenne des arbres binaires de taille n :

- selon la distribution uniforme : $\dfrac{1}{\#\mathcal{B}_n} \displaystyle\sum_{a \in \mathcal{B}_n} h(a)$
- et selon la distibution de probabilité des ABR :

$$\frac{1}{\#\mathfrak{S}_n} \sum_{\sigma \in \mathfrak{S}_n} h(\mathcal{A}(\sigma)) \;=\; \frac{1}{\#\mathfrak{S}_n} \sum_{a \in \mathcal{B}_n} \#\{\sigma \in \mathfrak{S}_n \text{ tq } \mathcal{A}(\sigma) = a\} \cdot h(a)$$

D'où l'importance de $\#\{\sigma \in \mathfrak{S}_n \text{ tq } \mathcal{A}(\sigma) = a\}$: le nombre de permutations différentes telles que l'insertion successive de $\sigma(1), \sigma(2), \ldots$ aboutisse à l'arbre a

Intuition avec les mains : les arbres filiformes, ou plus généralement de hauteur $\Theta(n)$, peuvent être obtenus de beaucoup moins de manières différentes que les arbres parfaits, ou plus généralement les arbres de hauteur $\Theta(\log n)$. La contribution des arbres filiformes est donc considérablement plus faible dans le cas de la distribution des ABR que dans le cas de la distribution uniforme.

Combien de permutations produisent le même ABR ?

Cas filiforme : c'est le cas le plus simple : puisque les insertions sont faites aux feuilles, les éléments ont nécessairement été insérés dans l'ordre de la seule branche : d'abord l'élément à la racine, puis son (unique) fils (gauche ou droit), puis son (unique) petit-fils... et pour finir l'élément dans la seule feuille de l'arbre. Donc :

> *Chacun des 2^{n-1} arbres filiformes de taille n compte seulement pour un poids 1 dans le calcul de la hauteur moyenne des ABR.*

Plus généralement, on a le résultat suivant, que j'espère assez clair :

Lemme

Soit a un ABR, de racine r, et de sous-arbres g et d. Si $a = \mathcal{A}(\sigma)$, alors :

- *r a été inséré en premier : $\sigma(1) = r$*
- *soit γ l'ordre d'insertion des éléments de g (dans a) ; alors $\mathcal{A}(\gamma) = g$*
- *soit δ l'ordre d'insertion des éléments de d (dans a) ; alors $\mathcal{A}(\delta) = d$*
- *γ et δ s'intercalent de manière quelconque dans σ*

Par exemple : pour $\sigma = 3\ 1\ 2\ 4$, on a $r = 3$, $\gamma = 1\ 2$ et $\delta = 4$.
Les seules permutations qui produisent le même ABR sont $3\ 1\ 4\ 2$ et $3\ 4\ 1\ 2$.

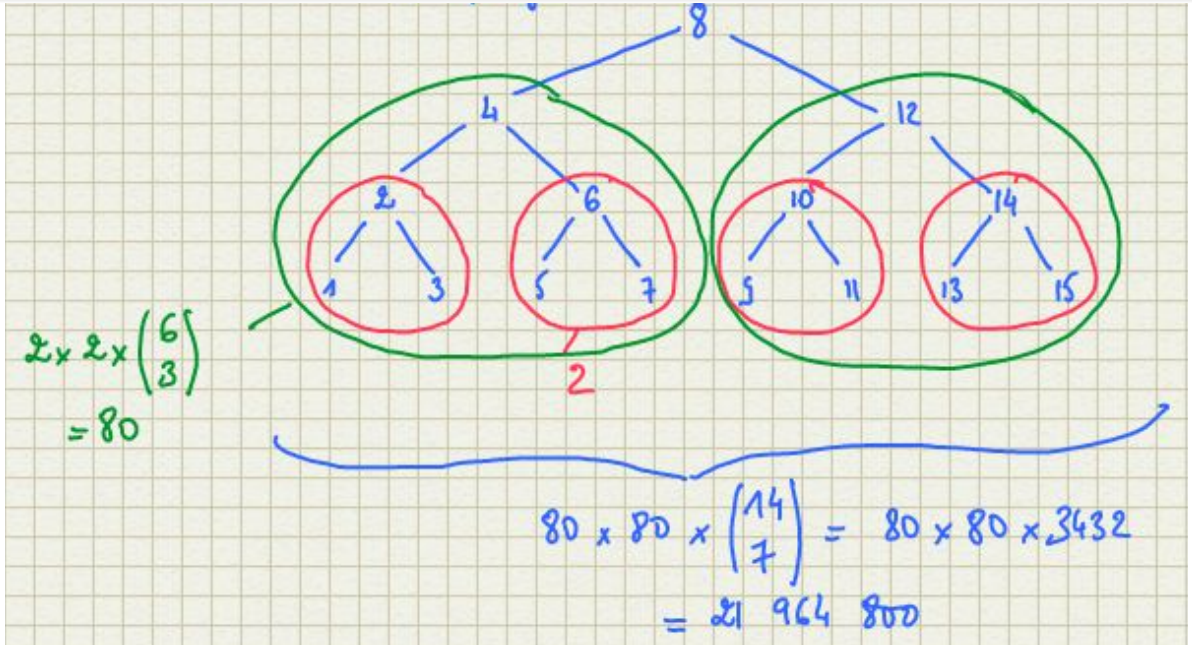
Combien de permutations produisent le même ABR ?

Ce lemme permet de calculer de manière récursive le nombre de permutations qui produisent un arbre donné : calculer le nombre de possibilités pour g, pour d, puis le nombre de manières d'intercaler les éléments de g et ceux de d, autrement dit de colorier en vert et en rouge le bon nombre de positions dans σ (sauf la première, réservée pour r).

D'où un facteur multiplicatif $\binom{\#g + \#d}{\#g} = \frac{(\#g + \#d)!}{\#g!\ \#d!}$.

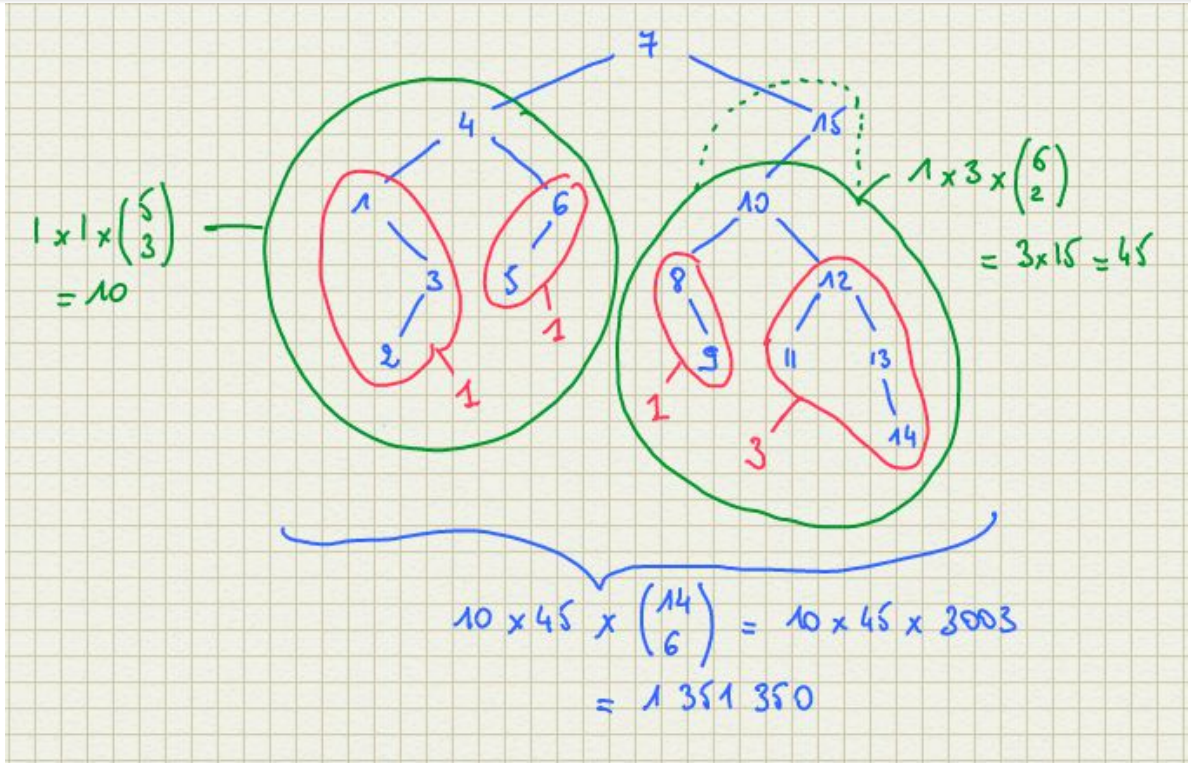
Noter que ce nombre est (beaucoup) plus grand quand #g et #d sont du même ordre de grandeur que lorsque le partage est déséquilibré.

Exemple : cas de l'ABR parfait de taille 15



Combien de permutations produisent le même ABR ?

Un cas un peu moins équilibré



Moralité : la contribution des arbres est d'autant plus importante qu'ils sont équilibrés, et la différence d'ordre de grandeur est énorme entre les arbres relativement équilibrés et les arbres dégénérés comme les arbres filiformes.

Il est donc moralement raisonnable que la hauteur moyenne des ABR soit nettement plus faible que celle des arbres binaires uniformes… mais ce n'est pas une preuve !

Hauteur moyenne des ABR (démonstration)

Pour étudier la hauteur moyenne des ABR, nous disposons essentiellement de la définition de la hauteur :

$h(a)$ = hauteur de l'arbre a ; si g et d sont les sous-arbres de a :

$$h(a) = 1 + \max(h(g), h(d))$$

Malheureusement, la seule majoration raisonnable de cette égalité est $h(a) \leqslant 1 + h(g) + h(d)$, trop grossière pour espérer montrer mieux que $h(a) \leqslant n$...

On introduit une grandeur plus sensible aux petites modifications en passant à l'exponentielle. Cette grandeur représente en quelque sorte la « capacité » d'un arbre de même hauteur :

Soit $H(a) = 2^{h(a)}$. On obtient alors une majoration plus fine :

$$H(a) = 2\max(H(g), H(d)) \leqslant 2(H(g) + H(d))$$

Soit $\overline{h}(n) = \dfrac{1}{n!} \displaystyle\sum_{\sigma \in \mathfrak{S}_n} h(\mathcal{A}(\sigma))$ et $\overline{H}(n) = \dfrac{1}{n!} \displaystyle\sum_{\sigma \in \mathfrak{S}_n} H(\mathcal{A}(\sigma))$

But : montrer que $\overline{h}(n) \in \Theta(\log n)$

Hauteur moyenne d'un ABR (démonstration)

par convexité de l'exponentielle, $\overline{H}(n) \geqslant 2^{\overline{h}(n)}$, donc $\overline{h}(n) \leqslant \log \overline{H}(n)$

$\Longrightarrow$ il suffit de montrer que $\overline{H}(n)$ est polynomiale (de degré quelconque, pas forcément linéaire : on a gagné une vraie marge de manœuvre en passant de h à H)

on découpe $\mathfrak{S}_n$ en n parties : $\mathfrak{S}_{n,i} = \{\sigma \in \mathfrak{S}_n \mid \sigma(1) = i\}$

$\mathfrak{S}_{n,i}$ est donc l'ensemble des σ t.q. $\mathcal{A}(\sigma)$ a i à la racine, un sous-arbre gauche de taille $i-1$, et un sous-arbre droit de taille $n-i$. Comme, pour chacun des arbres obtenus, $H(a) \leqslant 2(H(g) + H(d))$, on obtient en faisant la moyenne sur $\mathfrak{S}_{n,i}$:

$$\overline{H}(n,i) \leqslant 2\left[\overline{H}(i-1) + \overline{H}(n-i)\right]$$

pour tout i, $\mathfrak{S}_{n,i}$ a cardinal $(n-1)!$, donc $\overline{H}(n) = \frac{1}{n}\sum_{i=1}^{n} \overline{H}(n,i)$

donc finalement : $$\overline{H}(n) \leqslant \frac{2}{n}\sum_{i=1}^{n} \overline{H}(i-1) + \overline{H}(n-i) = \frac{4}{n}\sum_{i=0}^{n-1} \overline{H}(i)$$

on peut alors vérifier par récurrence que $\frac{1}{4}\binom{n+3}{3}$ (qui est un polynôme de degré 3) est une majoration de $\overline{H}(n)$. □

Corollaire

les opérations de recherche, d'insertion et de suppression dans les ABR ont une complexité (en temps) $\Theta(\log n)$ *(dans le pire cas de chaque ABR, en moyenne sur tous les ABR, considérés selon la distribution précédente)*

Corollaire

la construction d'un ABR par insertion successive de n *valeurs a une complexité* $\Theta(n \log n)$ *en moyenne*

Corollaire

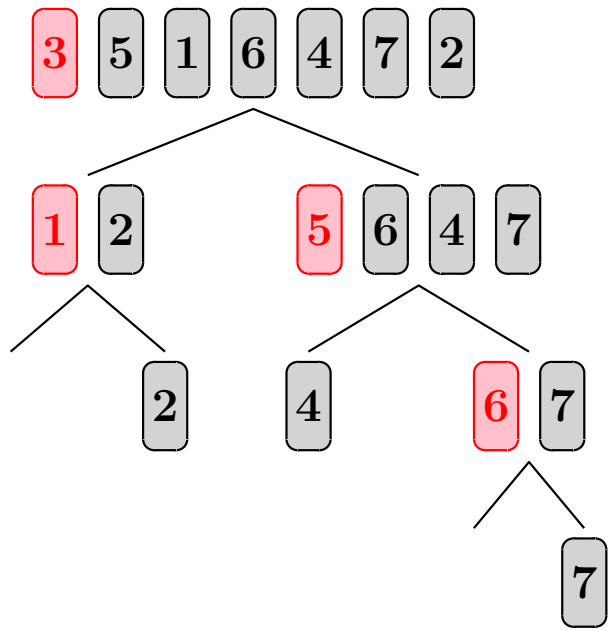
trier une liste par construction d'un ABR a une complexité $\Theta(n \log n)$ *en moyenne*

(en deux étapes : construction de l'ABR puis parcours infixe)

Lemme

l'arbre de récursion de QuickSort, étiqueté par les pivots, est précisément l'ABR obtenu par insertion successive des pivots

Exemple : `T = [3, 5, 1, 6, 4, 7, 2]`



+ le cumul des complexités par niveau est en $O(n)$, donc :

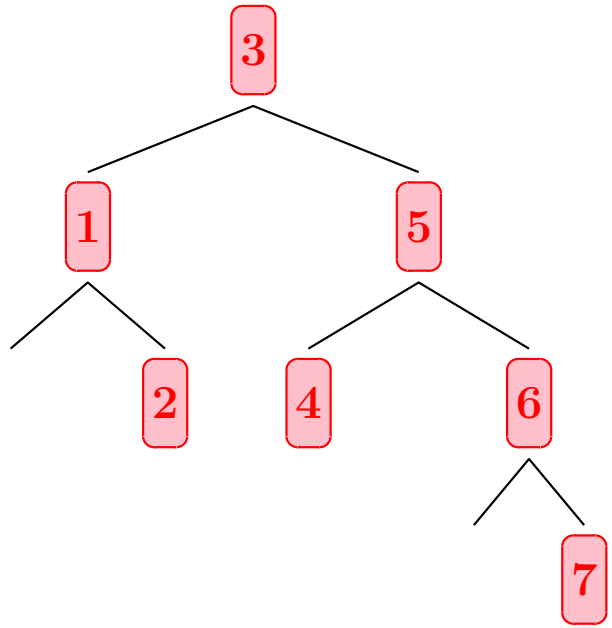
Corollaire

la complexité de QuickSort est également $\Theta(n \log n)$ ***en moyenne***

Lemme

l'arbre de récursion de QuickSort, étiqueté par les pivots, est précisément l'ABR obtenu par insertion successive des pivots

Exemple : `T = [3, 5, 1, 6, 4, 7, 2]`



+ le cumul des complexités par niveau est en $O(n)$, donc :

Corollaire

la complexité de QuickSort est également $\Theta(n \log n)$ ***en moyenne***

Comparaison ABR / liste

On peut donc compléter le tableau comparatif des complexités, avec une victoire nette des ABR sur les différents types de listes pour le traitement des ensembles dynamiques :

	tableau		liste chaînée		ABR
	non trié	trié	non triée	triée	
recherche	$\Theta(n)$	$\Theta(\log n)$	$\Theta(n)$	$\Theta(n)$	$\Theta(h)$
insertion	$+\,\Theta(1)$	$\Theta(n)$	$+\,\Theta(1)$	$+\,\Theta(1)$	$\Theta(h)$
suppression	$\Theta(n)$	$\Theta(n)$	$+\,\Theta(1)$	$+\,\Theta(1)$	$\Theta(h)$
minimum	$\Theta(n)$	$\Theta(1)$	$\Theta(n)$	$\Theta(1)$	$\Theta(h)$
sélection	$\Theta(n)$	$\Theta(1)$	$\Theta(n)$	$\Theta(k)$	$\Theta(\log n)$
union	$\Theta(n^2)$	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n)$	$\Theta(n)$

(pour l'union : c'est faisable en commençant par un (des) parcours infixes ; d'autres méthodes existent sans construire de listes inutiles, mais c'est plus compliqué)
(pour la sélection : pour obtenir une complexité de $\Theta(\log n)$, il faut enrichir la structure, par exemple en stockant dans chaque nœud la taille du sous-arbre correspondant ; cette information peut être conservée sans surcoût lors des insertions et suppressions)

Comparaison ABR / liste

On peut donc compléter le tableau comparatif des complexités, avec une victoire nette des ABR sur les différents types de listes pour le traitement des ensembles dynamiques :

	tableau		liste chaînée		ABR
	non trié	trié	non triée	triée	(en moyenne)
recherche	$\Theta(n)$	$\Theta(\log n)$	$\Theta(n)$	$\Theta(n)$	$\Theta(\log n)$
insertion	$+\,\Theta(1)$	$\Theta(n)$	$+\,\Theta(1)$	$+\,\Theta(1)$	$\Theta(\log n)$
suppression	$\Theta(n)$	$\Theta(n)$	$+\,\Theta(1)$	$+\,\Theta(1)$	$\Theta(\log n)$
minimum	$\Theta(n)$	$\Theta(1)$	$\Theta(n)$	$\Theta(1)$	$\Theta(\log n)$
sélection	$\Theta(n)$	$\Theta(1)$	$\Theta(n)$	$\Theta(k)$	$\Theta(\log n)$
union	$\Theta(n^2)$	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n)$	$\Theta(n)$

(pour l'union : c'est faisable en commençant par un (des) parcours infixes ; d'autres méthodes existent sans construire de listes inutiles, mais c'est plus compliqué)
(pour la sélection : pour obtenir une complexité de $\Theta(\log n)$, il faut enrichir la structure, par exemple en stockant dans chaque nœud la taille du sous-arbre correspondant ; cette information peut être conservée sans surcoût lors des insertions et suppressions)