



LSINF1121: Intro 2

Les arbres de recherches

Pierre Schaus

Les tables de symboles

- Type abstrait de données permettant:
 - D'**insérer** une valeur avec une clef 
 - Etant donné la clef  de **retrouver** la valeur correspondante

Quelques exemples

Application	Objectif de la recherche	clef	Valeur
Web search	Trouver des pages intéressantes	Mot-clefs	Liste de pages
DNS inversé	Trouver l'adresse IP	Nom de domaine	Adresse IP
Système de fichier	Trouver un fichier	Nom du fichier	Localisation sur le disque
Compilateur	Trouver les propriétés d'une variable	Nom de la variable	Type et valeur
Table de routage	Trouver un chemin de routage pour un paquet	Destination	Interface de sortie du routeur (meilleure route)

Un tableau peut être vu comme une table de symboles

- Les clefs sont les indices
- Les valeurs sont les entrées du tableau
- Python adopte cette convention.

Convention du livre

- Une clef ne peut se retrouver qu'une seule fois dans la table de symbole.
- Les clefs sont différentes de null et les valeurs également.
 - En effet get(key) retourne null si la clef n'est pas présente
- Etant donné que l'égalité des clefs est basée sur la méthode “equals”, il est plus prudent que les clefs soient des objets immuables.

Rappel sur l'égalité et equals

- Propriété demandées:
 - Réflexivité: $x.equals(x)$ est vrai
 - Symétrique: $x.equals(y) \Leftrightarrow y.equals(x)$
 - Transitivité: $x.equals(y)$ et $y.equals(z) \Rightarrow x.equals(z)$
 - Non null: $x.equals(y)$ est faux

Exemple

```
public final class Date implements Comparable<Date>
{
    private final int month;
    private final int day;
    private final int year;

    public boolean equals(Object y)
    {
        if (y == this) return true;
        if (y == null) return false;
        if (y.getClass() != this.getClass())
            return false;
        Date that = (Date) y;
        if (this.day != that.day) return false;
        if (this.month != that.month) return false;
        if (this.year != that.year) return false;
        return true;
    }
}
```

Contrat de base

La classe est final donc la question ne se pose pas

Le cast ne peut pas échouer

On teste tous les champs.

!!! Ici ce sont des types primitifs !!!

Sinon il aurait fallu utiliser equals et si c'était un tableau,
il faudrait tester l'égalité sur chaque élément du tableau
(deepEqual)

Implémentation possible des tables de symboles

- Implémentations possibles pour une tables de symboles:
 - Sequential Search = simple liste chainée qui contient les clefs et les valeurs
 - * insert(key,value) en $O(n)$,
 - * get(key) / delete(key) en $O(n)$
- En partie 4 nous verrons une autre structure (table de hashage) qui permet $O(1)$ amortie pour toutes les opérations. Patience ...

Relation d'ordre entre les clefs

- Il existe souvent une relation d'ordre entre les clefs. On parle alors de généralement de dictionnaire

```
public class ST<Key extends Comparable<Key>, Value>
```

...

Key min()

smallest key

Key max()

largest key

Key floor(Key key)

largest key less than or equal to key

Key ceiling(Key key)

smallest key greater than or equal to key

int rank(Key key)

number of keys less than key

Key select(int k)

key of rank k

void deleteMin()

delete smallest key

void deleteMax()

delete largest key

int size(Key lo, Key hi)

number of keys between lo and hi

Iterable<Key> keys()

all keys, in sorted order

Iterable<Key> keys(Key lo, Key hi)

keys between lo and hi, in sorted order

Utilisation de l'ordre

- Pour améliorer les complexités grâce à la recherche binaire (= dichotomique)

	sequential search	binary search
search	N	$\log N$
insert / delete	N	N
min / max	N	1
floor / ceiling	N	$\log N$
rank	N	$\log N$
select	N	1
ordered iteration	$N \log N$	N

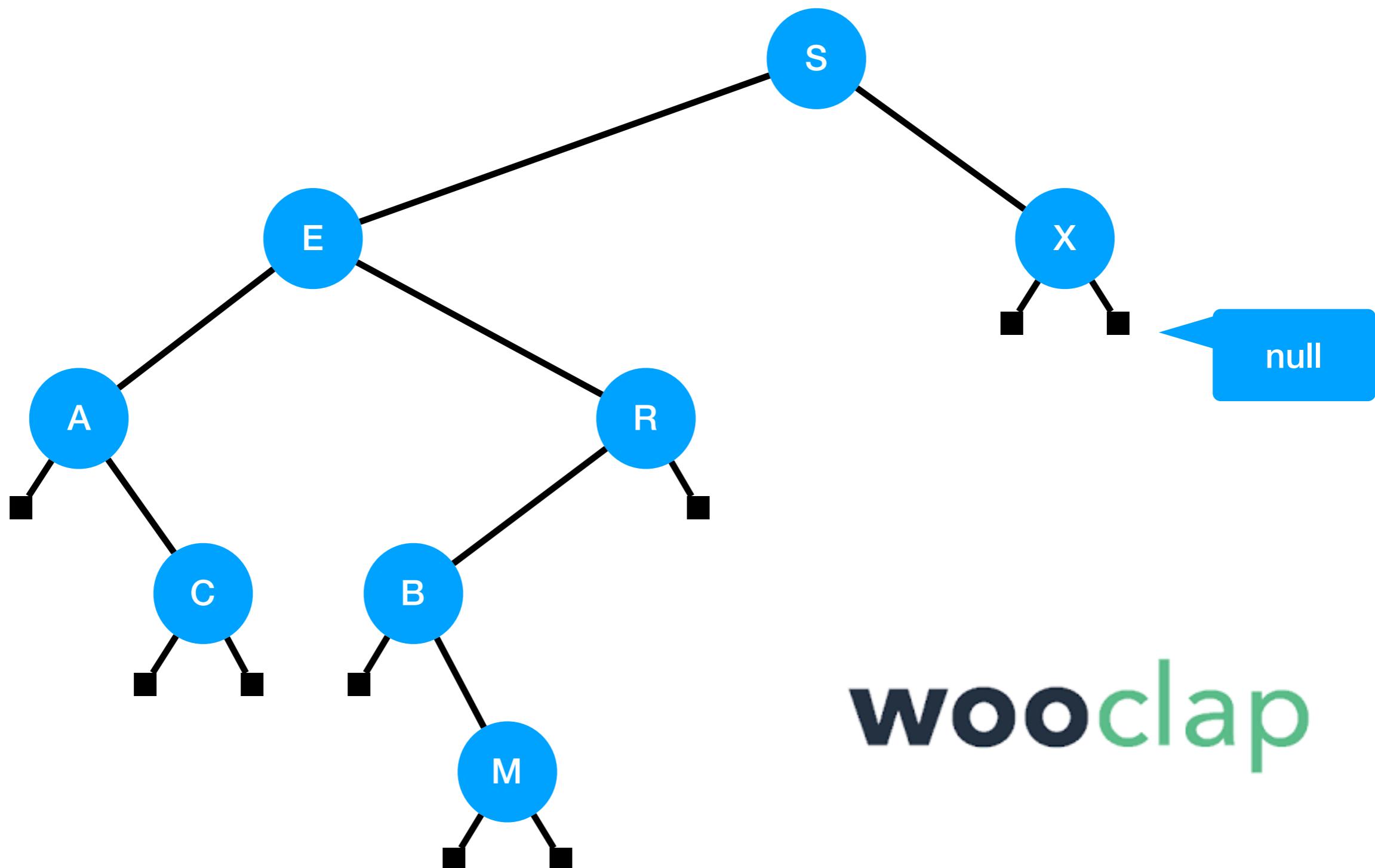
On peut encore améliorer grâce à une nouvelle structure de données: Les arbres de recherche

Les arbres de recherche

- ▶ Arbre binaire:
 - * Vide
 - * Non-vide: Il contient deux arbres binaire (gauche et droit)
- ▶ Arbre binaire de recherche = arbre binaire avec une clef-valeur dans chaque noeud telle que
 - * Cette clef est plus grande que toutes les clefs dans le sous arbre de gauche
 - * Cette clef est plus petite que toutes les clefs dans le sous arbre de droite

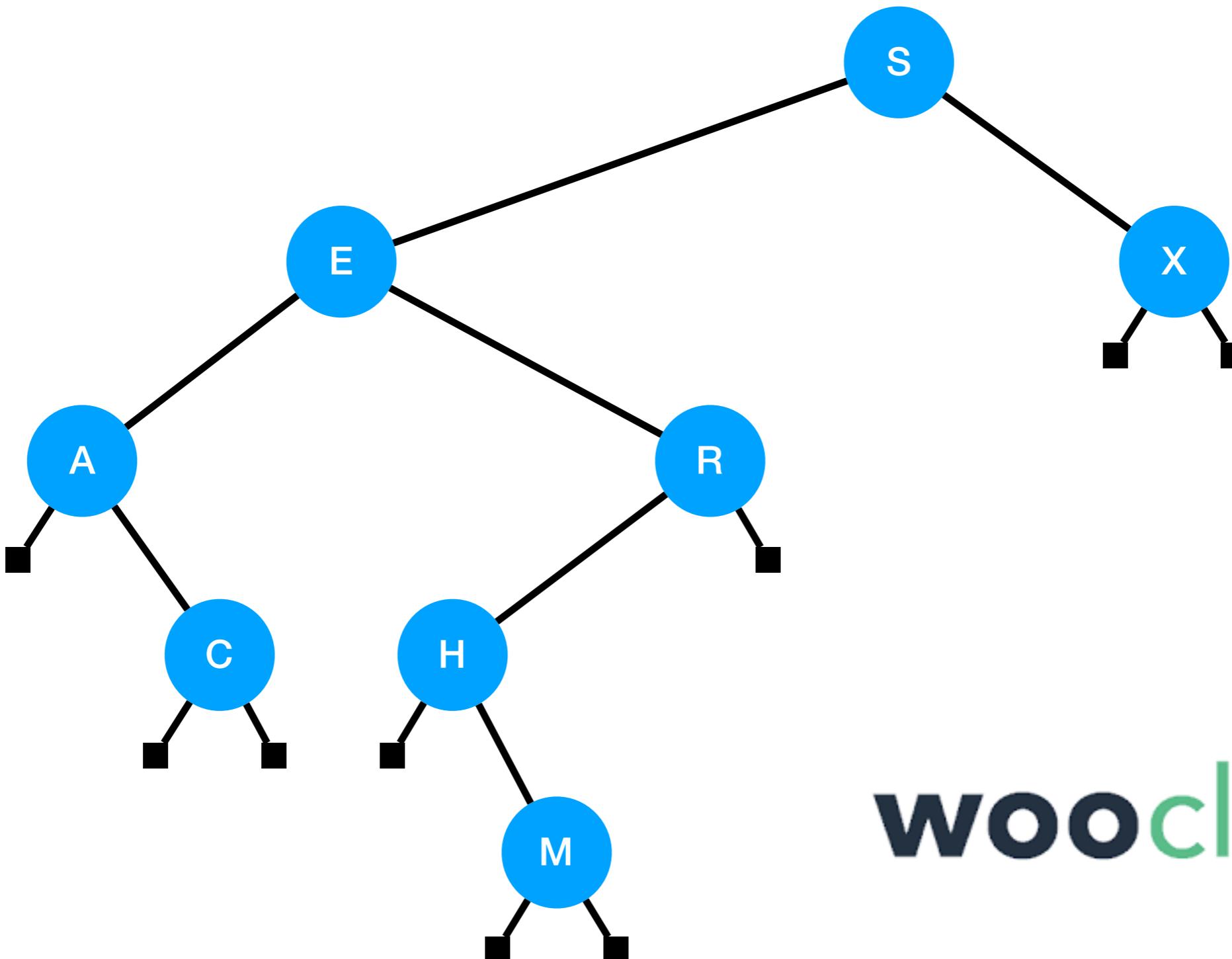
Exemple

- Arbre de recherche non valide ?



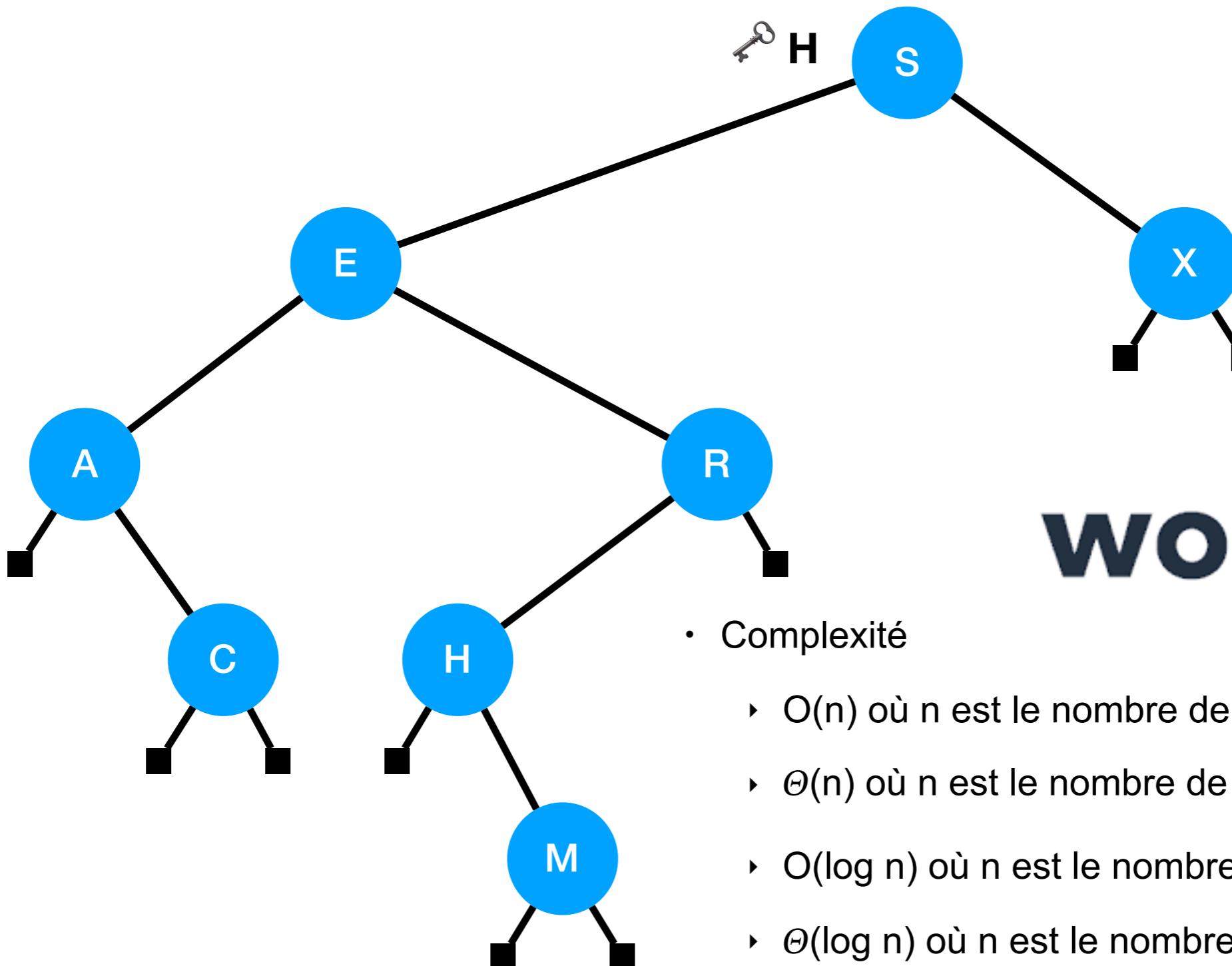
Exemple

- Arbre de recherche valide ?



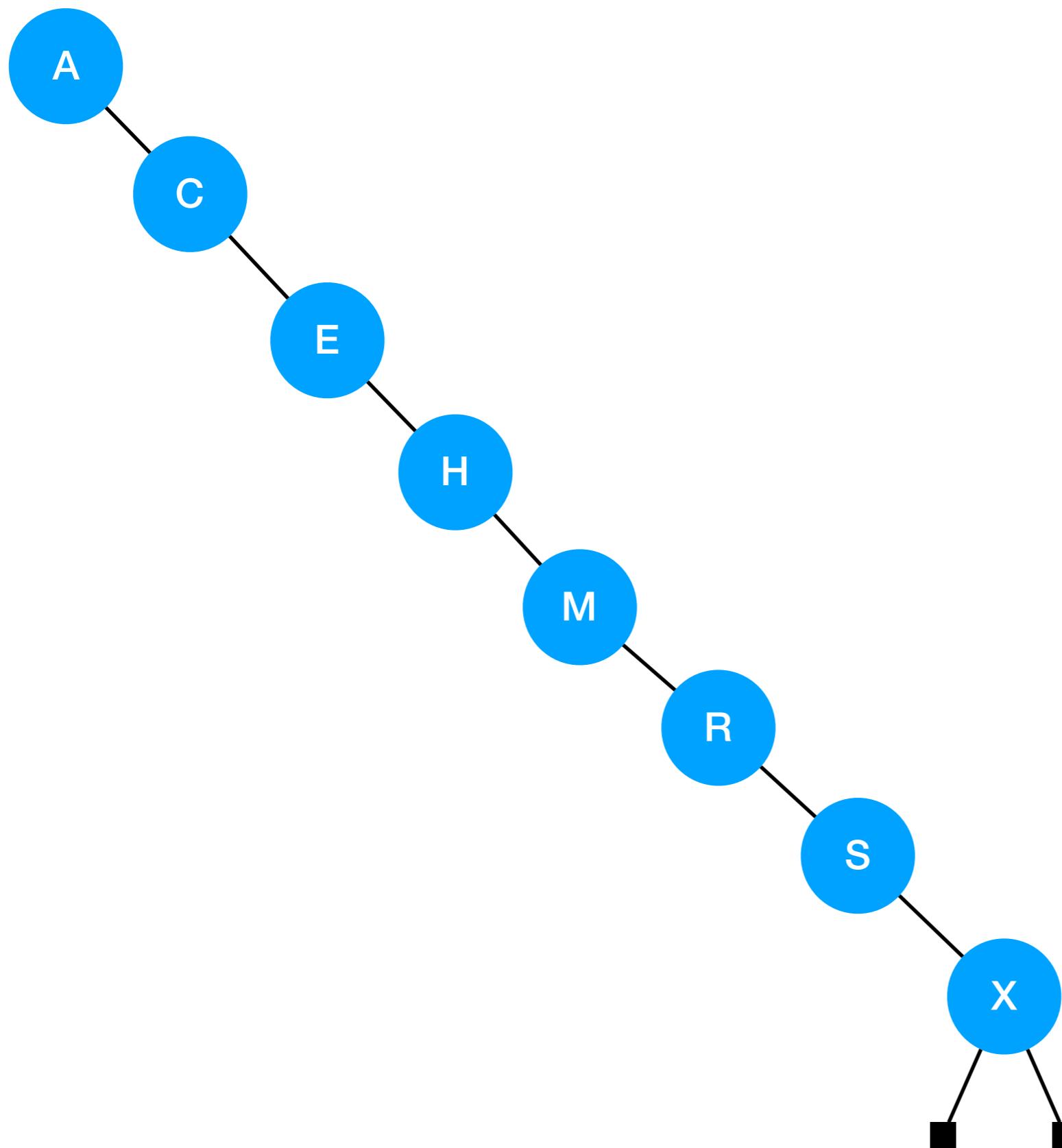
woodlap

Recherche dans un arbre de recherche



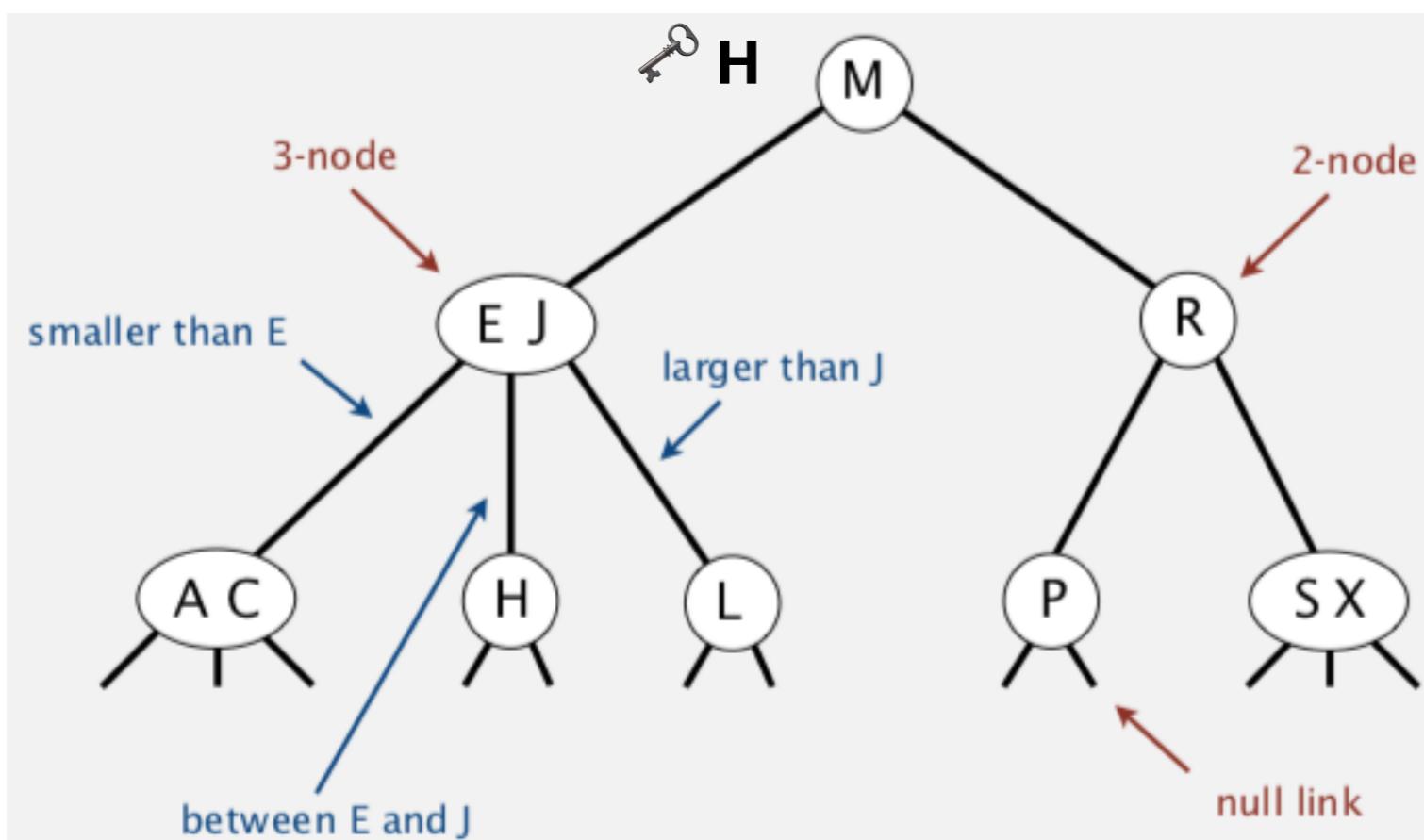
wooclap

Le pire cas pour la hauteur



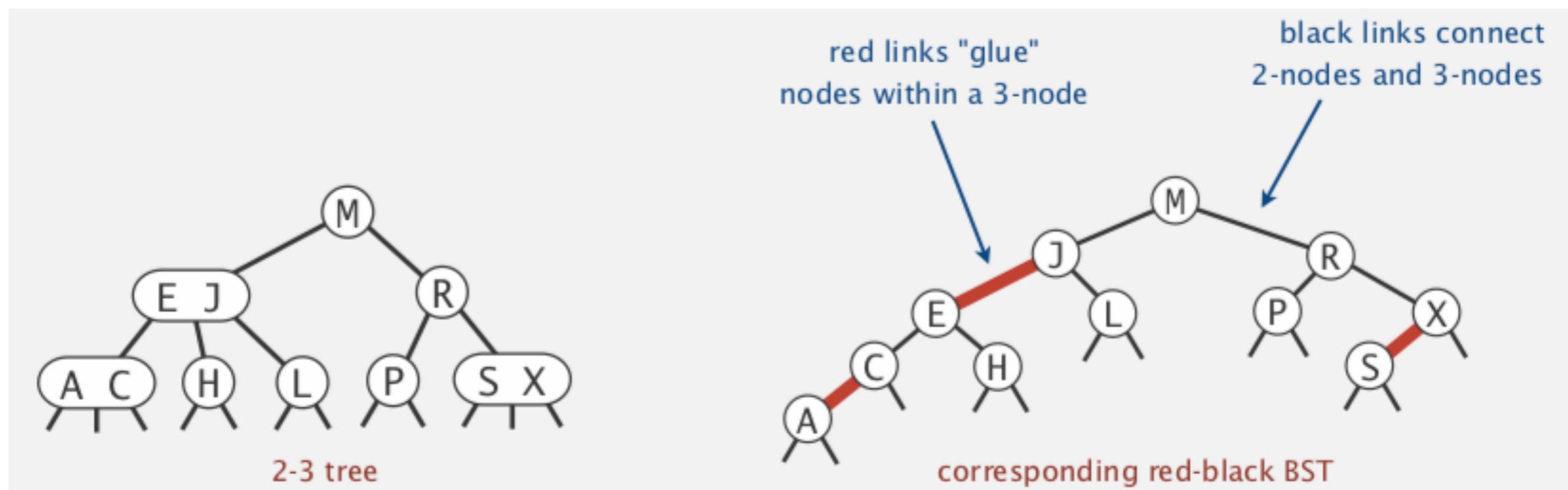
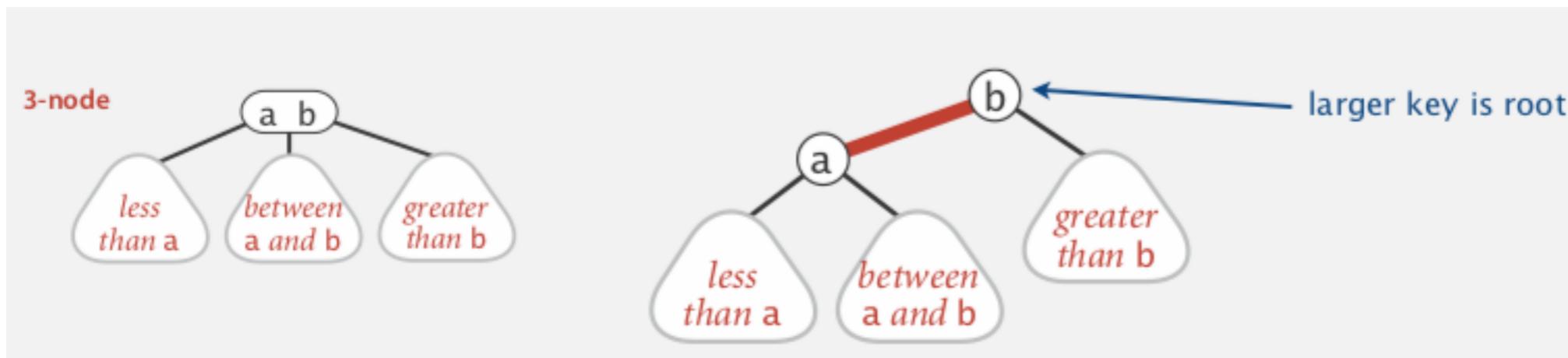
Arbre équilibré O(h) = O(log(n))

- Arbres 2-3, 1 ou 2 clefs par noeud
 - ▶ Equilibre parfait: Chaque chemin depuis la racine vers une feuille (lien null) a exactement la même longueur.



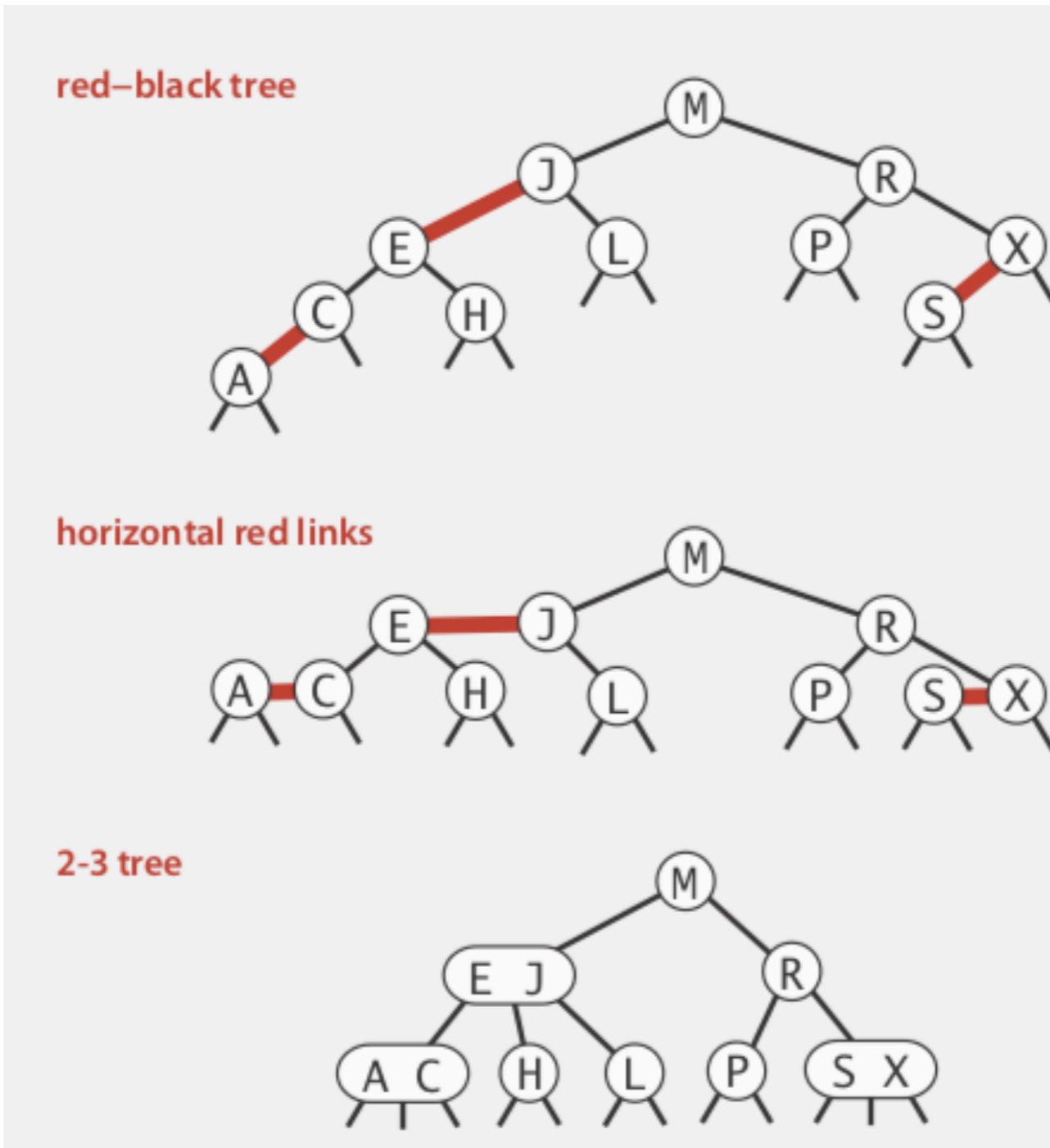
Arbre équilibré binaire

- Arbre red-black penchant à droite
 - ▶ = representation d'un arbre 2-3 mais avec un arbre binaire.



Red-black vs 2-3

- La correspondance est parfaite
 - Il suffit de mettre les liens rouge à l'horizontal



Résumé des complexité

implementation	guarantee		
	search	insert	delete
sequential search (unordered list)	N	N	N
binary search (ordered array)	$\lg N$	N	N
BST	N	N	N
2-3 tree	$c \lg N$	$c \lg N$	$c \lg N$
red-black BST	$2 \lg N$	$2 \lg N$	$2 \lg N$

Dans les deux semaines qui viennent

- Lecture approfondie des chapitres 3.1,
- Exercices théoriques pour maîtriser les arbres de recherche et les arbres de recherches équilibrés
- Exercices d'implémentation
- S6: Mid term test, ne compte dans la note finale que pour 2 points et uniquement s'il fait remonter la note.
 - Inginious le lundi de 16h à 18h
 - Individuellement: toute tentative de tricherie ou detection de plagiat sera sanctionnée d'un zero pour le cours.

