LSINF1121 ALGORITHMIQUE ET STRUCTURES DE DONNÉES

TP3: ARBRES DE RECHERCHE

QUESTION 3.1.1 SEQUENTIAL VS BINARY SEARCH

	Sequential	Binary
Insertion	O(1) amorti	O(n) O(n/2) moyen
Recherche	O(n)	O(log n)

QUESTION 3.1.1 SEQUENTIAL VS BINARY SEARCH

Essayons de d'abord faire les puts, puis les gets

$$\text{seq} = P + PG$$

$$\text{bin} = (\sum_{i=1}^{P} i) + G \log P = \frac{1}{2} P(P+1) + G \log P$$

QUESTION 3.1.1 SEQUENTIAL VS BINARY SEARCH

Essayons de d'abord faire les puts, puis les gets

$$\text{seq} = P + PG \qquad \text{bin} = \frac{P(P+1)}{2} + G \log P$$

$$\text{bin} < \text{seq}$$

$$\frac{P(P+1)}{2} + G \log P < P + PG$$

$$G > \frac{P^2 - P}{2(P - \log P)} \sim \frac{P}{2}$$

RAPPELS SUR LES BSTS

Propriété d'un Binary Search Tree:

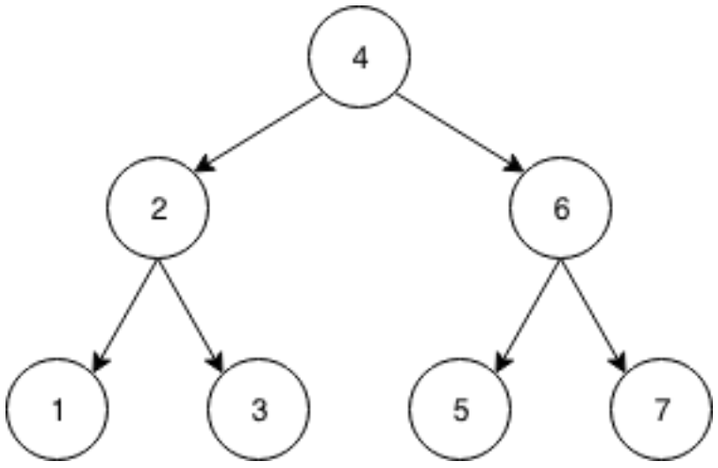
- C'est un arbre:
  - C'est un graphe
  - Connexe
  - Possédant n-1 arêtes pour n noeuds
  - Pas de cycles
  - Au plus un chemin entre toute paire de noeuds
- Il est binaire (deux enfants par noeuds au maximum)
- Propriété sur les clés:
  - Les noeuds en dessous, et à gauche d'un noeud donné doivent avoir des clés plus petites
  - Les noeuds en dessous, et à droite d'un noeud donné doivent avoir des clés plus grandes

RAPPELS SUR LES BSTS

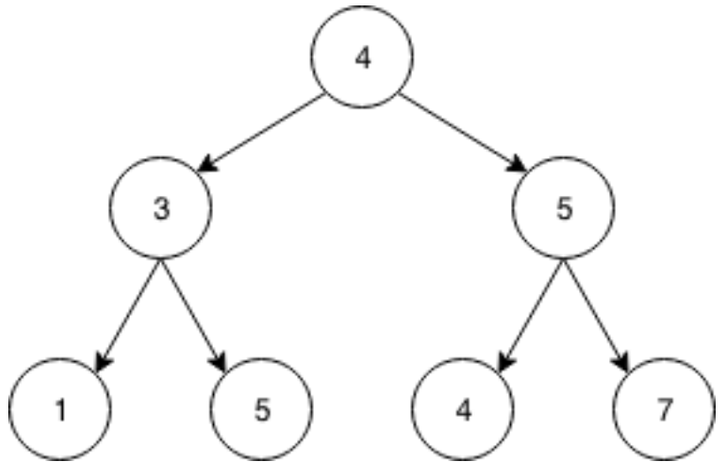
Autre propriété: équilibre. Un arbre est équilibré si la taille de tout chemin entre la racine et n'importe quelle feuille est de même taille (+/- 1 au maximum)

Un arbre binaire est parfaitement équilibré si la taille de ces chemins est de log_2(n). (+/- 1). Cela correspond a utiliser tout les enfants de tout les noeuds: tout les niveaux, sauf le dernier, sont remplis.

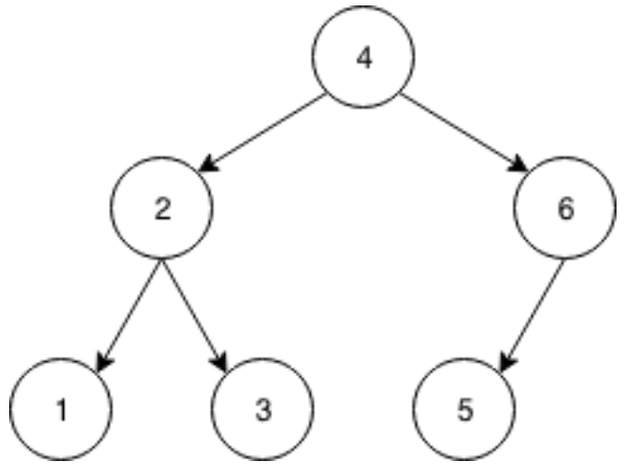
QUESTION 3.1.5 ISBST()



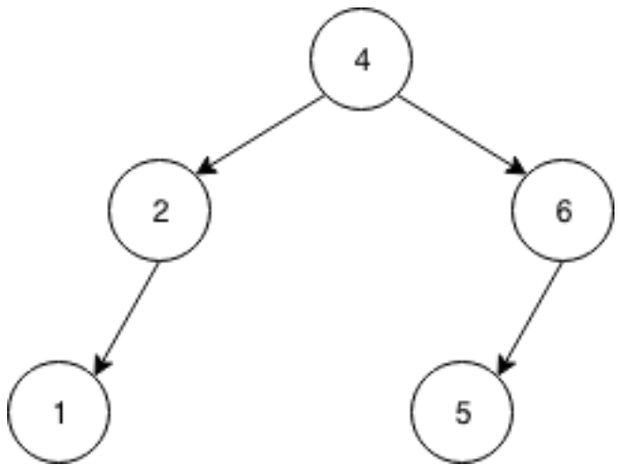
A



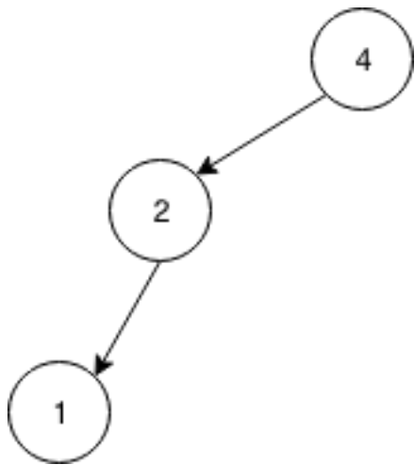
B



C



D



E

QUESTION 3.1.5 IS BST

QUESTION 3.1.6 VISITE POSSIBLES EN DFS

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

QUESTION 3.1.8 ENUMERER EN ORDRE CROISSANT LES CLÉS

2-3, RED-BLACK

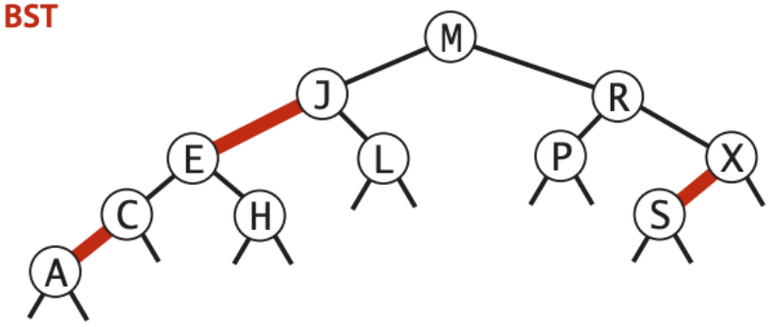
Un arbre 2-3 a des noeuds contenant au plus deux clés et trois enfants: l'enfant de gauche est < que la première clé, l'enfant du milieu > que la première clé et < que la seconde, et l'enfant de droite est > que la seconde clé. Il est facile de maintenir l'équilibre d'un arbre 2-3.

Un arbre red-black, est un BST avec des liens rouges et noirs, avec quelques contraintes:

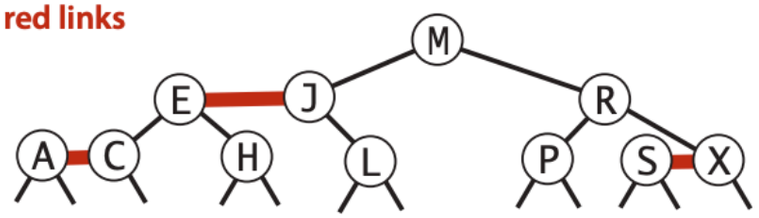
- Les liens rouges sont uniquement vers la gauche
- Pas deux liens rouges d'affilées
- Equilibre parfait sur les liens noirs

Il y a correspondance parfaitement entre 2-3 et red-black.

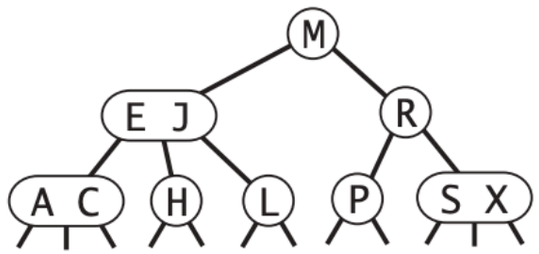
red-black BST



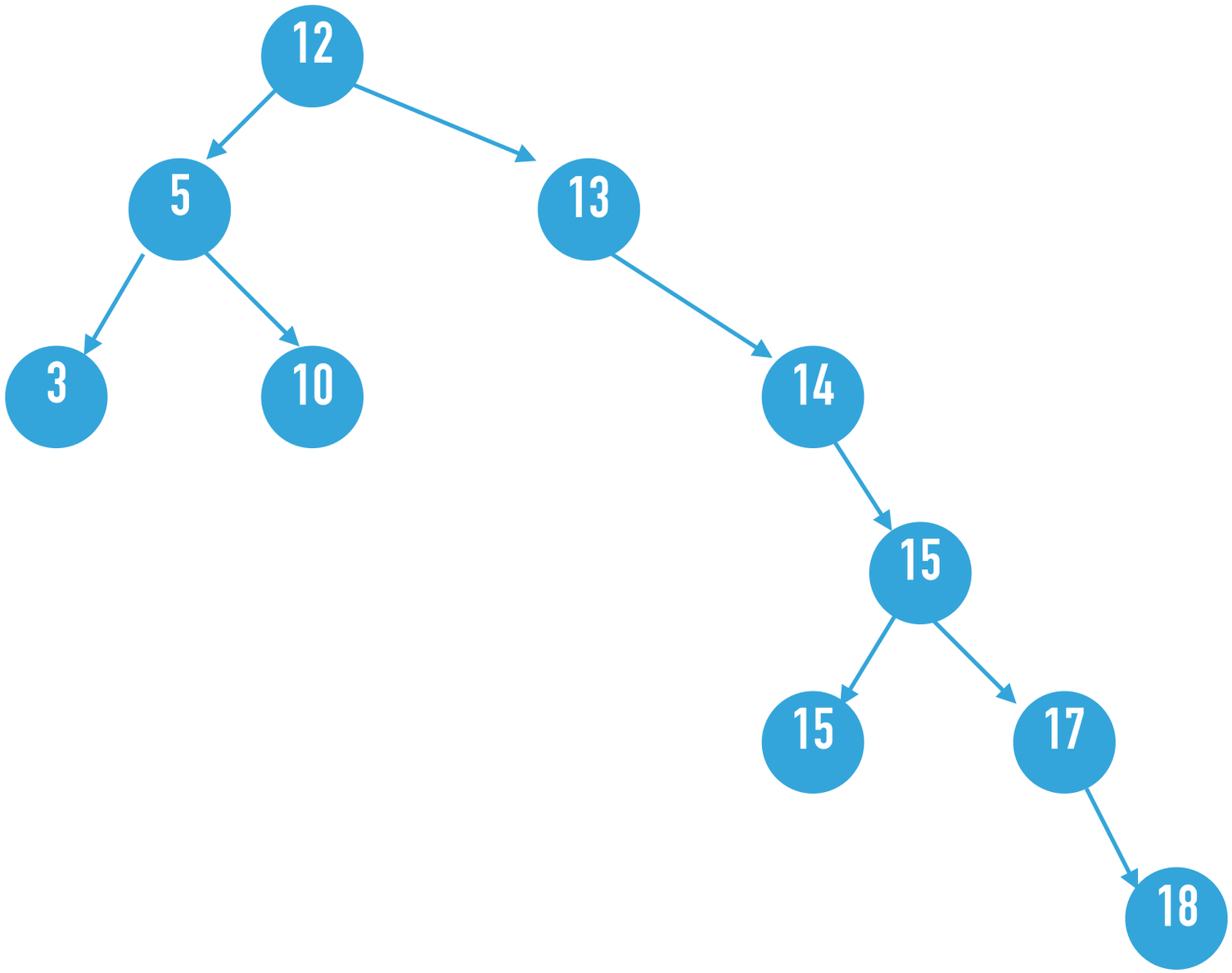
horizontal red links



2-3 tree



QUESTION 3.1.9 INSERTIONS



QUESTION 3.1.9 INSERTIONS

12

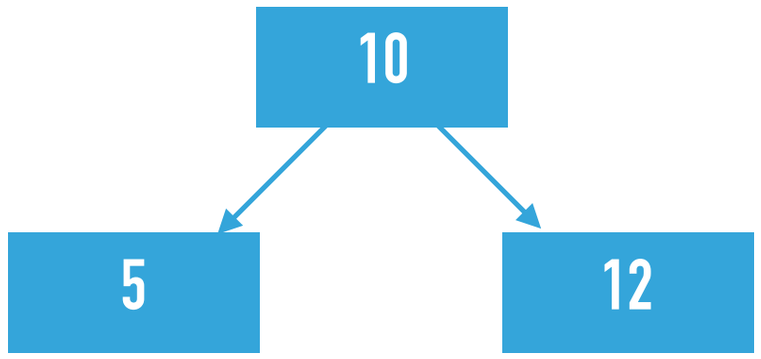
~~12~~, 5, 10, 3, 13, 14, 15, 17, 18, 15

QUESTION 3.1.9 INSERTIONS

5 12

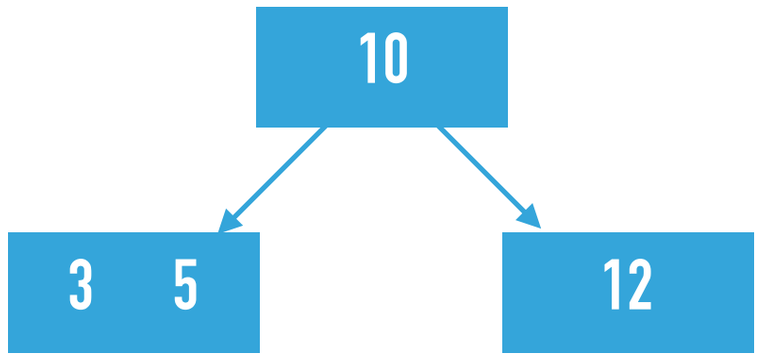
~~12~~, ~~5~~, 10, 3, 13, 14, 15, 17, 18, 15

QUESTION 3.1.9 INSERTIONS



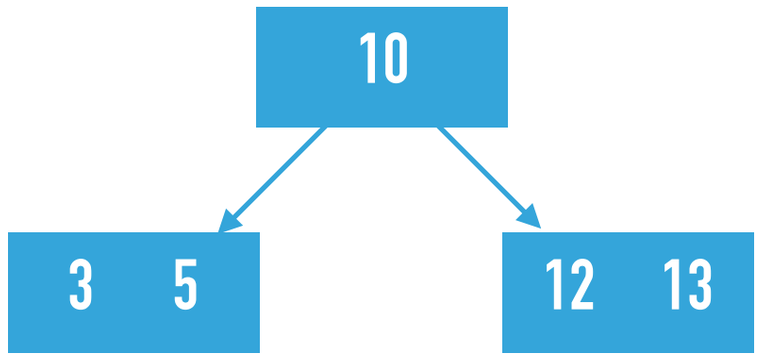
~~12, 5, 10~~, 3, 13, 14, 15, 17, 18, 15

QUESTION 3.1.9 INSERTIONS



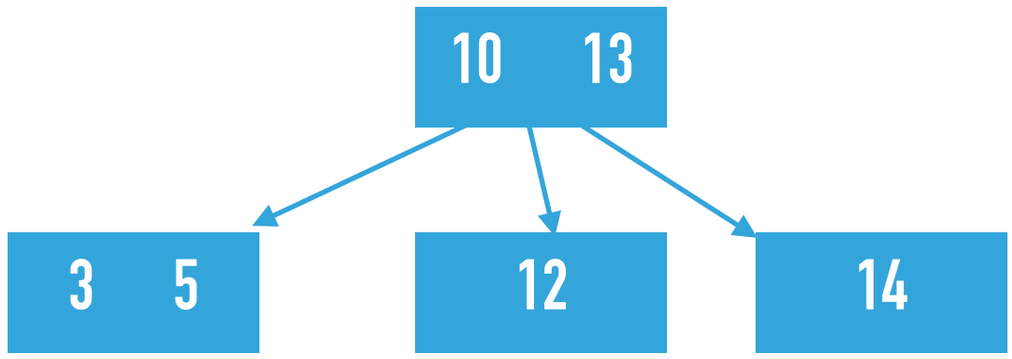
~~12, 5, 10, 3~~, 13, 14, 15, 17, 18, 15

QUESTION 3.1.9 INSERTIONS



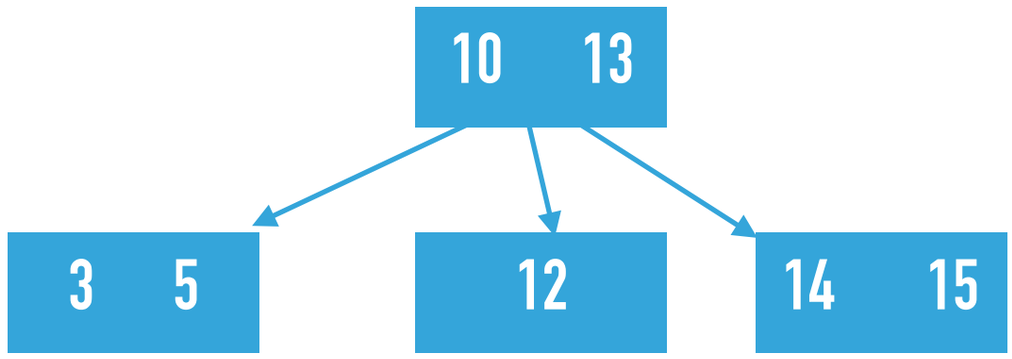
~~12, 5, 10, 3, 13~~, 14, 15, 17, 18, 15

QUESTION 3.1.9 INSERTIONS



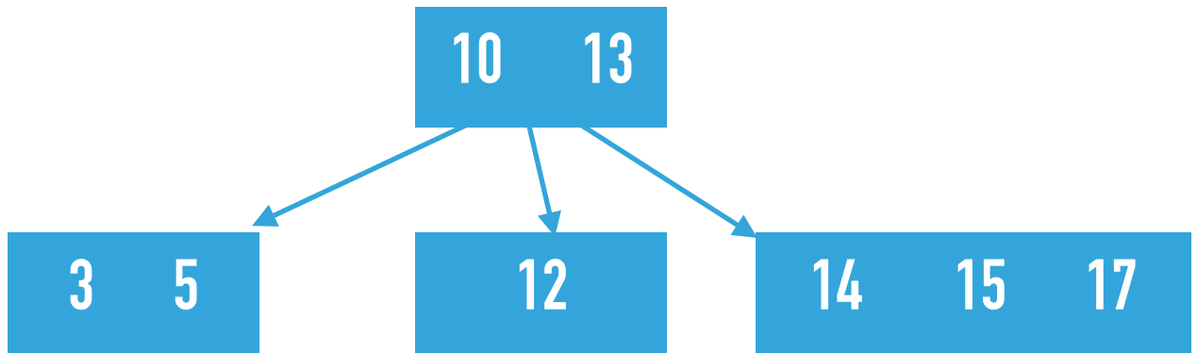
~~12, 5, 10, 3, 13, 14~~, 15, 17, 18, 15

QUESTION 3.1.9 INSERTIONS



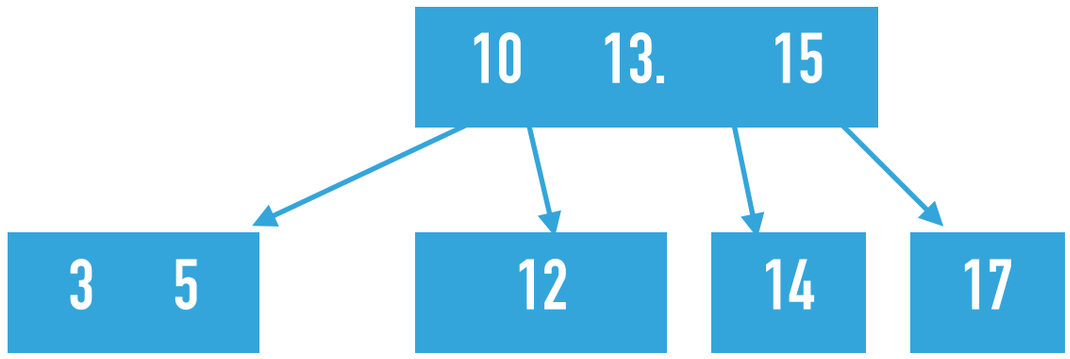
~~12, 5, 10, 3, 13, 14, 15~~, 17, 18, 15

QUESTION 3.1.9 INSERTIONS



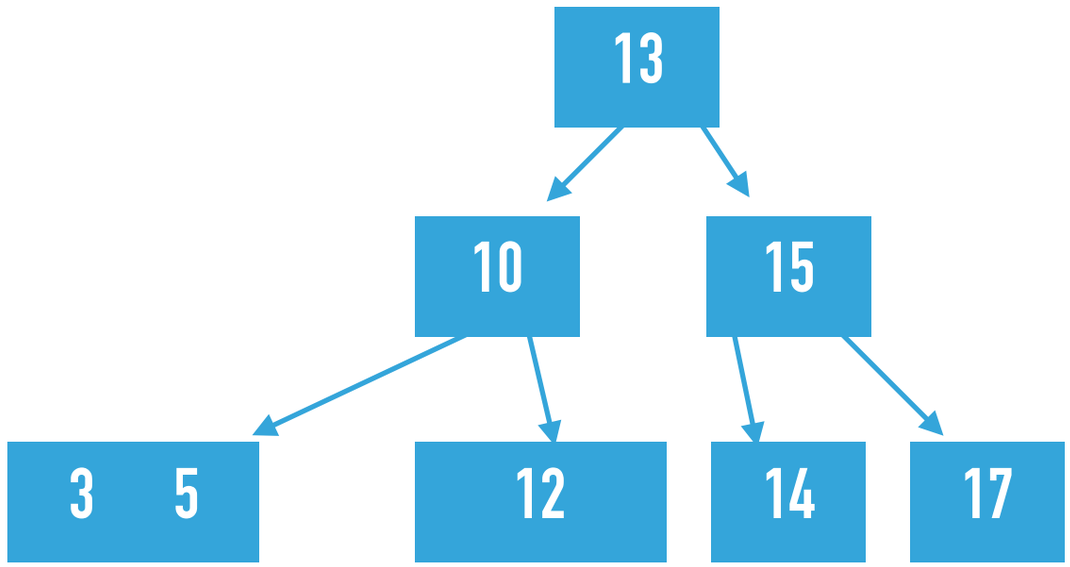
~~12, 5, 10, 3, 13, 14, 15, 17~~, 18, 15

QUESTION 3.1.9 INSERTIONS



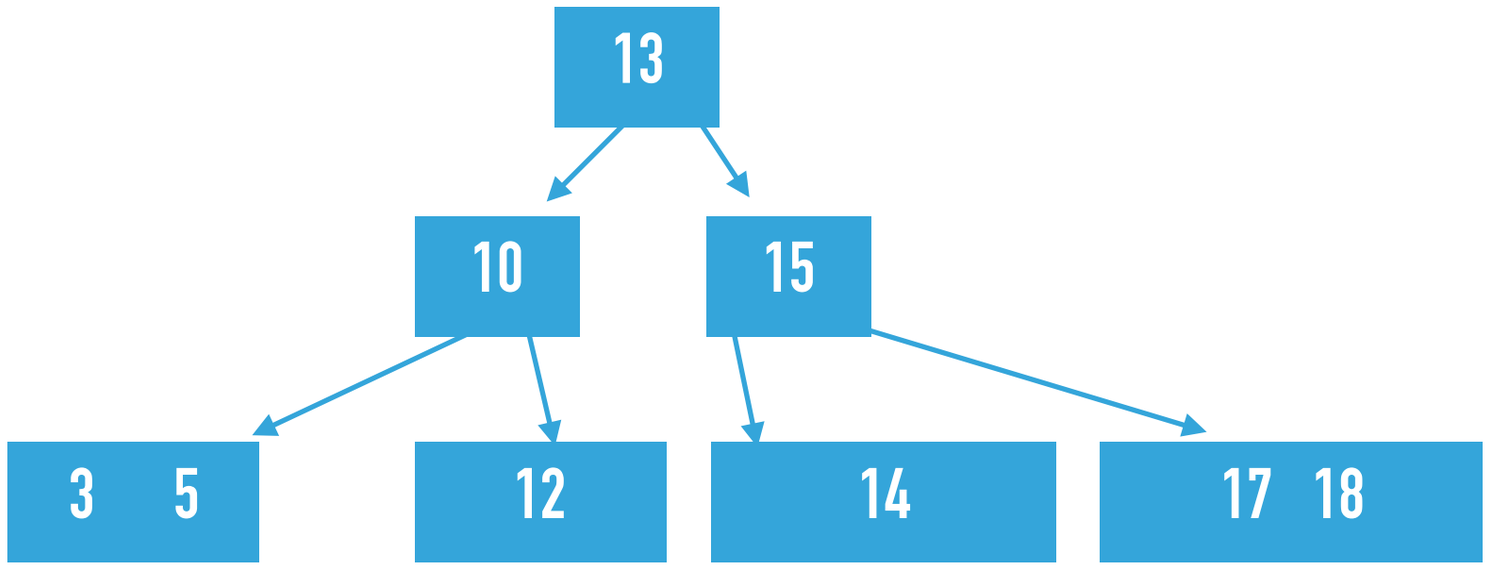
~~12, 5, 10, 3, 13, 14, 15, 17~~, 18, 15

QUESTION 3.1.9 INSERTIONS



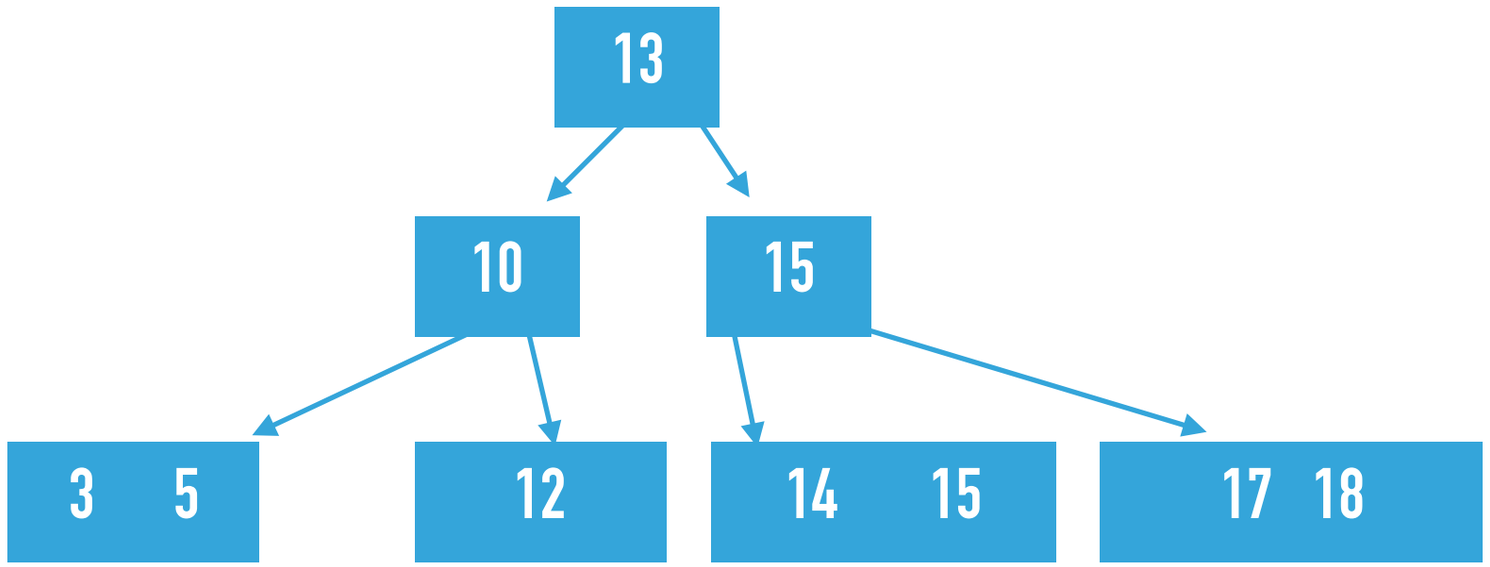
~~12, 5, 10, 3, 13, 14, 15, 17~~, 18, 15

QUESTION 3.1.9 INSERTIONS



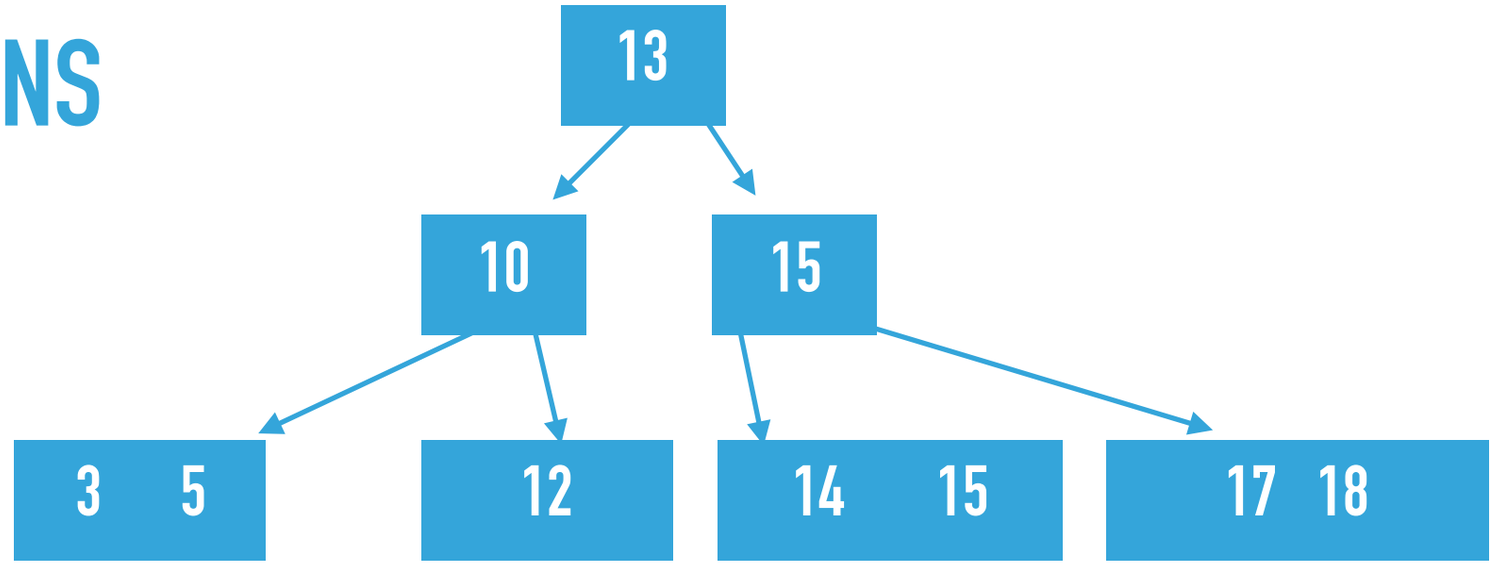
~~12, 5, 10, 3, 13, 14, 15, 17, 18~~, 15

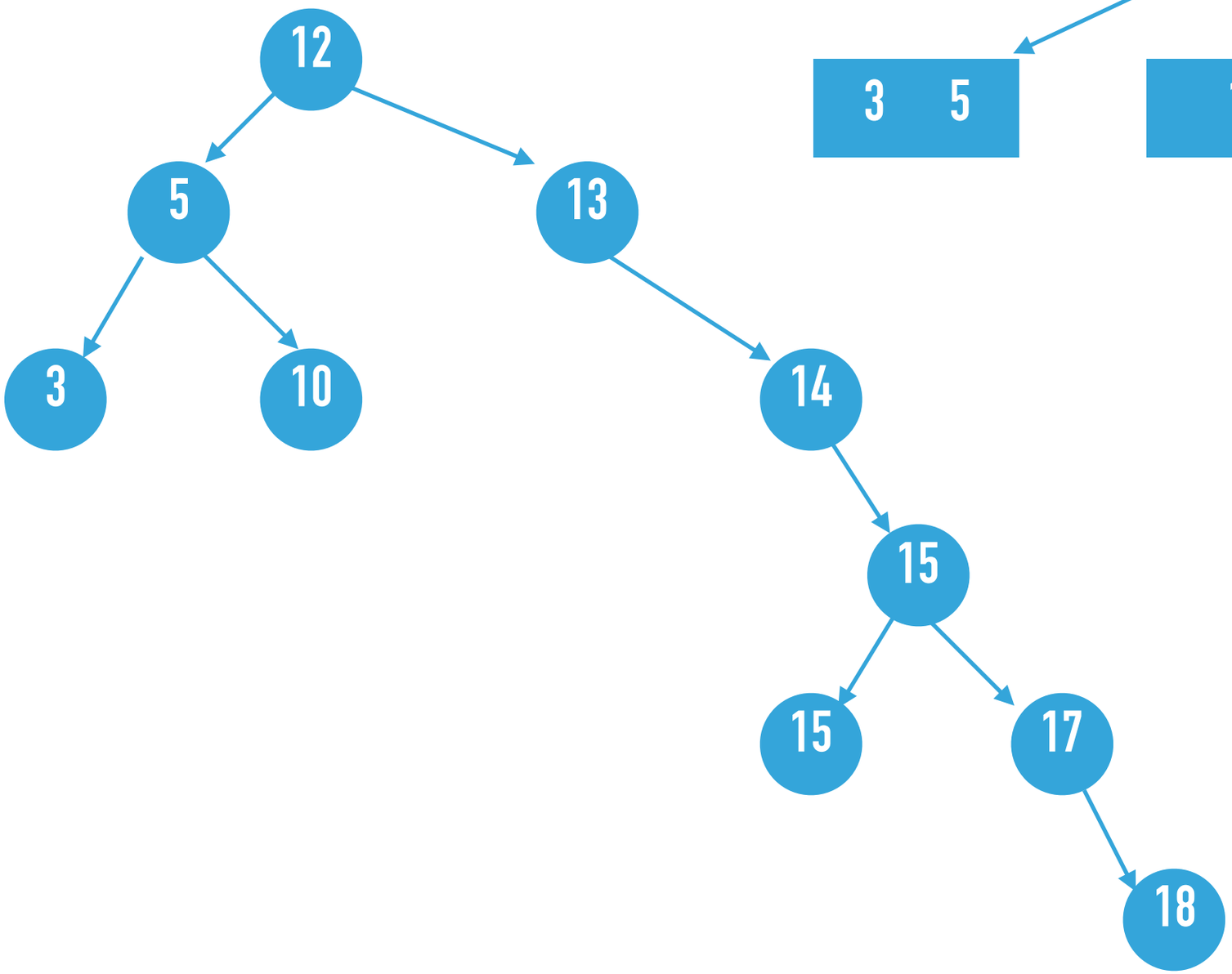
QUESTION 3.1.9 INSERTIONS



~~12, 5, 10, 3, 13, 14, 15, 17, 18, 15~~

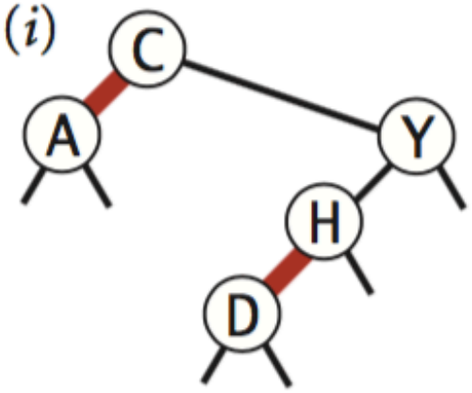
QUESTION 3.1.9 INSERTIONS



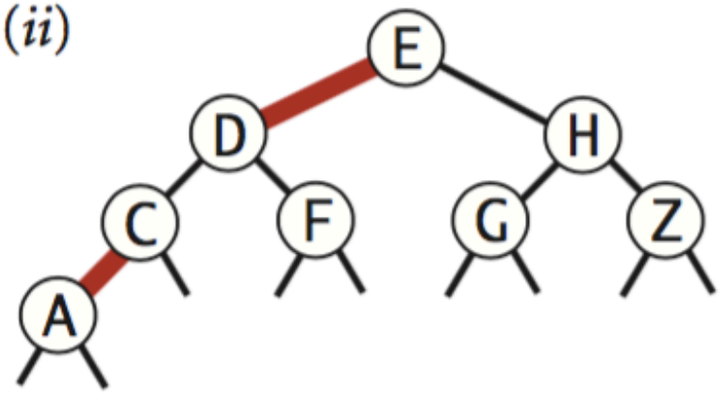


QUESTION 3.1.10 RB?

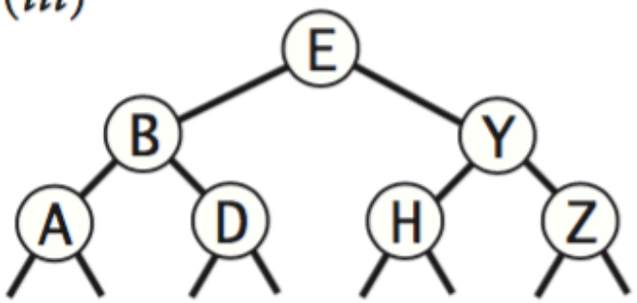
(i)



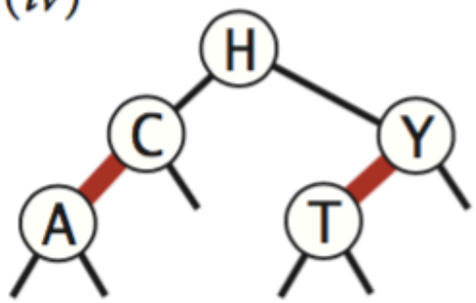
(ii)



(iii)



(iv)



QUESTION 3.1.7 IS RED BLACK

algorithm (data structure)	worst-case cost (after N inserts)		average-case cost (after N random inserts)		efficiently support ordered operations?
	search	insert	search hit	insert	
sequential search (unordered linked list)	N	N	$N/2$	N	no
binary search (ordered array)	$\lg N$	N	$\lg N$	$N/2$	yes
binary tree search (BST)	N	N	$1.39 \lg N$	$1.39 \lg N$	yes
2-3 tree search (red-black BST)	$2 \lg N$	$2 \lg N$	$1.00 \lg N$	$1.00 \lg N$	yes

Cost summary for symbol-table implementations (updated)