Arbres B (B-Tree)

Section 4.7 (Weiss)

É. Baudry, Notes de cours

Motivation

- Les arbres binaires offrent des opérations d'insertion, de recherche et d'enlèvement avec une complexité temporelle de

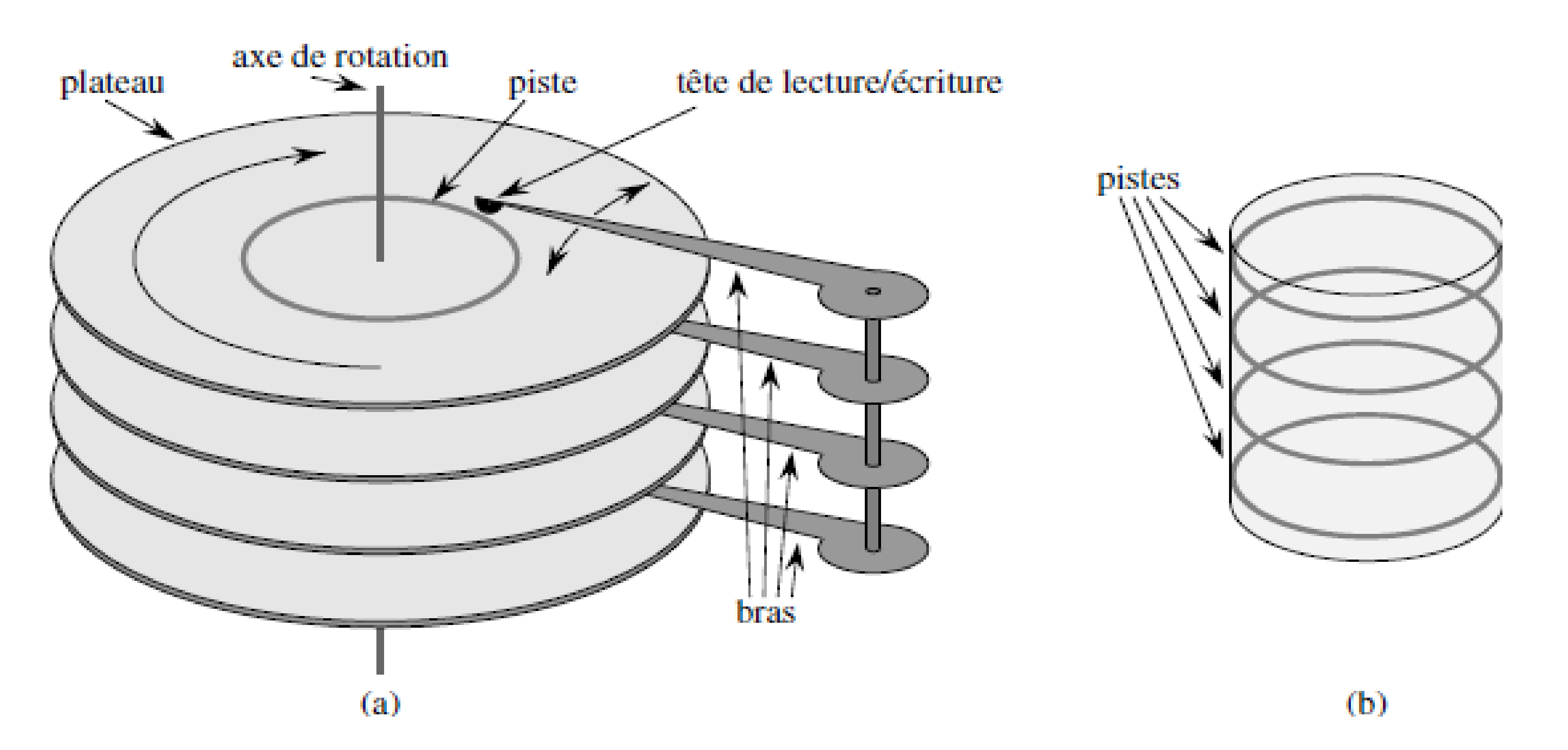
  O(log n)
- Le temps effectif de ces opérations:
  - **c** log n

Motivation

- Les arbres B visent à réduire la constante **c** lorsque :
  - l'accès à la mémoire (généralement secondaire) a un temps de latence significatif ;
  - le débit de lecture de la mémoire contiguë est élevé
- « Dans une application de B-arbre classique, la quantité de données gérées est si grandes qu'elles ne tiennent pas toutes en même temps dans la mémoire principale. » [1]

Arbres B

- Motivation, Lecteur de disque



Arbres B

- Deux composantes principales du temps d'exécution
  - Le nombre d'accès disque
    - Se mesure en fonction du nombre de pages de données à lire ou écrire sur le disque
    - La complexité en temps de l'algorithme de recherche externe sera calculée en fonction du nombre de transferts d'information
  - Le temps processeur (temps de calcul)
- Algorithmes de B-arbres copient certaines pages du disque vers la mémoire principale et réécrivent sur le disque les pages modifiées

Arbres B

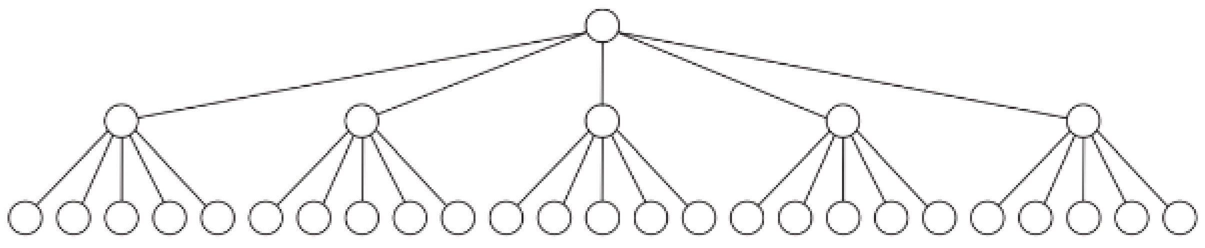
- À un instant donné, le système ne peut garder qu'un nombre de pages limité en mémoire principale
- Il faut lire ou écrire le plus d'informations possible à chaque accès
- Nœud d'arbre B a souvent la taille d'une page de disque entière et cette taille limite le nombre d'enfants que peut posséder un nœud

Arbres B

- Arbres B offrent une solution au problème de temps de latence (temps d'accès)
  - La durée écoulée entre le moment où la requête est donnée et le moment de réponse
- Stratégie consiste à couper l'espace de recherche par un facteur significativement plus grand que deux
- Un grand facteur de ramification réduit énormément et la hauteur de l'arbre et le nombre d'accès disque nécessaires

Arbre B

- Nous souhaitons réduire le nombre d'accès au disque à une valeur très petite, comme trois ou quatre
- Nous sommes prêts à écrire un code complexe pour y parvenir, car les instructions machine sont essentiellement gratuites comparativement à la latence d’accès au disque



5-ary tree of 31 nodes has only three levels

Arbres B

- Nœuds avec des dizaines ou même des centaines et plus d'enfants
- Nombre d'enfants optimal à utiliser définit par
  - la taille d'un bloc contiguë de mémoire pouvant être lu avec un seul accès disque (exemple : un secteur)

Arbres B

- Exemple: Service d’authentification de Facebook
  - Plus de un milliard d’utilisateur: $n = 10^9$
  - Ce service peut être implémenté à l’aide d’un ABR équilibré qui associe une adresse courriel (la clé) à une référence vers l’objet fiche de l’utilisateur (mot de passe, etc.)

Arbres B

- Exemple: Service d'authentification de Facebook, opération recherche
- Arbre est stocké sur un disque
- Le temps total pour effectuer une recherche dans cet arbre
- $t = \log n * latance_{moyenne} + \frac{\log n * sizeof(noeud)}{débit}$

Arbres B

- Exemple: Service d'authentification de Facebook
- opération recherche, supposons
  - Arbre AVL
  - Latence moyenne => 10 ms
  - Débit moyen de lecture de 100 Mo/s
  - Taille de la clé (courriel): 104 octets
  - Références gauche, droite + contenu : 8 octets chacun => 8*3 = 24 octets

128 octets Par nœud

- $t = \log n * latance_{moyenne} + \frac{\log n * sizeof(noeud)}{débit}$

Arbres B

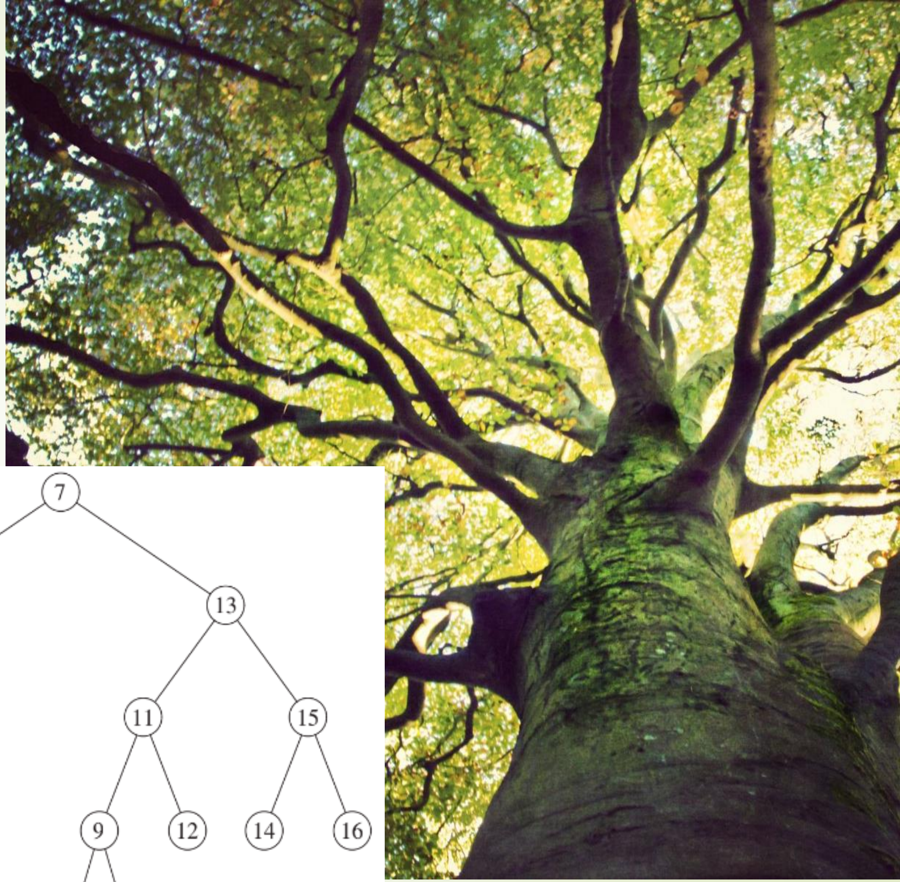
- Opération recherche
- Latence moyenne => 10 ms
- Débit moyen de lecture de 100 Mo/s
- Noeud: 128 octets

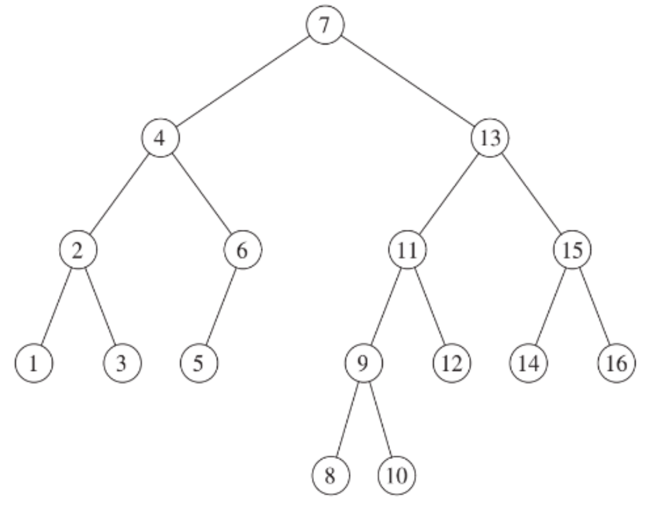
$$t = (1.44 log_2 n - 0.328) * 10ms + \frac{(1.44 log_2 n - 0.328) * 128 octets}{100 * 10^6 octets/s}$$

$$t = 0.3 + 2.34 * 10^{-9}$$

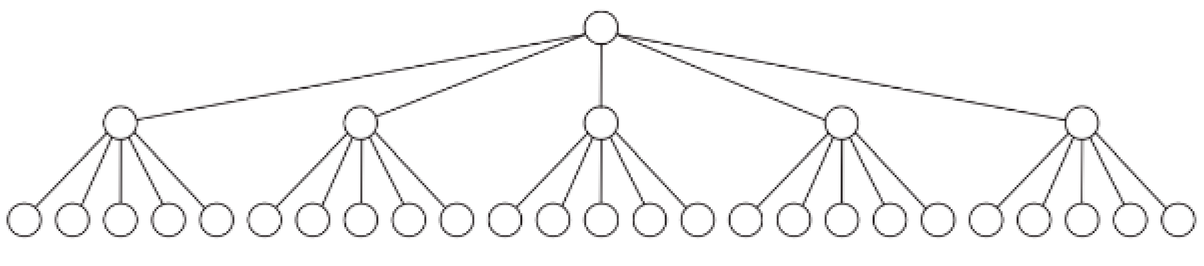
$$t \approx 0.3ms$$

Hauteur d'un arbre B





- Hauteur de
  - l'ABR équilibré avec n nœuds $h \approx log_2 n$

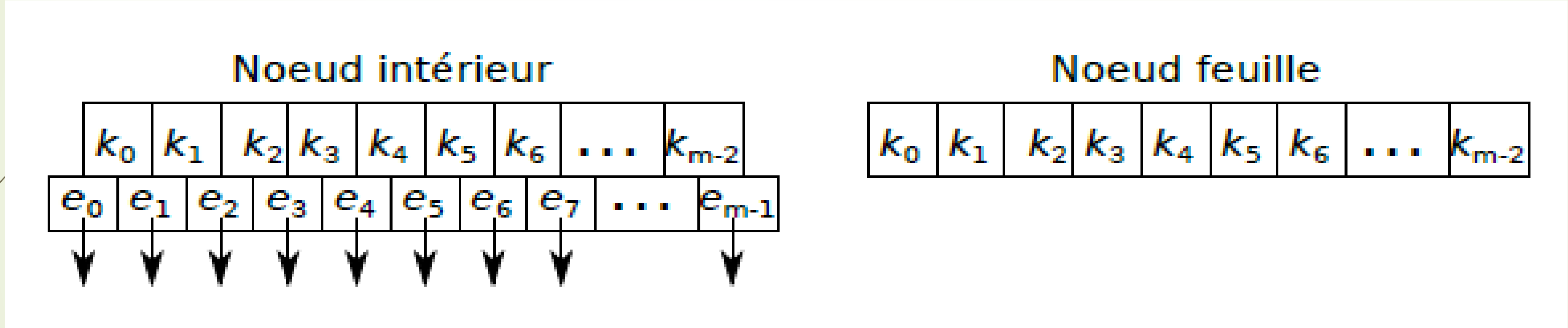


5-ary tree of 31 nodes has only three levels

- Arbre avec m nœuds
  - $h \approx log_m n$

Définition

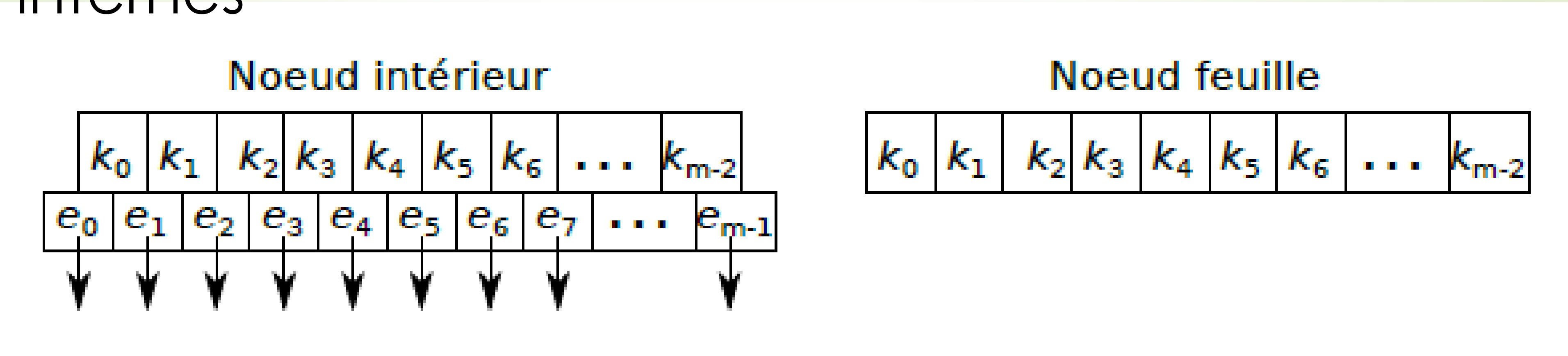
- **Un arbre B d'ordre m** ⟶ **B(m/2,m)** est un arbre équilibré ayant les caractéristiques suivantes:
  - Tous les nœuds ont au plus **m** enfants;
  - Les nœuds intérieurs stockent une liste de clés triées < k_0, k_1,…> et des pointeurs vers les enfants;



  - Nombre de clés dans un nœud est égale au nombre d'enfants moins un;
  - Tous les nœuds intérieurs ont au moins $\left\lfloor \frac{m}{2} \right\rfloor$ enfants;
  - Toutes les feuilles sont à distance égale de la racine;
  - Tous les éléments accessibles depuis le i-ème enfant sont supérieurs à la clé k_{i-1} et inférieurs à la clé k_i

Arbres B, nœuds

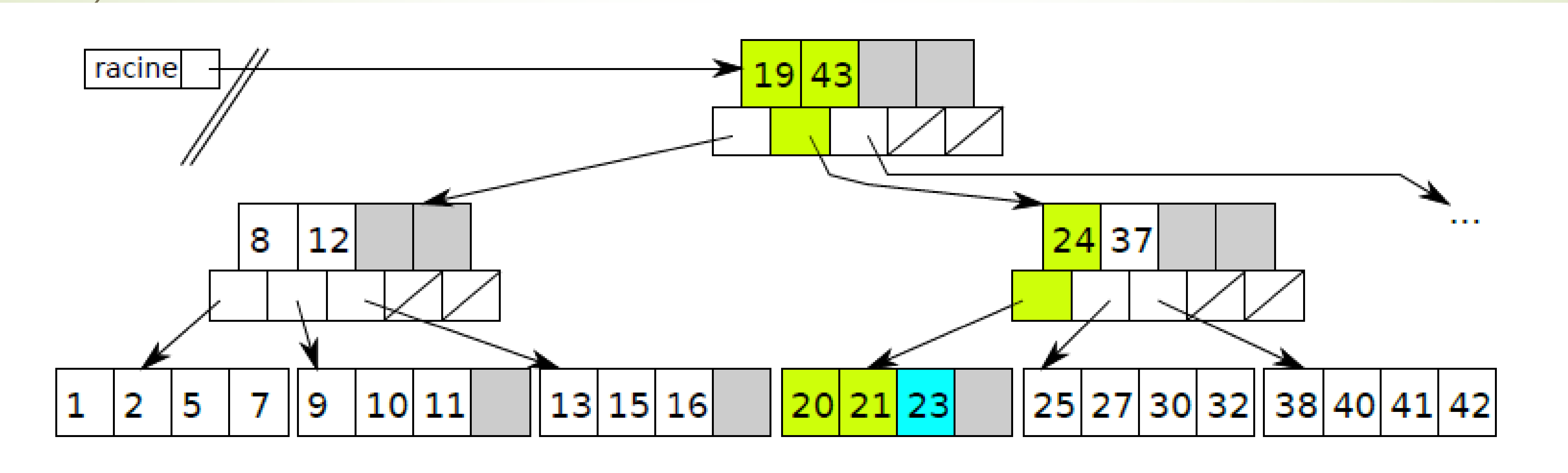
La structure des feuilles diffère de celle des nœuds internes



Arbres B d'ordre 5, B(2,5)

- Rechercher un élément e
- 1. Départ: la racine
- 2. Recherche la plus grande clé k_i dans le nœud courant tel quel k_i <= e ;
- si k_i = e => trouvé
- sinon, le i-ème nœud enfant devient le nœud courant et on répète l'étape 2 de la procédure ;
- 3. si on ne trouve pas la clé dans une feuille => l'élément e n'existe pas dans l'arbre.

  - 23?

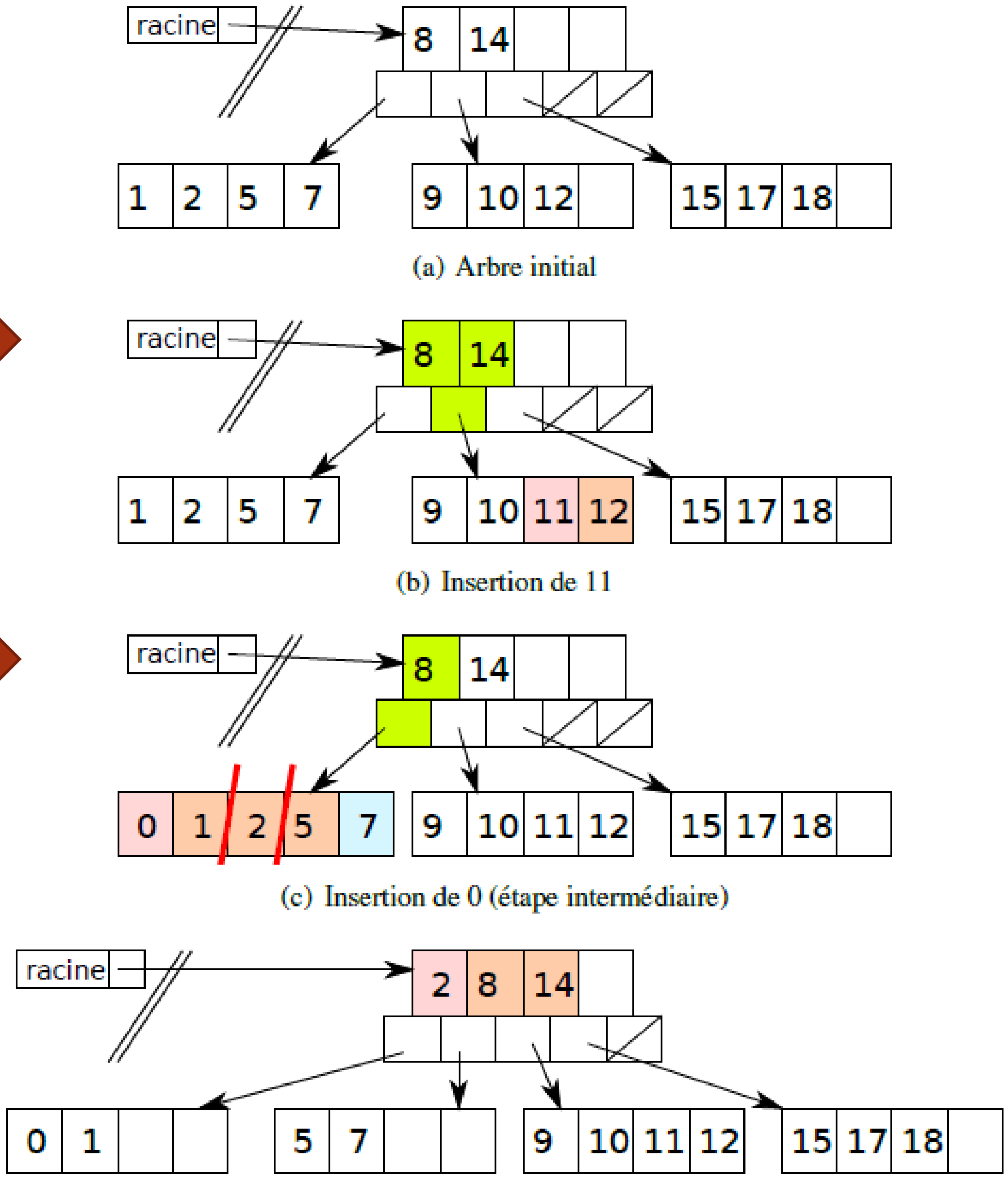


Insertion

- 1. Rechercher la feuille où devrait se trouver l'élément e
- 2. Insérer l'élément dans la feuille trouvée
  - Si la capacité du nœud n'est pas dépassée (un nœud contient au plus m-1 clés) – terminé
  - si la capacité est dépassée, on scinde le nœud en deux, et on remonte la clé médiane vers le nœud parent
  - si le nœud parent dépasse sa capacité, on le scinde à nouveau, et ce, jusqu'à temps de remonter à la racine ;
  - enfin, dans le cas où la capacité de la racine est dépassée, une nouvelle racine est créée

Insertion

- Insertion de 11
- Insertion de 0



(a) Arbre initial

(b) Insertion de 11

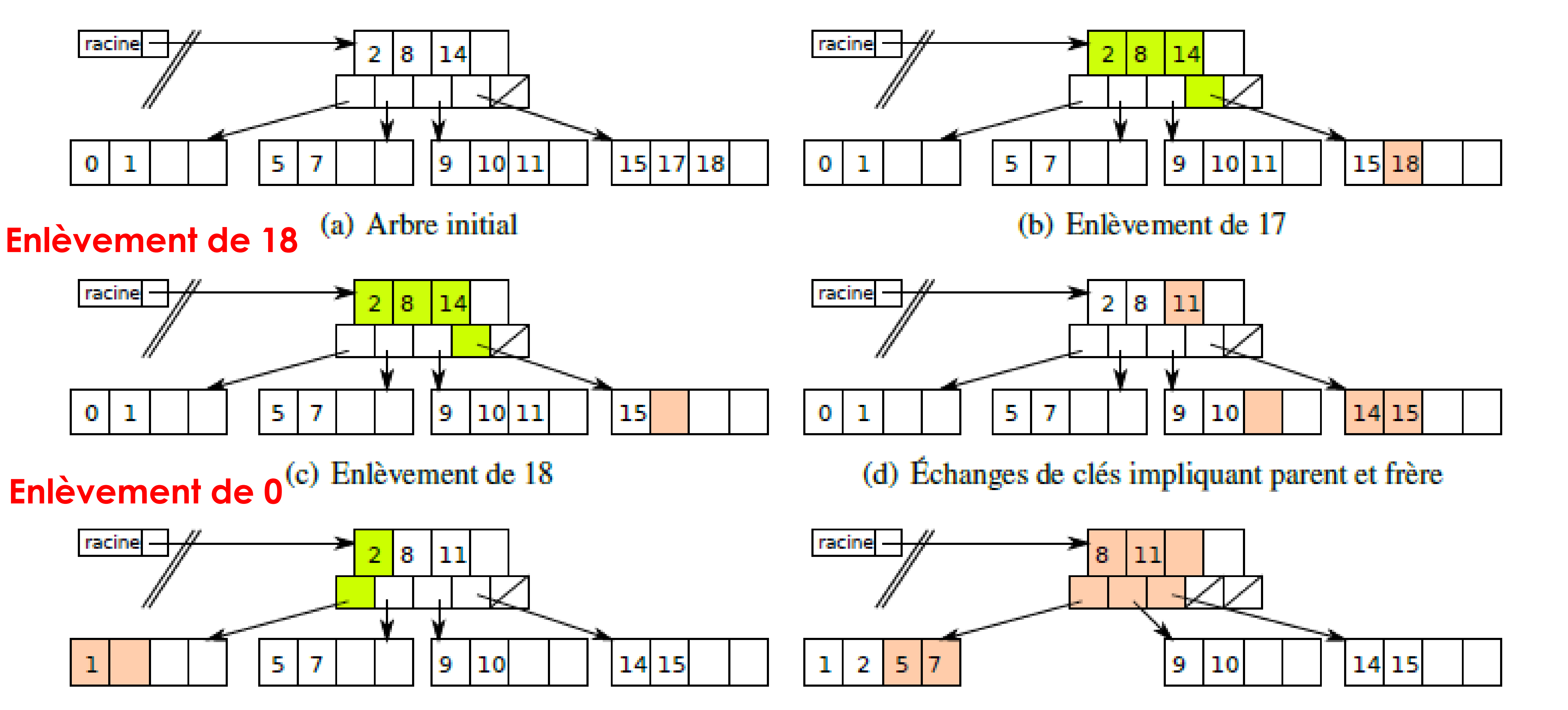
(c) Insertion de 0 (étape intermédiaire)

Arbres B, Enlèvement

- 1. Rechercher la feuille où se trouve l'élément e et on y enlève e;
- 2. Si le nombre de clés devient inférieur à $\left\lfloor \frac{m}{2} \right\rfloor$:
  - Si les 2 nœuds frères ont un nombre de clés supérieur à $\left\lfloor \frac{m}{2} \right\rfloor$, emprunter une clé d'un frère;
  - Sinon, on fusionne le nœud avec l'un de ses frères
    - Lors d'une fusion, on descend une clé du nœud parent et on répète l'étape 2 sur ce dernier

Enlèvement

Enlèvement de 17



(a) Arbre initial

(b) Enlèvement de 17

(c) Enlèvement de 18

(d) Échanges de clés impliquant parent et frère

(e) Enlèvement de 0)

(f) Fusion des feuilles

Arbres B

- Démo en ligne
- https://www.cs.usfca.edu/~galles/visualization/BTree.html