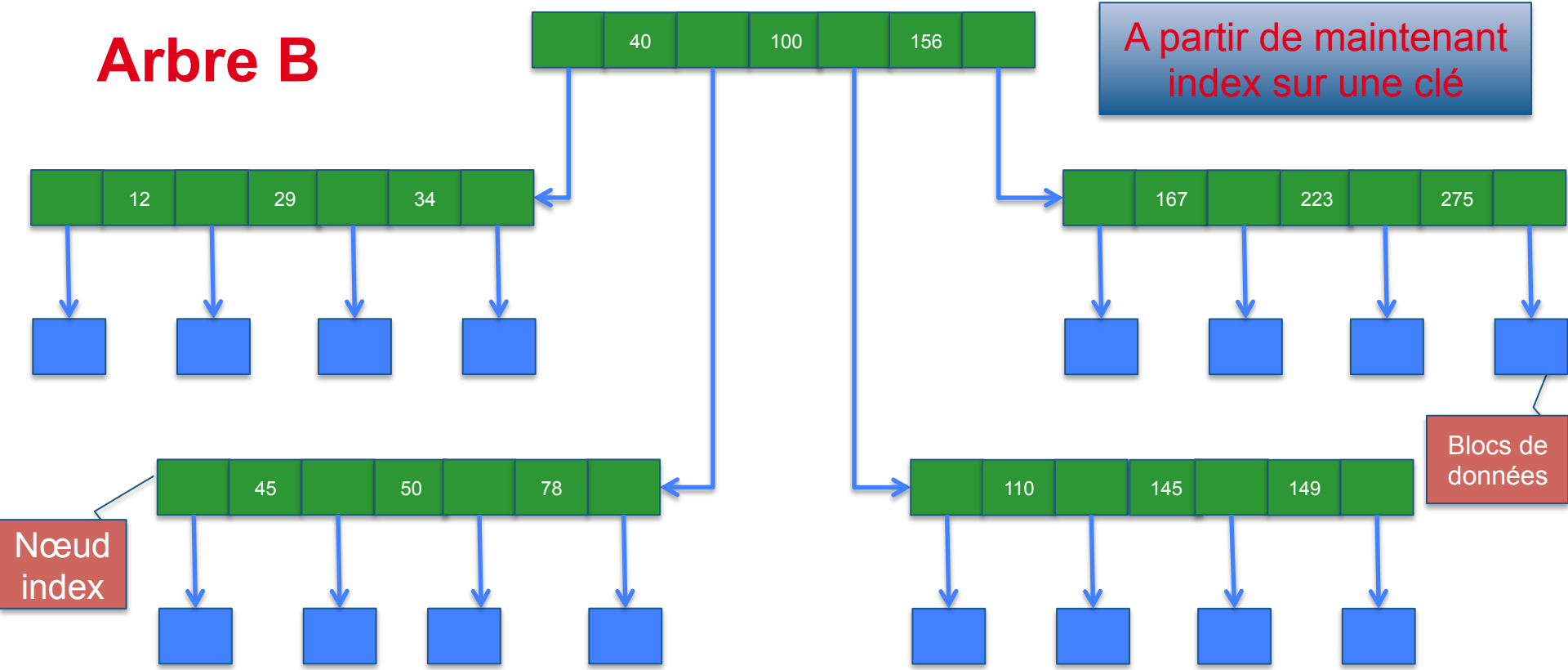
C018SA-W2-S4

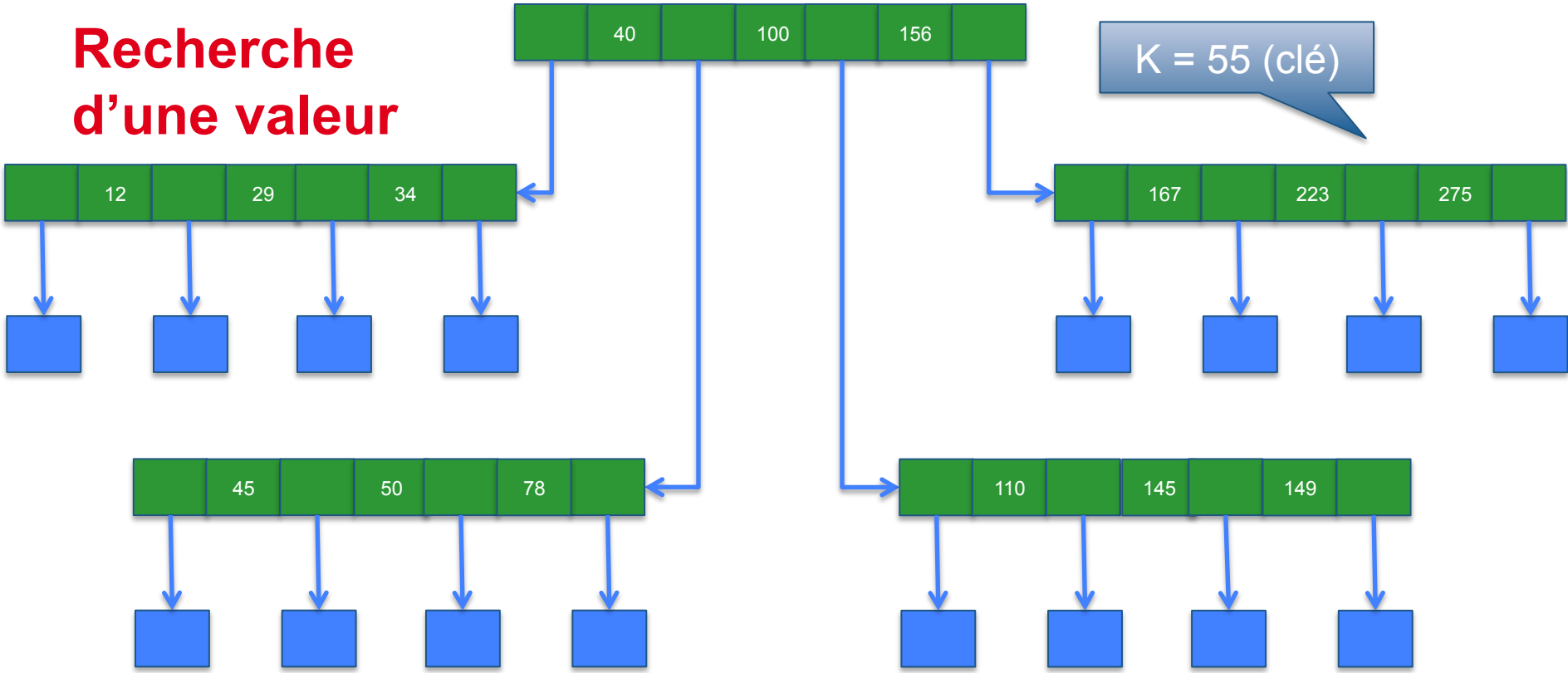
SEMAINE 2 : Indexation

1. Introduction
2. Hiérarchie de mémoire
3. Fichiers indexés
4. **Arbre-B**
5. Hachage
6. Hachage dynamique
7. Multi-hachage

Arbre B


40
100
156
12
29
34
167
223
275
45
50
78
110
145
149
A partir de maintenant index sur une clé
Blocs de données
Nœud index

Recherche d'une valeur



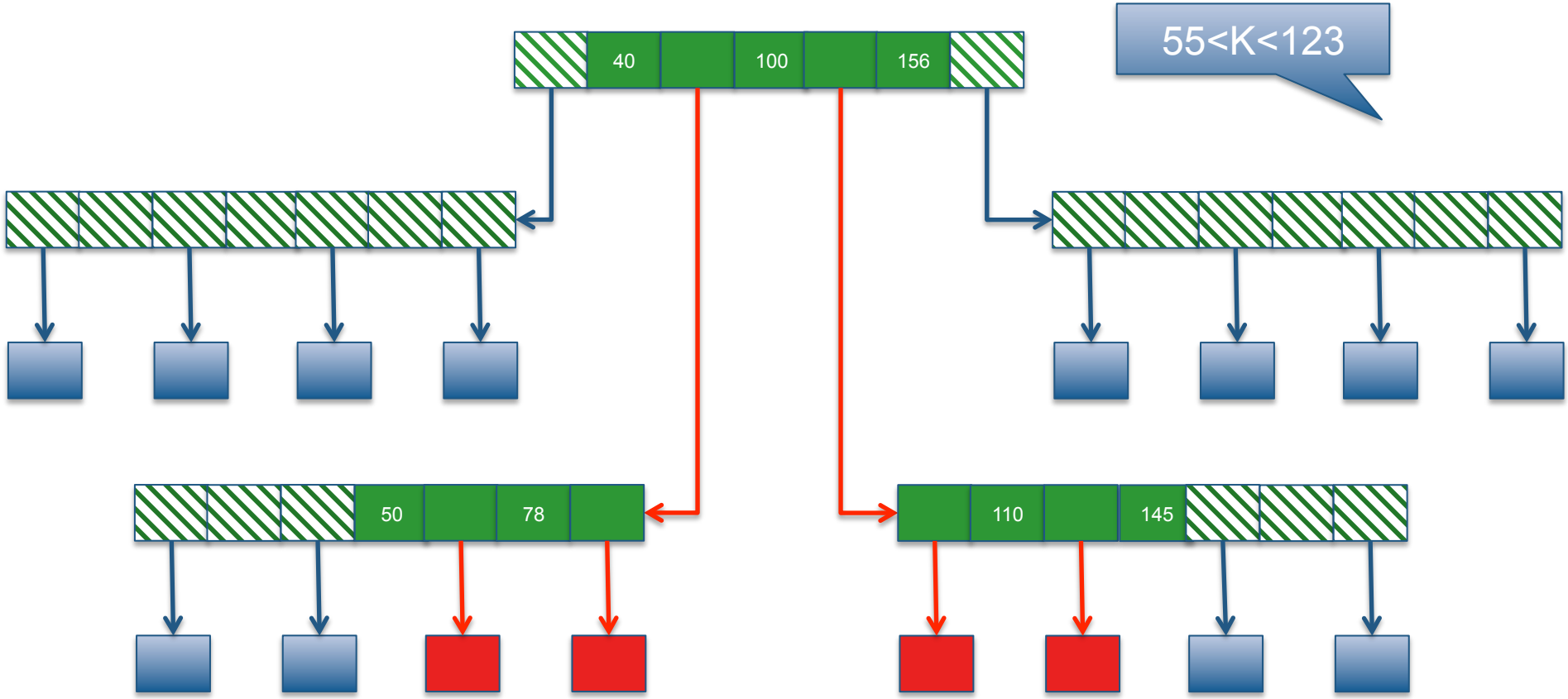
- Recherche d'une clé : descente en O(log(N)) accès disque (profondeur)
- Insertion : Recherche du bloc; insertion s'il y a de la place; sinon : éclatement

Arbre B : exemple

- En supposant
  - Des blocs de 8000 octets, adresses de 10 octets, clés de 4 octets
  - 8000/14 ≈ 571 adresses par nœud
- En supposant un remplissage de 75% des blocs de données

- Avec 1 niveau d'index : 3.5 mégaoctets
  - 571 × 571 ≃ 320 000 blocs de données
  - 8000 × 0.75 × 571 ≈ $3.5.10^6$ octets
- Avec 2 niveaux d'index : 2 gigaoctets
  - 571 × $3.5.10^6$ ≈ 2.10^9 octets
- Avec 3 niveaux d'index: 1 téraoctet
  - 571 × 2.10^9 ≈ 10^{12} octets

Possibilité de faire des requêtes d'intervalle

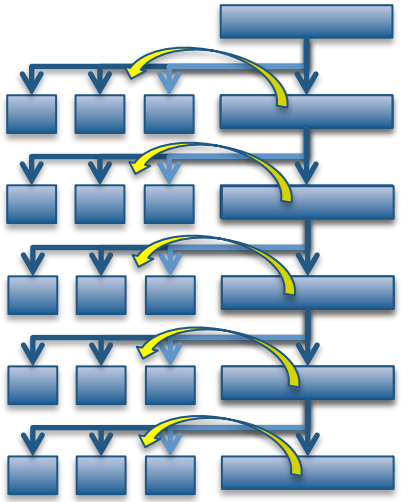


Arbre B - équilibrage

Si l'arbre est équilibré, OK

Sinon la profondeur peut être en O(n)

- Techniques pour équilibrer l'arbre à coût minimum
  - Amortir le travail : à chaque lecture/ écriture, on équilibre localement



Arbre déséquilibré - rééquilibrage

MERCI