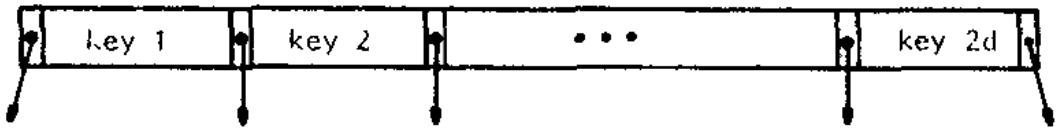
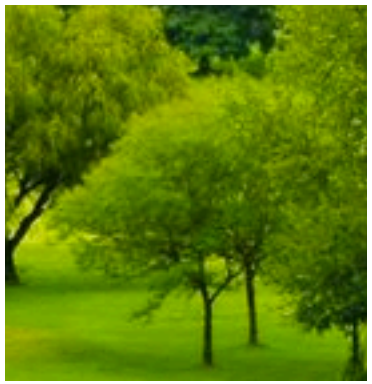


FIGURE 4. A node in a B-tree of order d with $2d$ keys and $2d + 1$ pointers.

Conception Avancée de Bases de Données

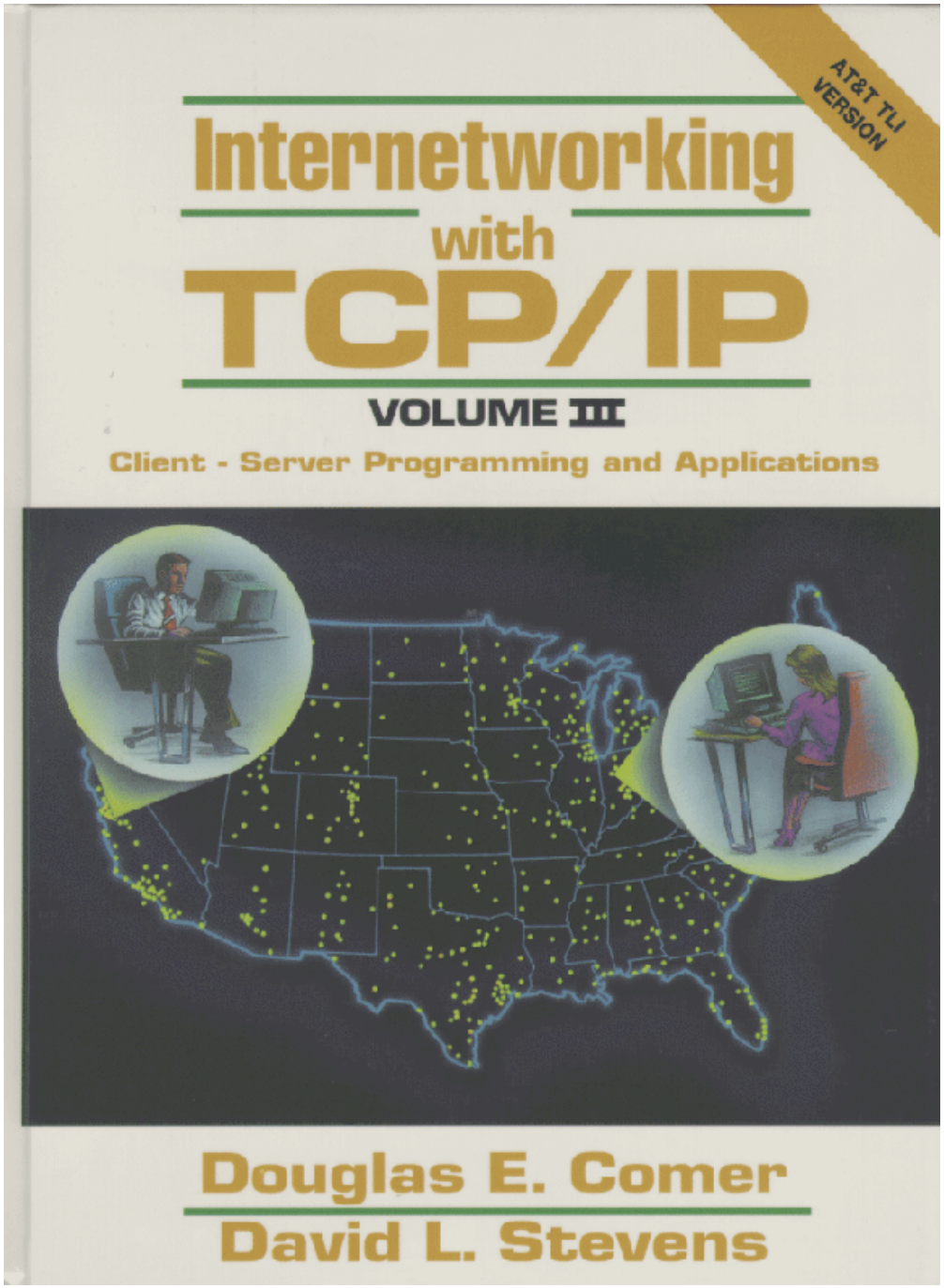
B+Tree



Traduction en cours

Seminal Article

- **The Ubiquitous B-Tree**
- DOUGLAS COMER
- *Computer Sctence Department, Purdue Untverstty, West Lafayette, Indiana 47907*


AT&T TLI VERSION
Internetworking with TCP/IP
VOLUME III
Client - Server Programming and Applications
Douglas E. Comer
David L. Stevens

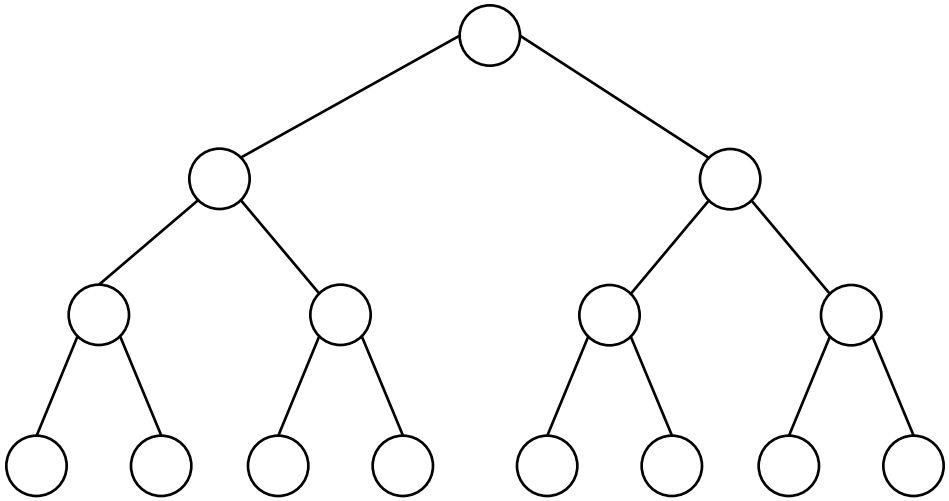
Trees (dads : Dictionary of Algorithms and Data Structures)

- Binary Tree
  - A tree with at most two children for each node.
- Binary Search Tree
  - A binary tree where every node's left subtree has keys less than the node's key, and every right subtree has keys greater than the node's key.
- B-Tree
  - A BST in which every node has between m and 2m children, where m is a fixed integer.

NIST
National Institute of Standards and Technology

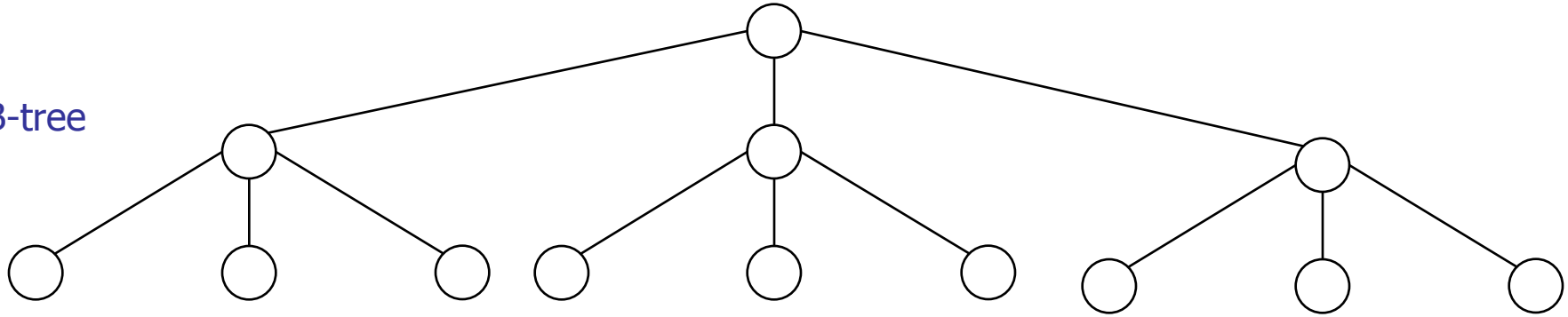
http://xlinux.nist.gov/dads/HTML/

BST B-tree

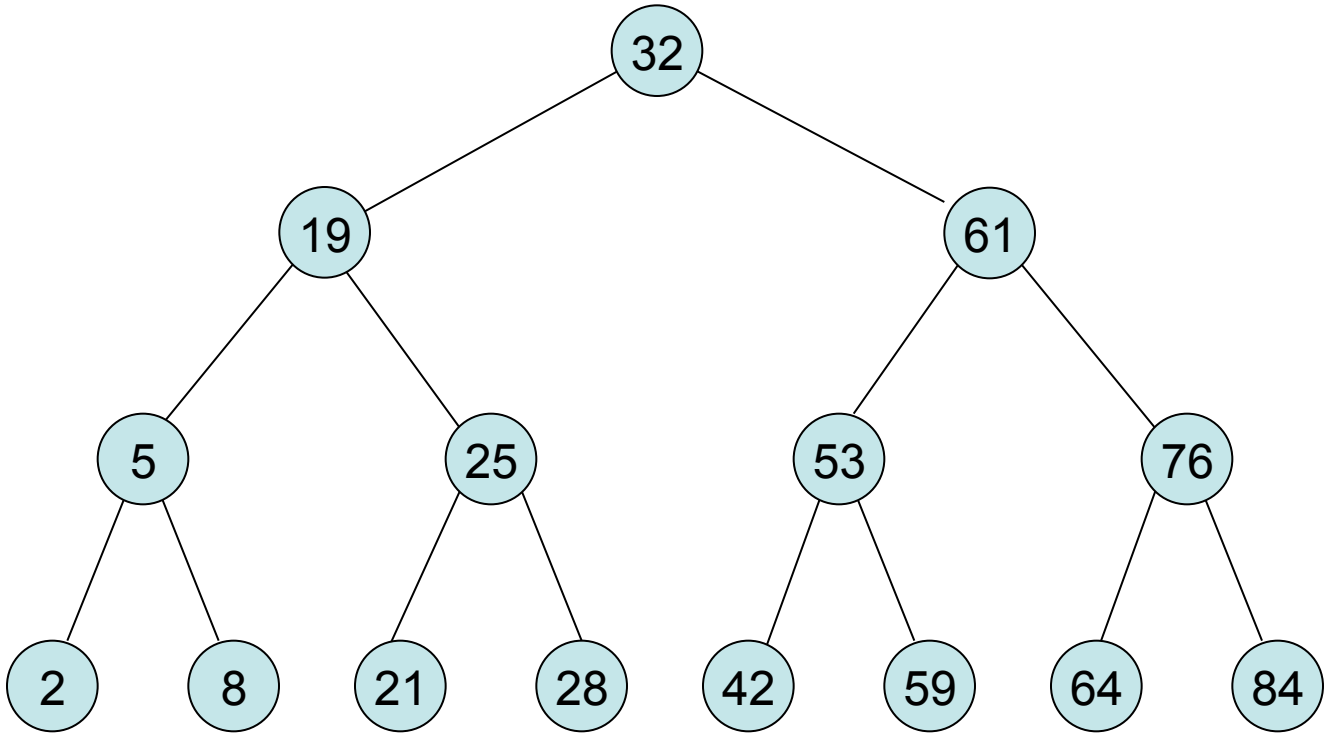


BST

B-tree

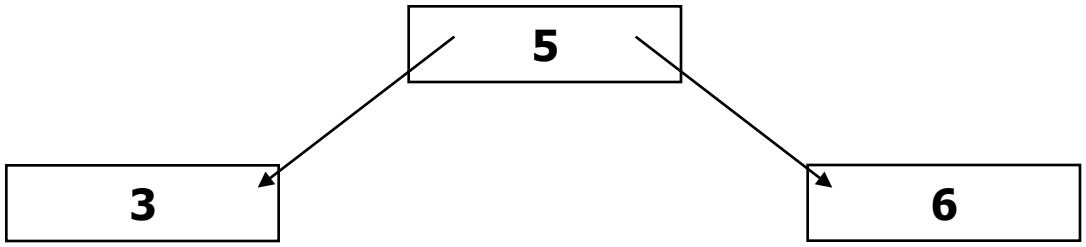


BST example



Arbre binaire de recherche (BST)

- Un arbre binaire de recherche est un arbre binaire dans lequel :
  - Chaque noeud possède une clé,
  - Telle que chaque nœud du sous-arbre gauche ait une clé inférieure ou égale à celle du nœud considéré,
  - Et chaque noeud du sous-arbre droit possède une clé supérieure ou égale à celle-ci

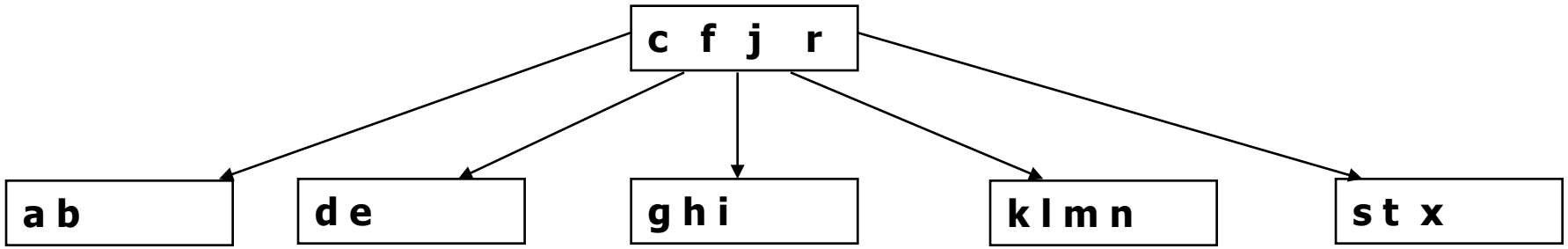


Arbre équilibré (B tree)

- Structure arborescente dans laquelle tous les chemins de la racine aux feuilles ont même longueur.
  - Tous les noeuds feuilles sont au même niveau
  - La hiérarchie de l'arbre grossit par la racine

- Par opposition aux arbres dégénérés (arbres peignes).
  - Opération de recherche de complexité logarithmique au lieu de complexité linéaire.

Arbre B (Btree) d'ordre 2

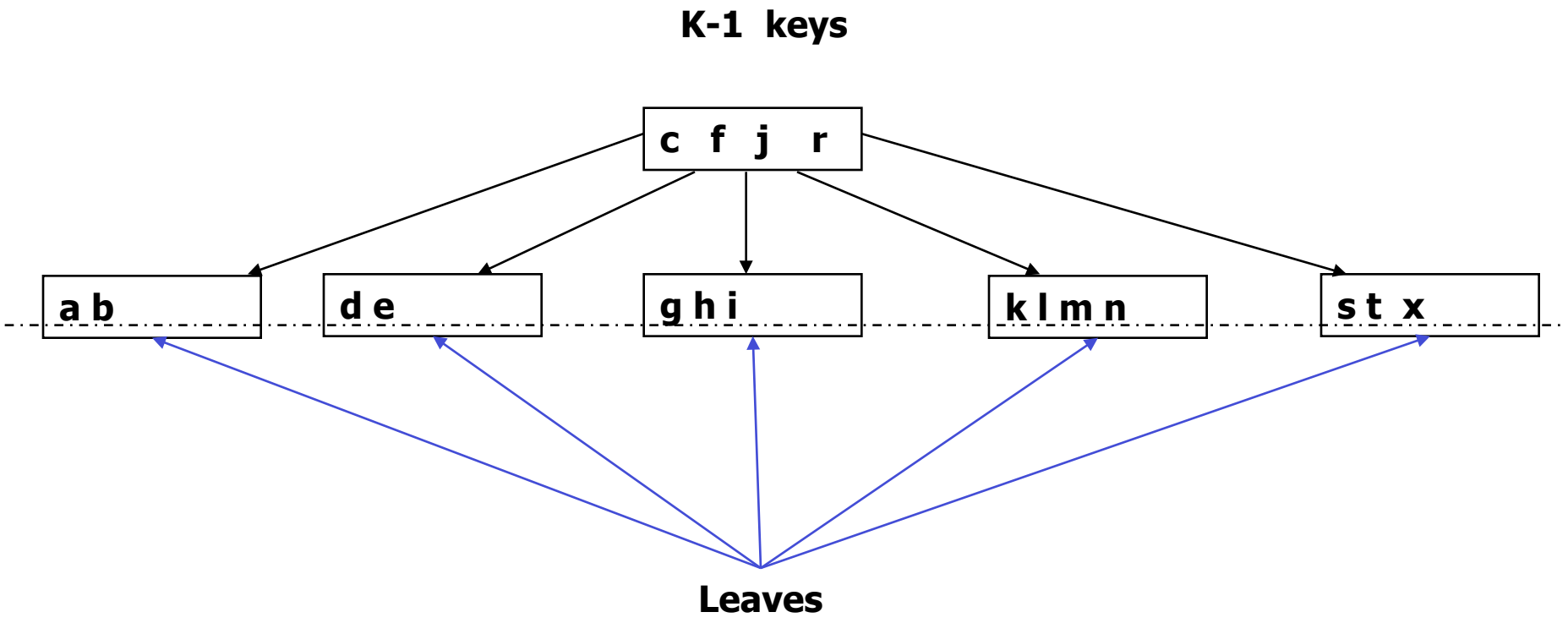


Un arbre-B d' ordre M Chaque noeud contient au moins M et au plus 2M éléments

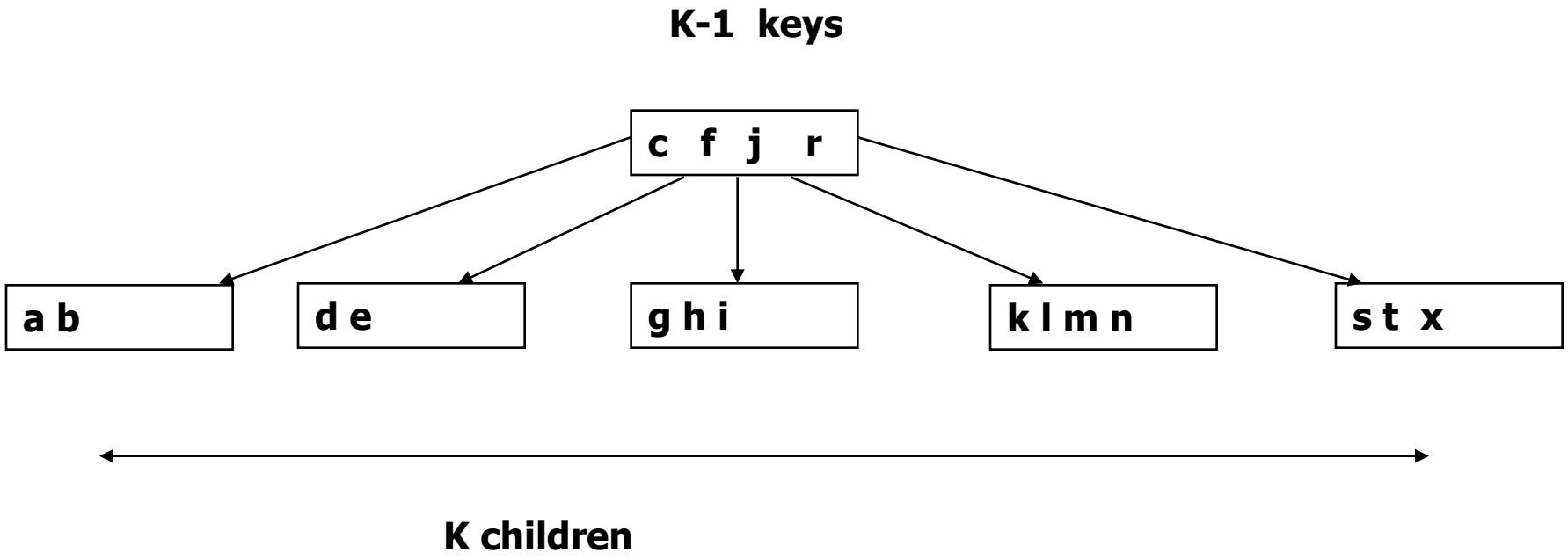
M order B-tree

- A B-tree of order m satisfies:
  1. Every node has at most 2m children.
  2. Every node has at least m children.
  3. The root has at least two children.
  4. All leaves appear in the same level
  5. A non-leaf node with k children contains k−1 keys.

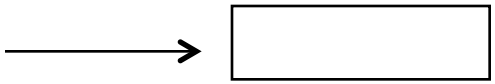
All leaves appear in the same level



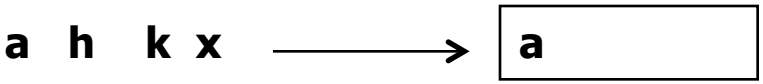
A non-leaf node with k children contains k–1 keys.



Arbre B (ordre 2) création

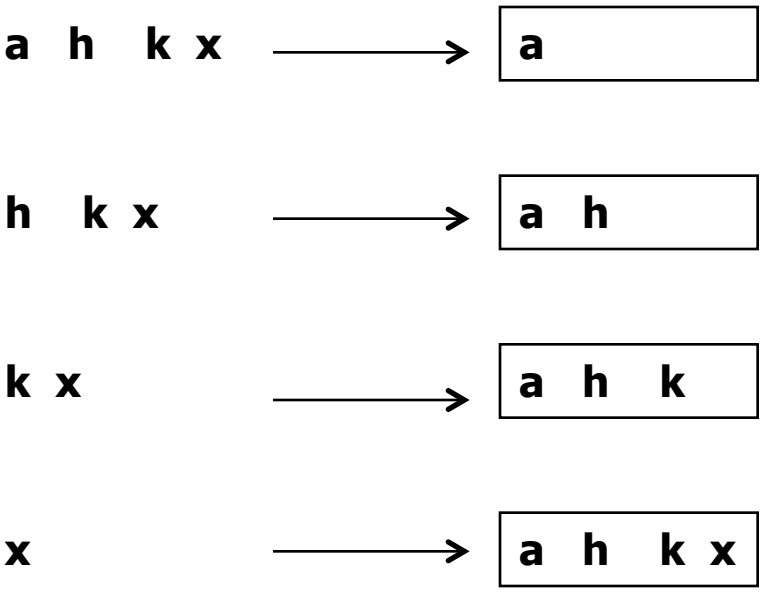
a h k x  **Noeud Racine (Root)**

Arbre B (ordre 2) création



Seule la racine peut avoir un nombre d'élements inférieure à m

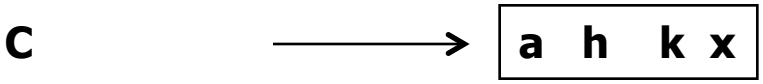
Arbre B (ordre 2) création



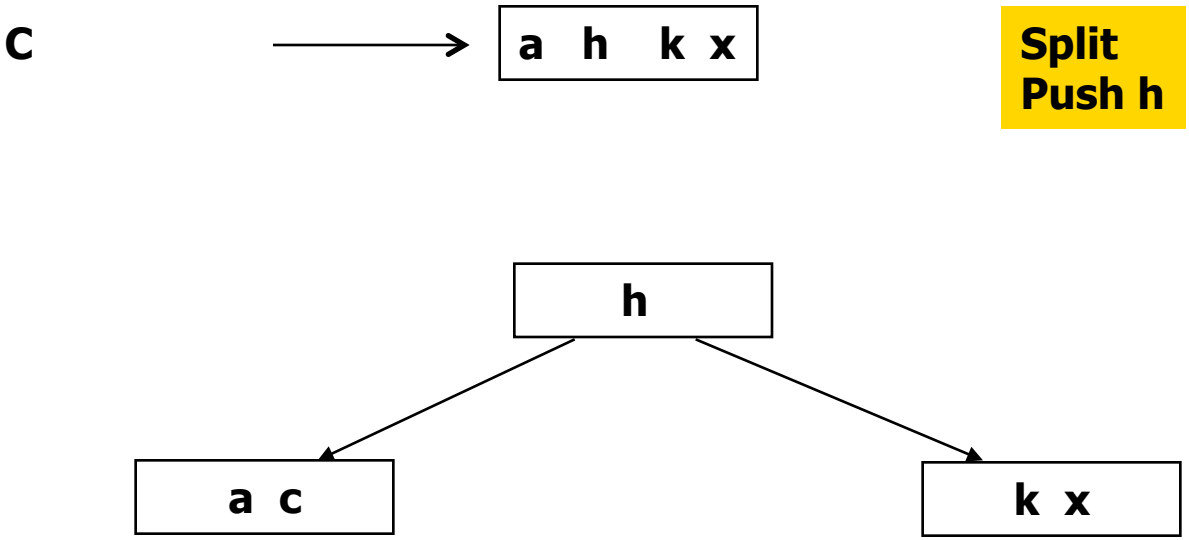
Noeud Racine (Root)

Root Full

Arbre B création



Arbre B création



Algorithme : Insert(key)

Find(key) to locate new position

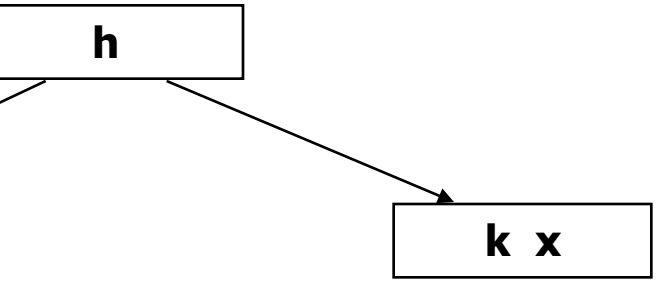
move right neighbors within node

if overflow split into 2 nodes

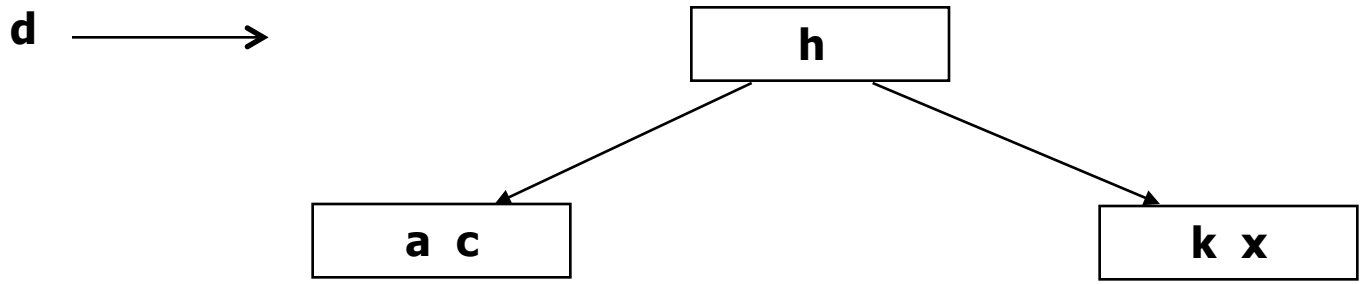
    promote middle key to parent node

    if parent overflows, repeat

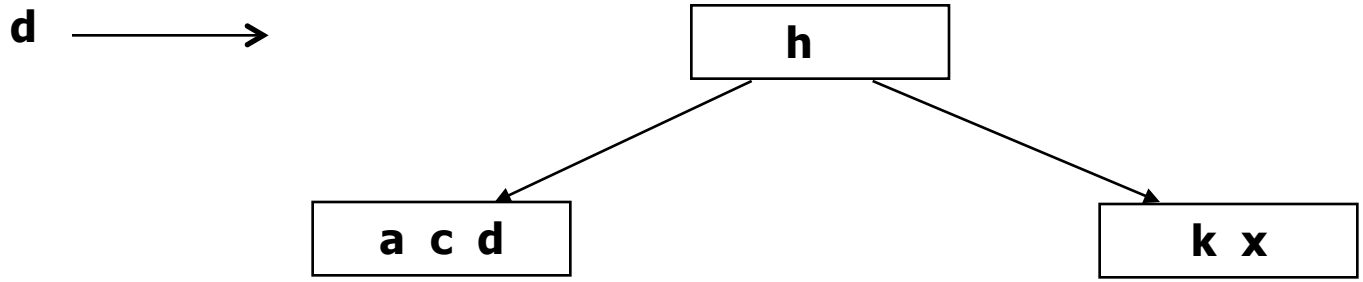
Arbre B insertion



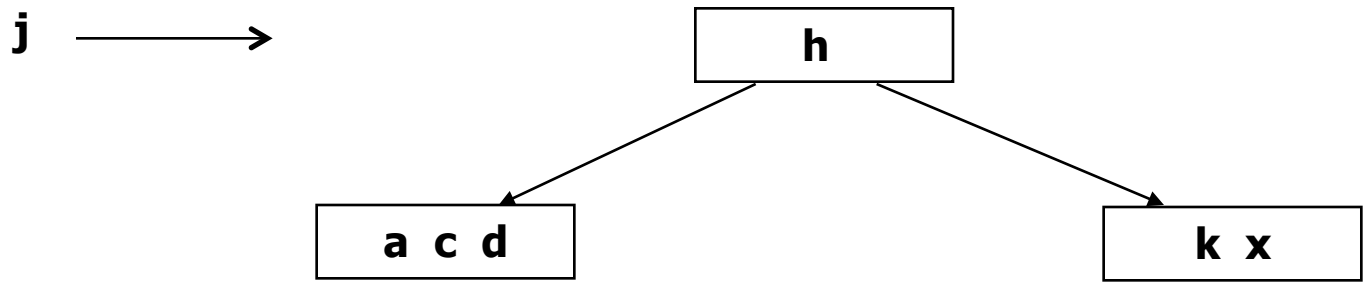
Arbre B insertion



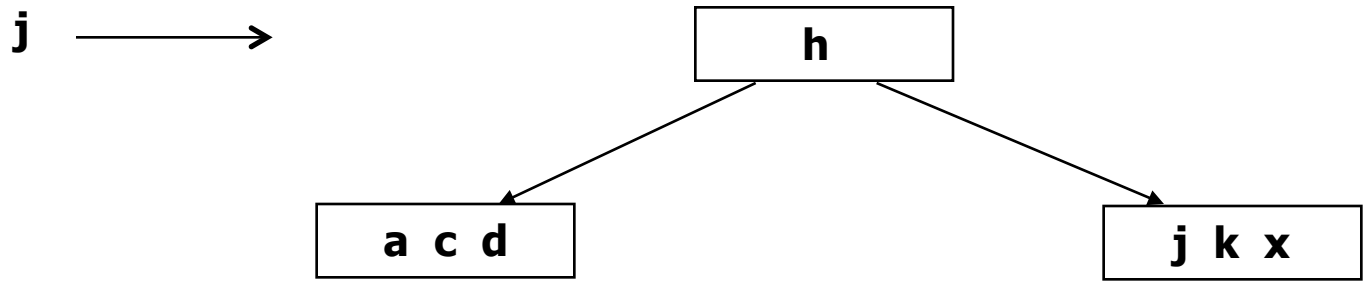
Arbre B insertion



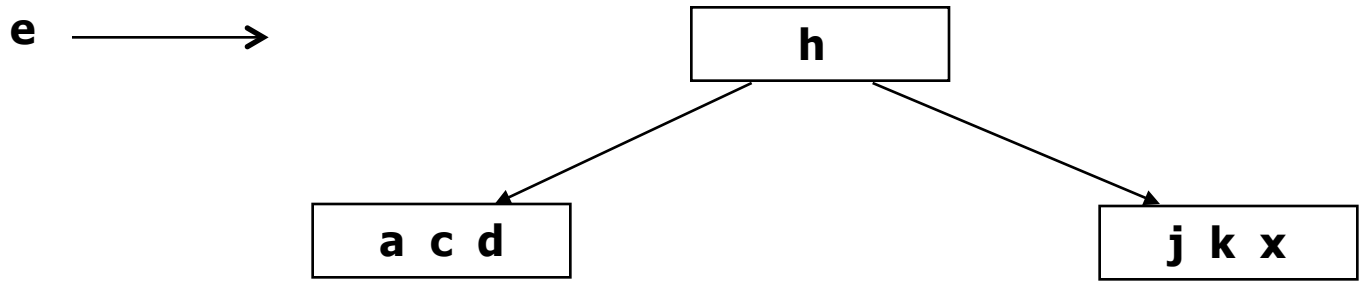
Arbre B insertion



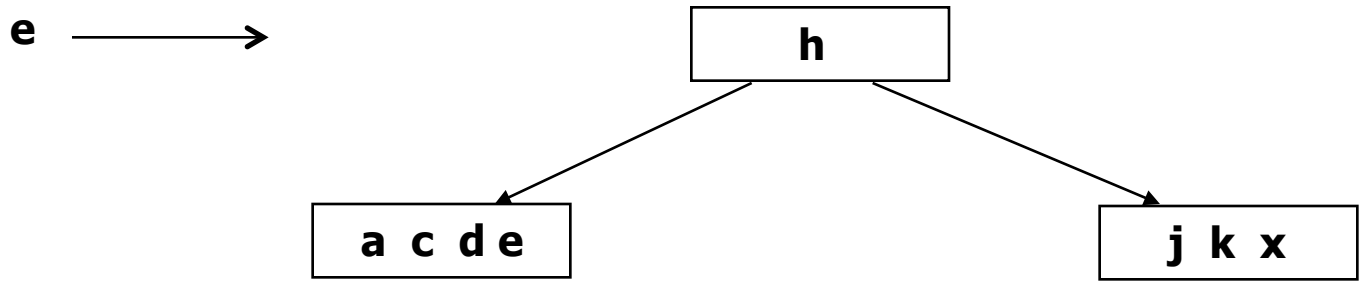
Arbre B insertion



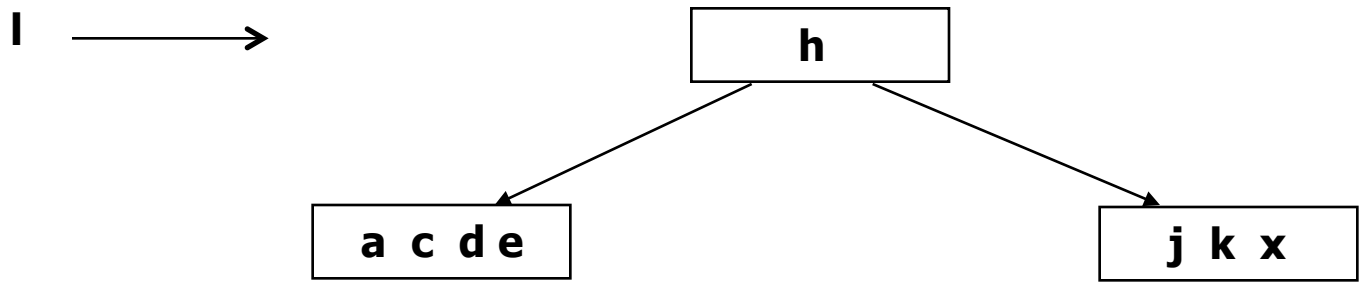
Arbre B insertion



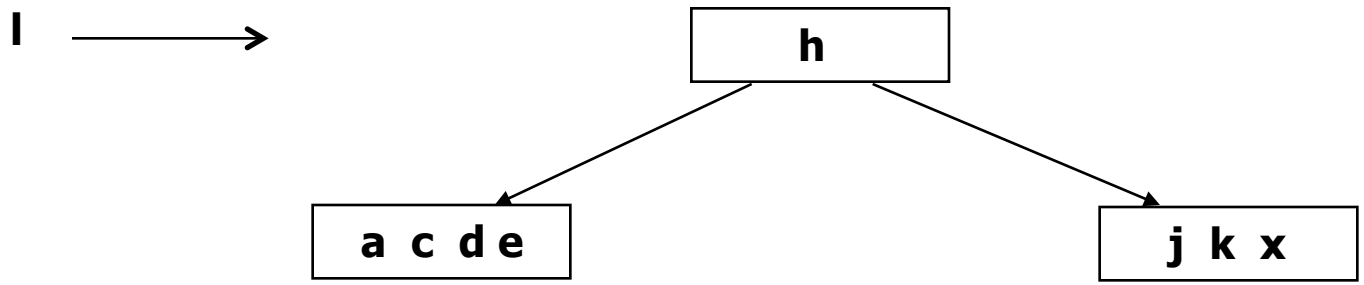
Arbre B insertion



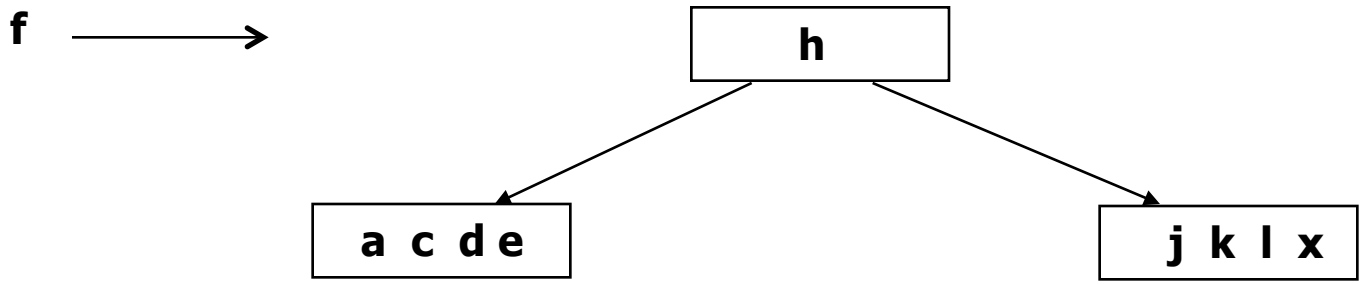
Arbre B insertion



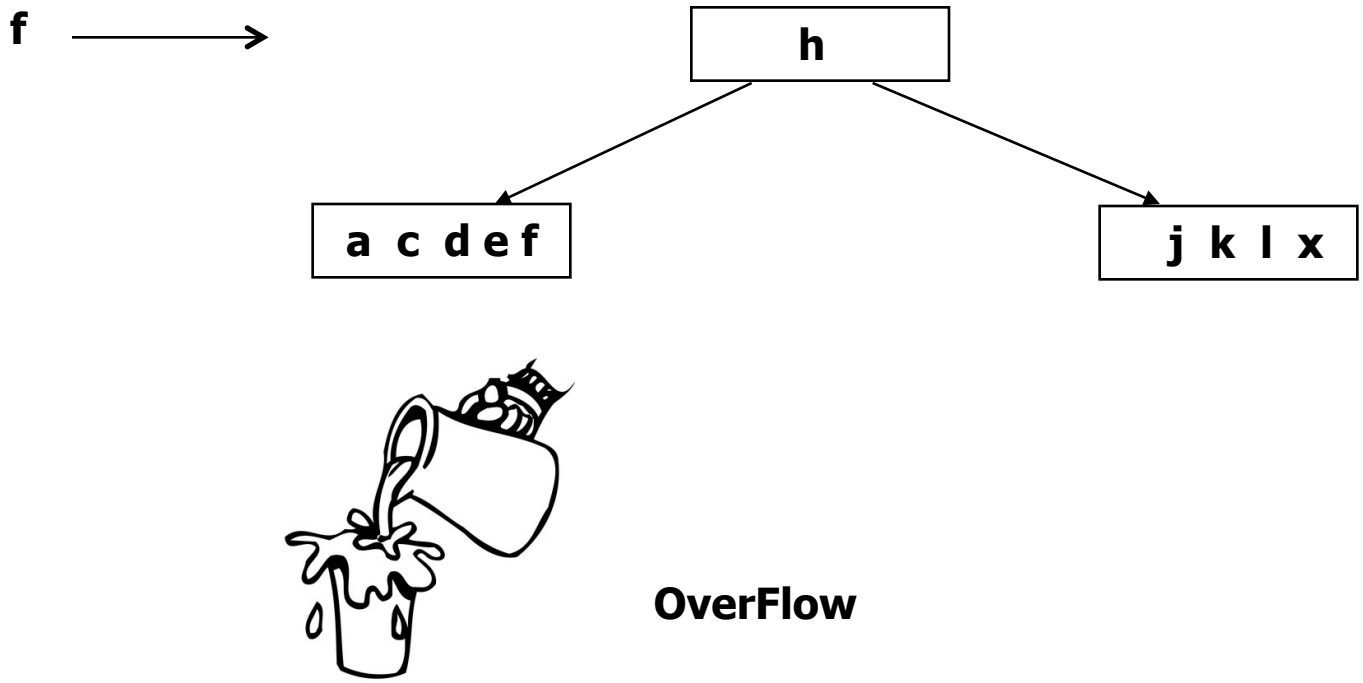
Arbre B insertion



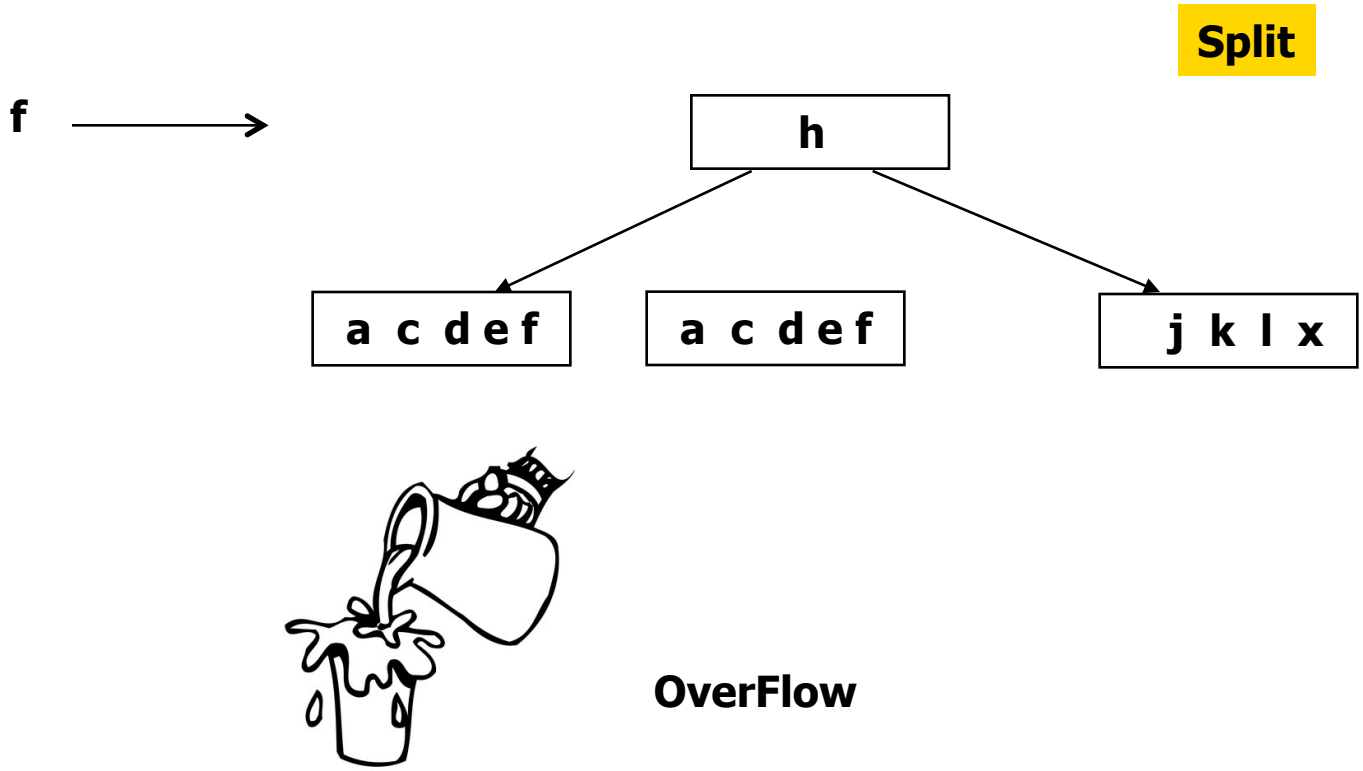
Arbre B insertion



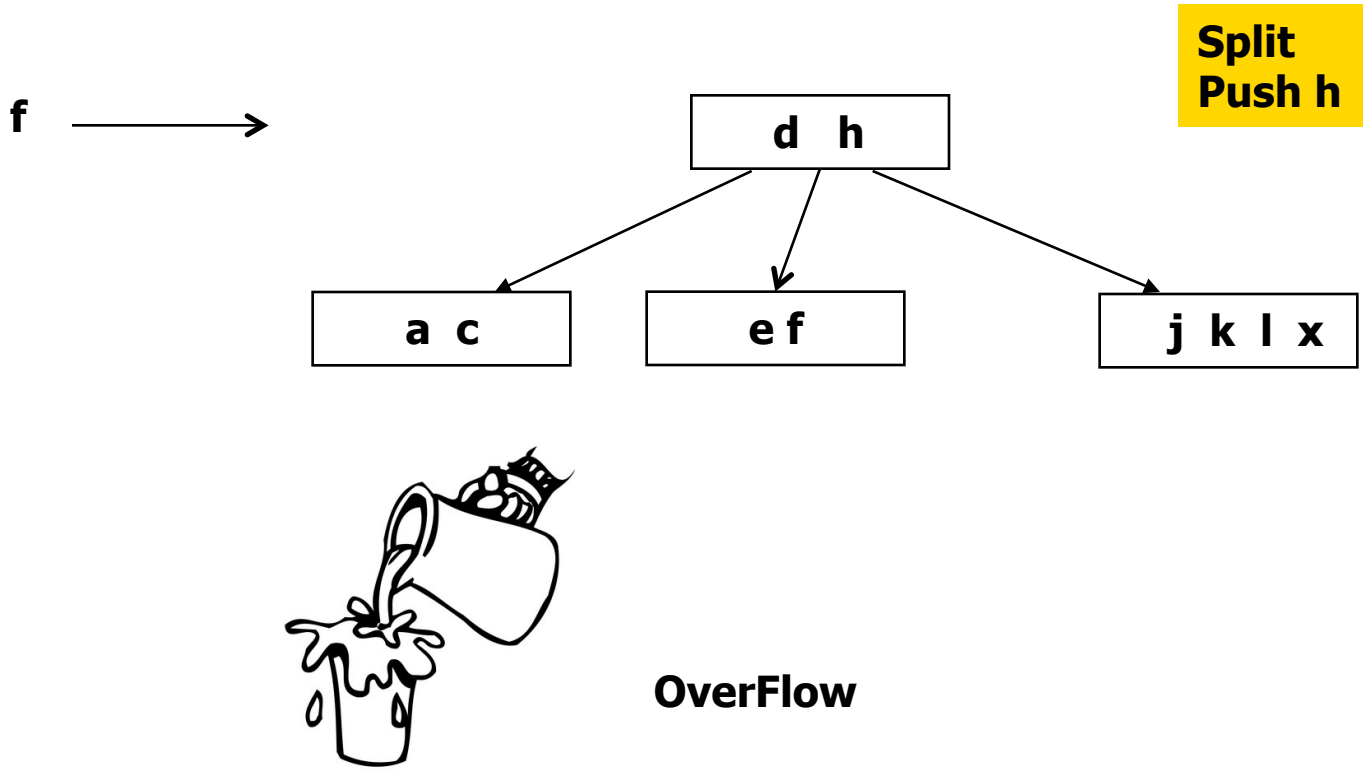
Arbre B insertion



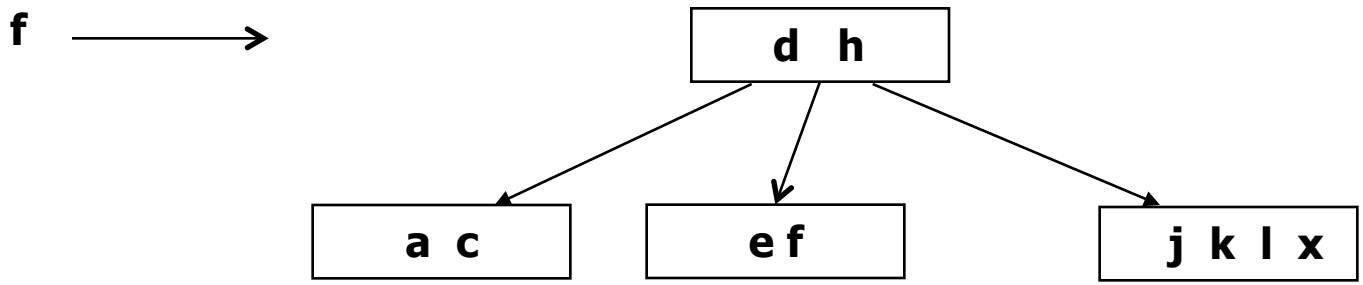
Arbre B insertion



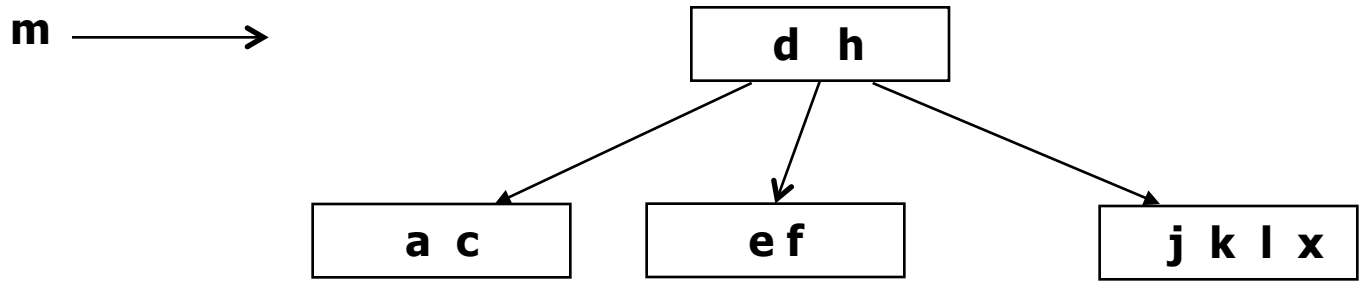
Arbre B insertion



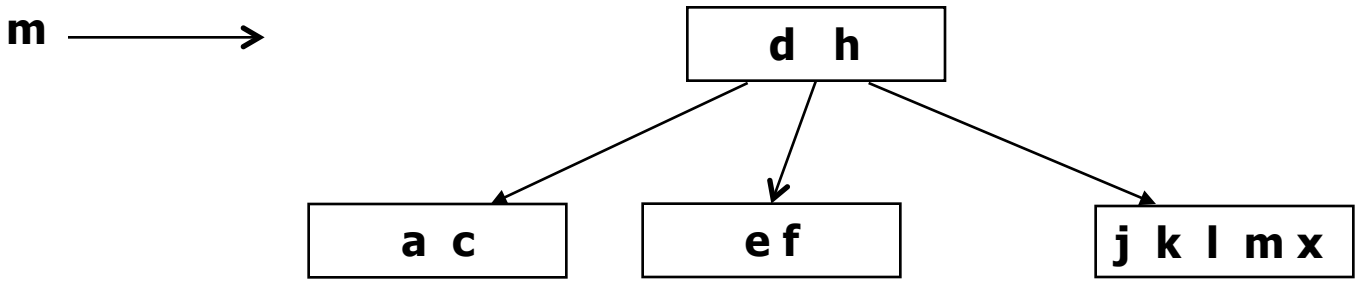
Arbre B insertion



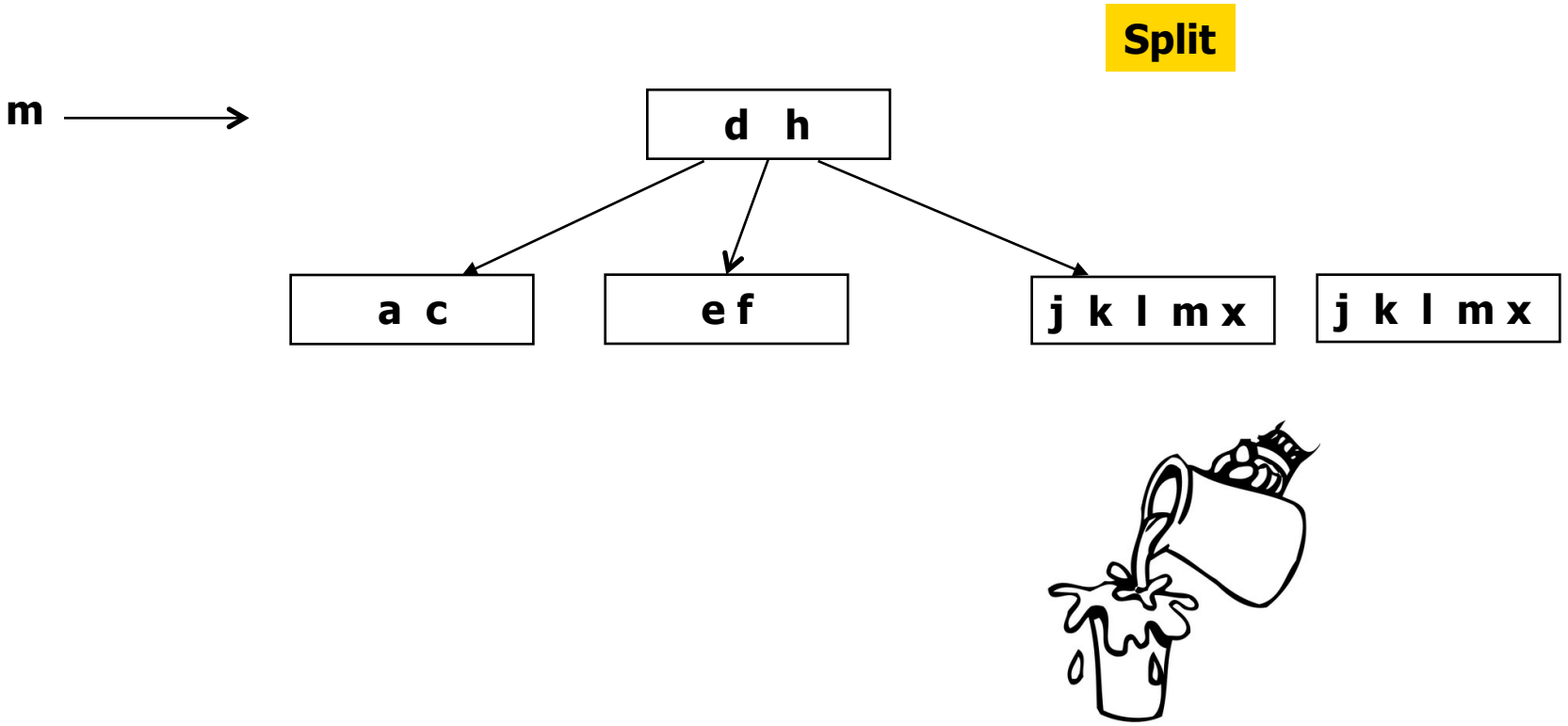
Arbre B insertion



Arbre B insertion

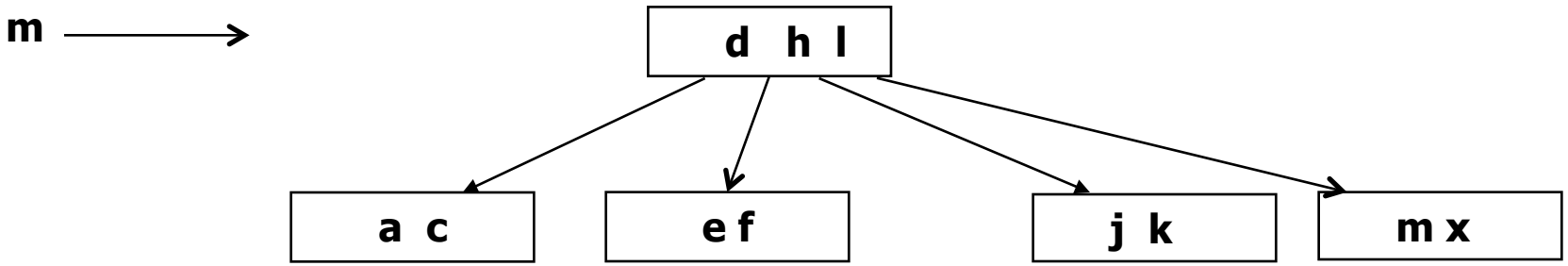


Arbre B insertion

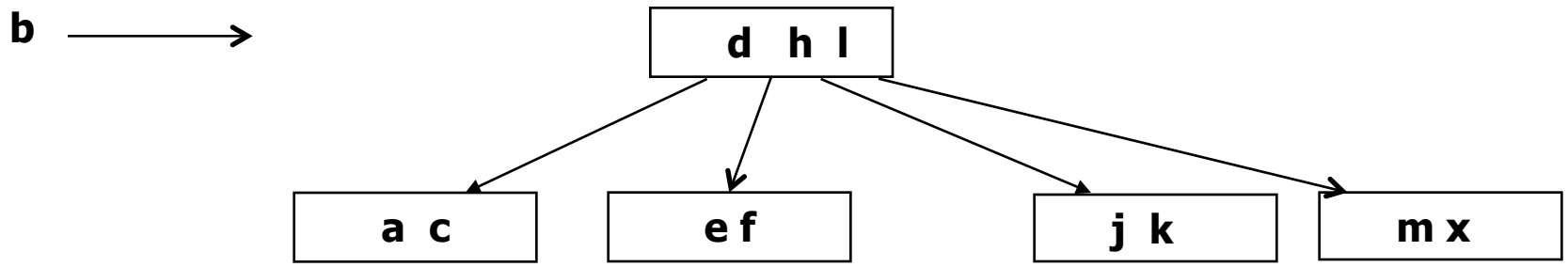


Arbre B insertion

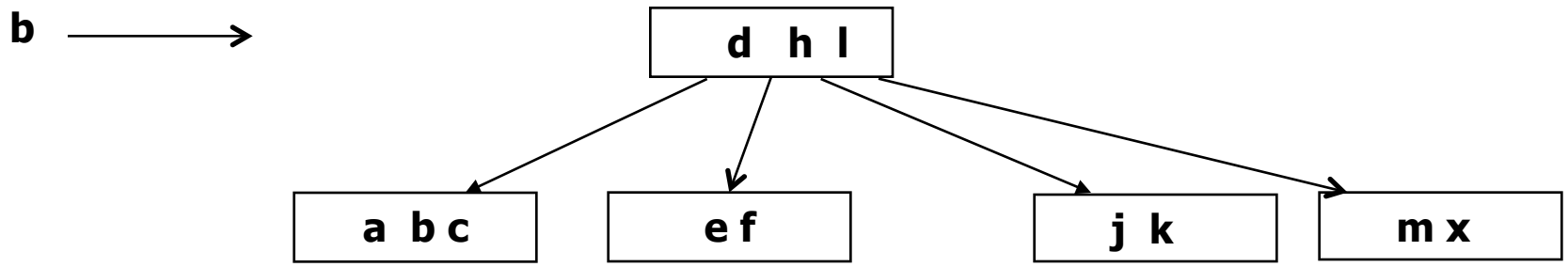
Split
Push l



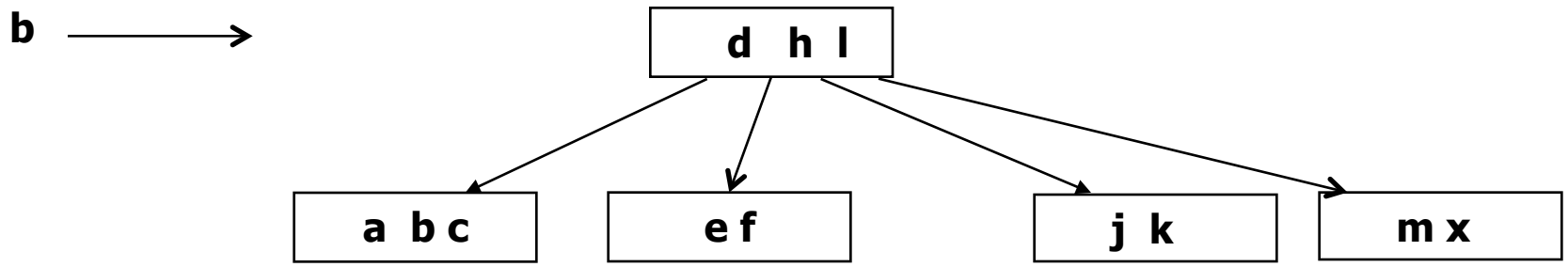
Arbre B insertion



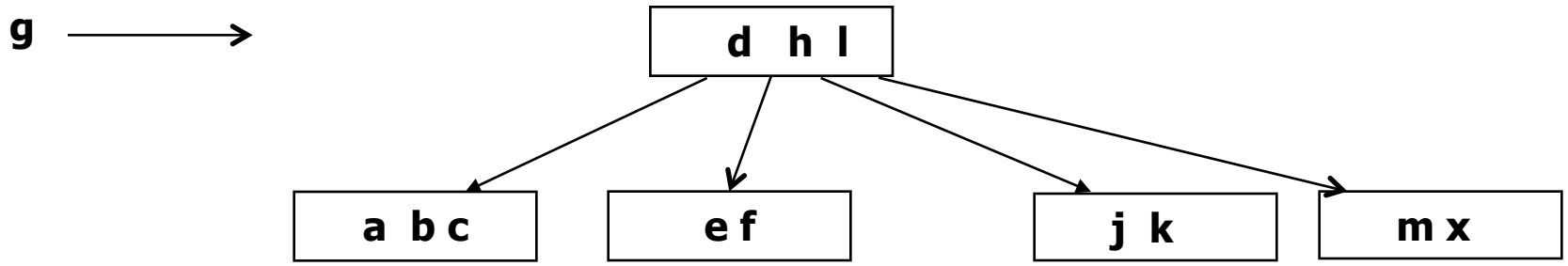
Arbre B insertion



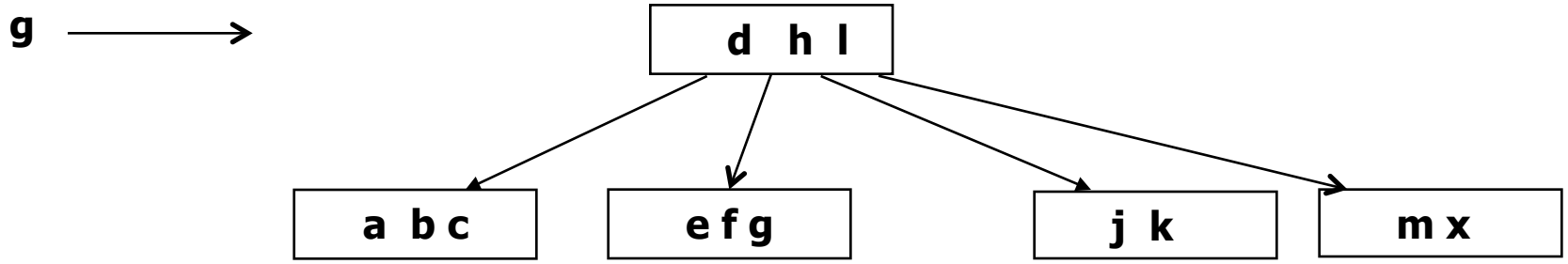
Arbre B insertion



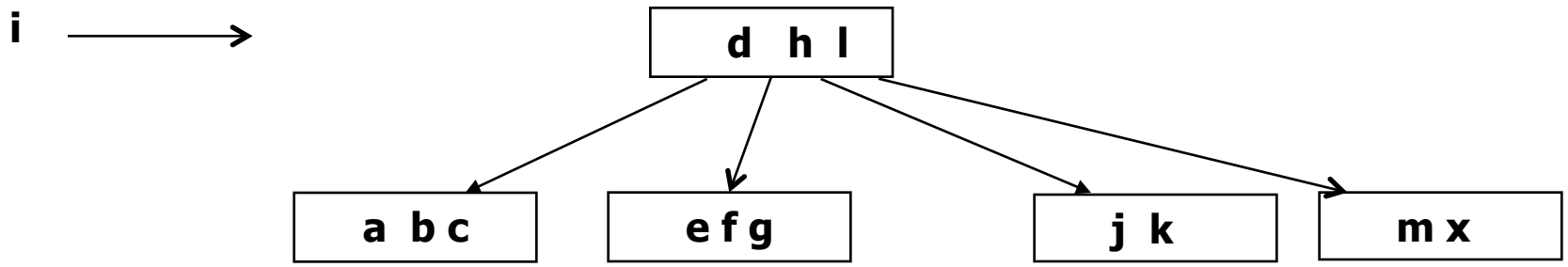
Arbre B insertion



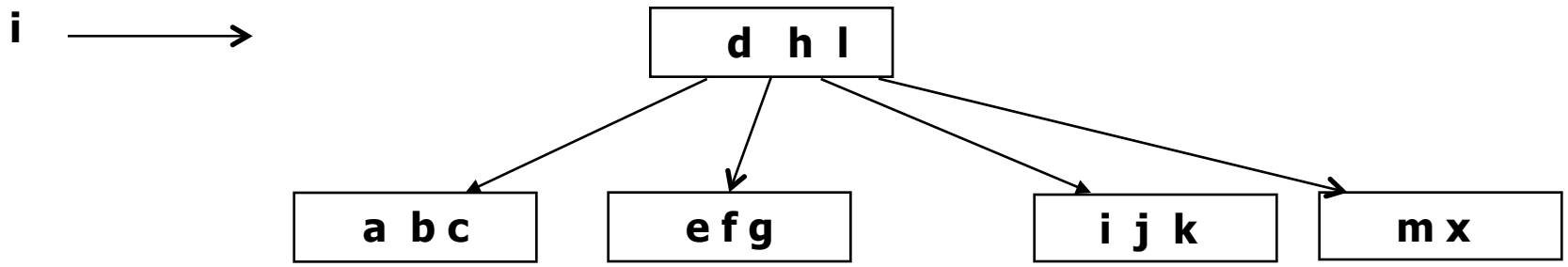
Arbre B insertion



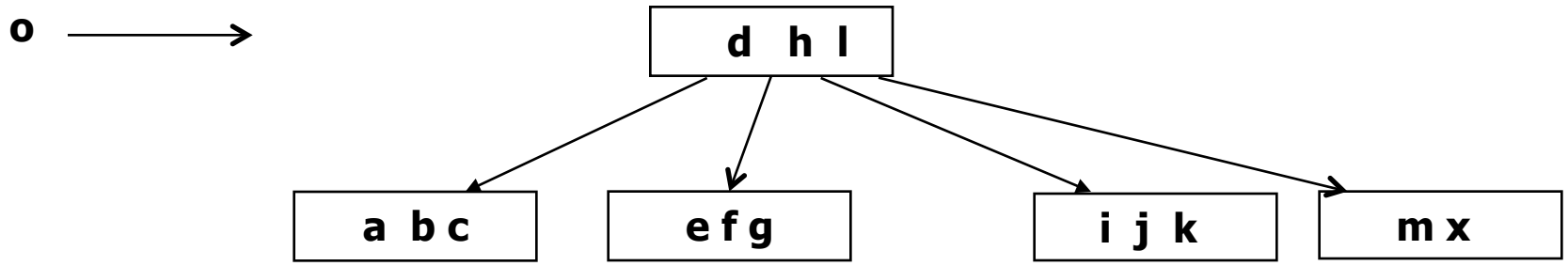
Arbre B insertion



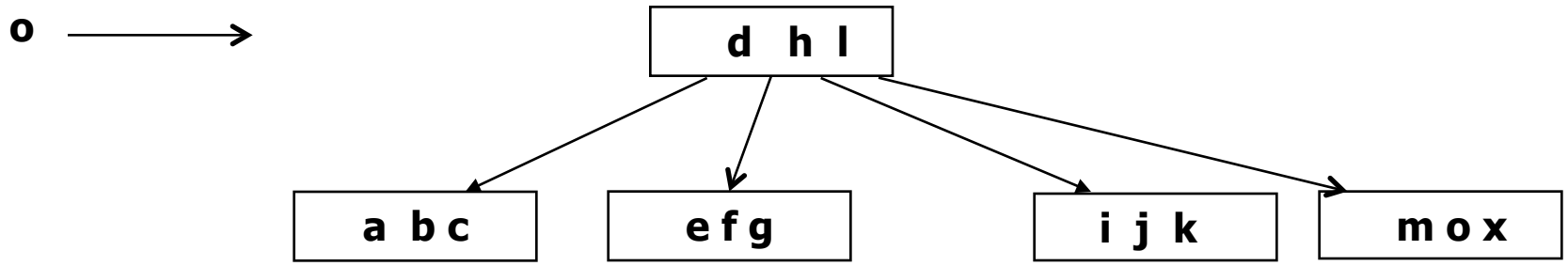
Arbre B insertion



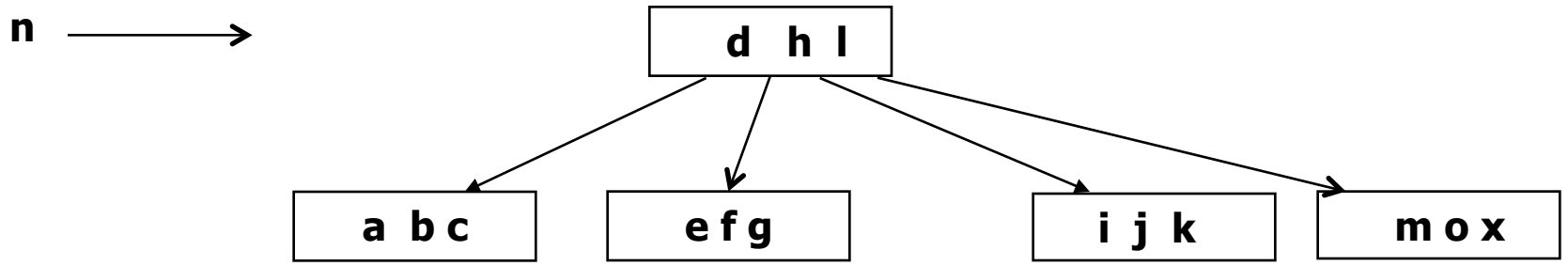
Arbre B insertion



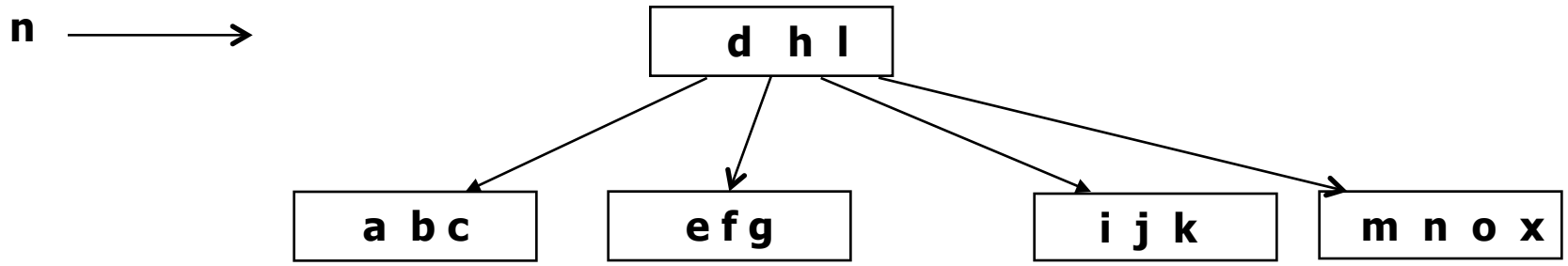
Arbre B insertion



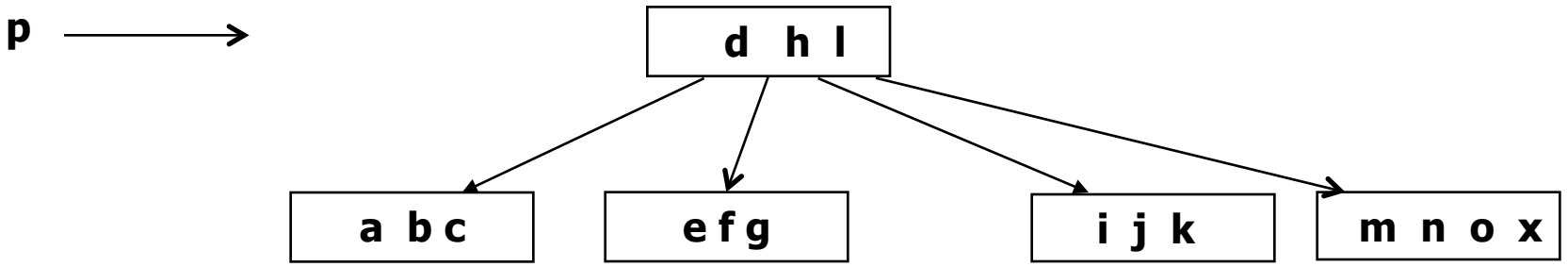
Arbre B insertion



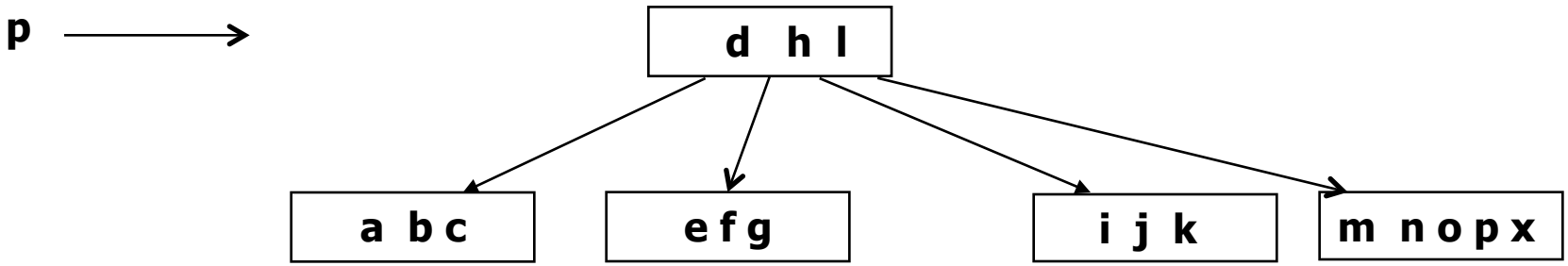
Arbre B insertion



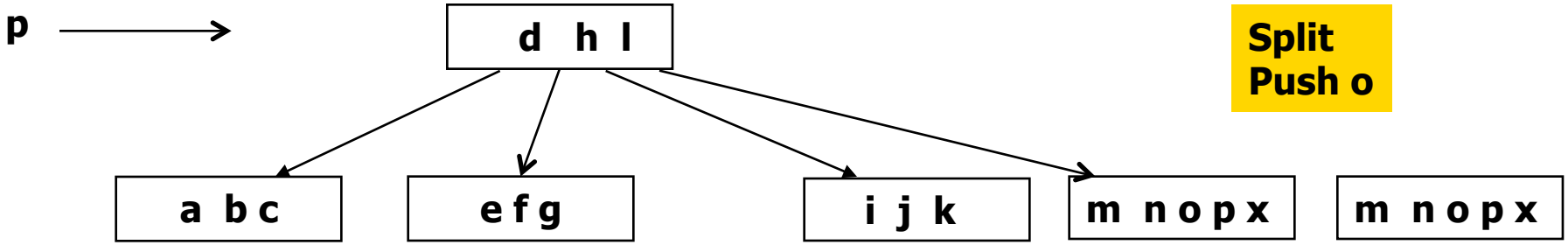
Arbre B insertion



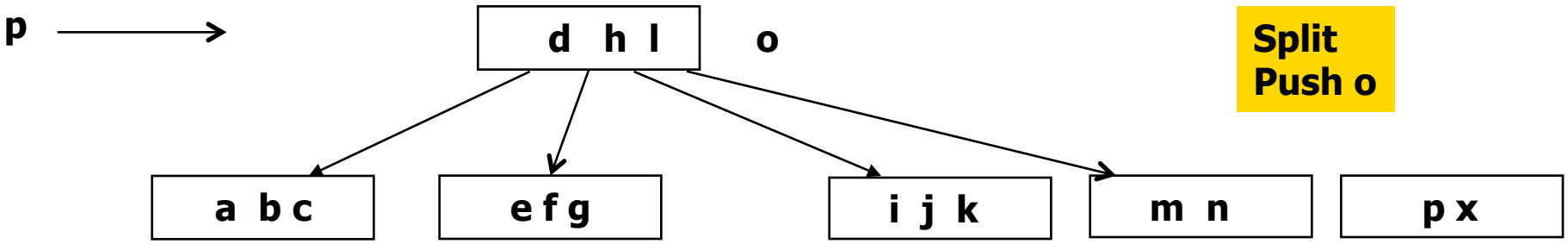
Arbre B insertion



Arbre B insertion

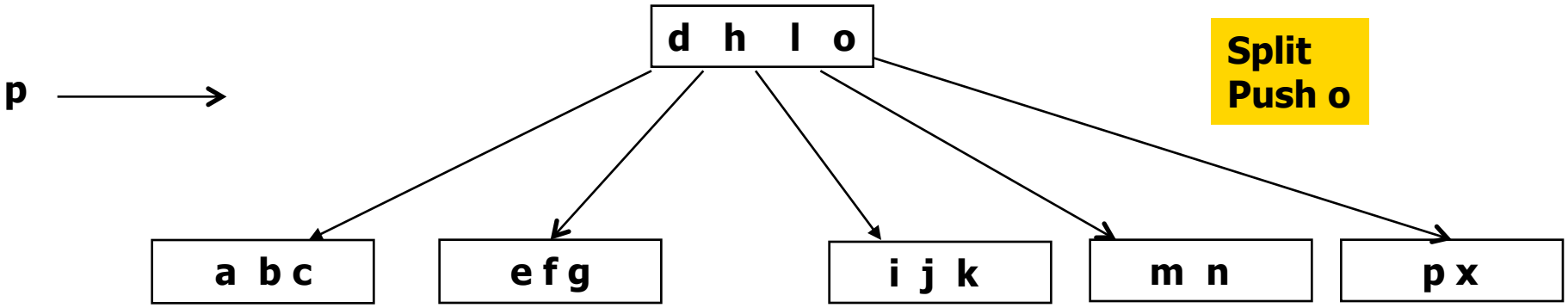


Arbre B insertion



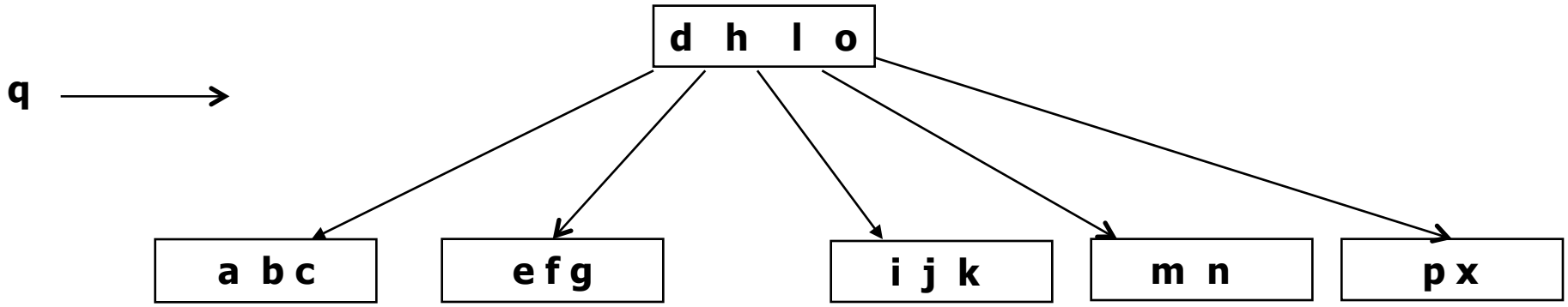
Split
Push o

Arbre B insertion

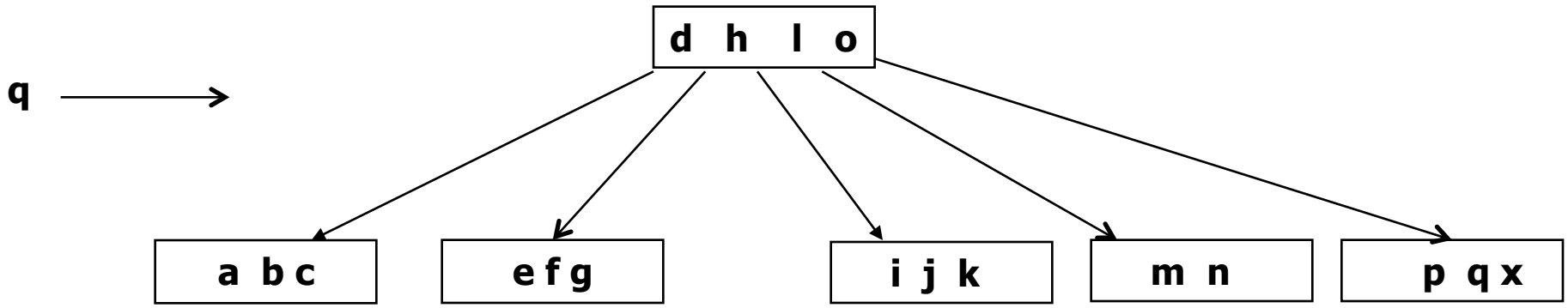


Split
Push o

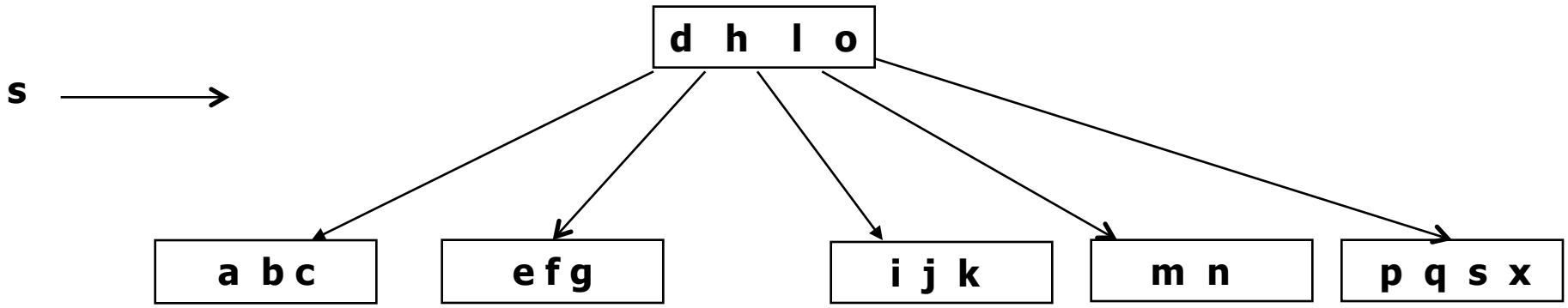
Arbre B insertion



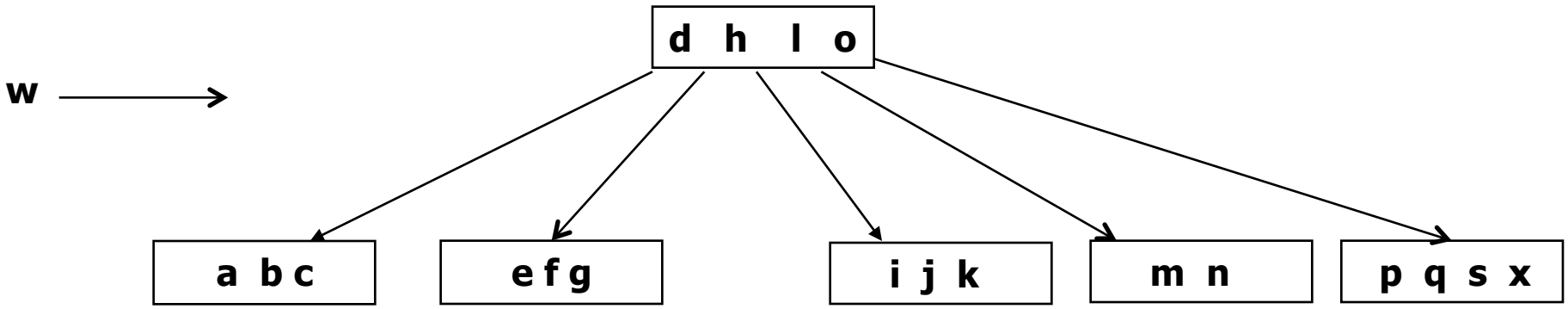
Arbre B insertion



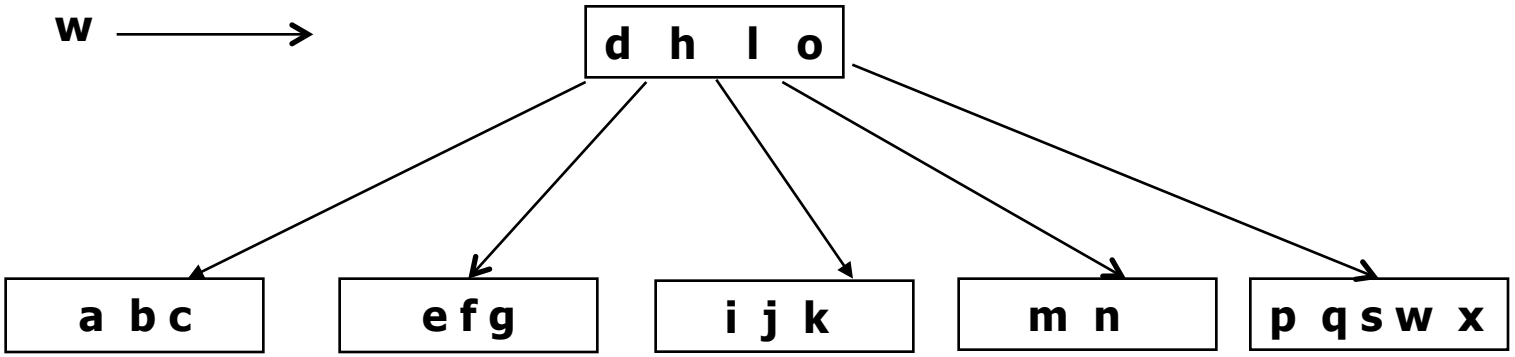
Arbre B insertion



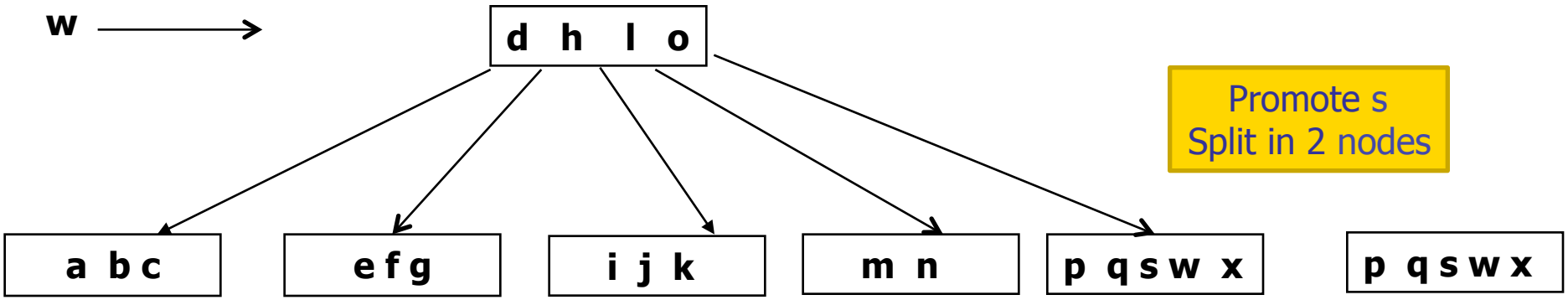
Arbre B insertion



Arbre B insertion

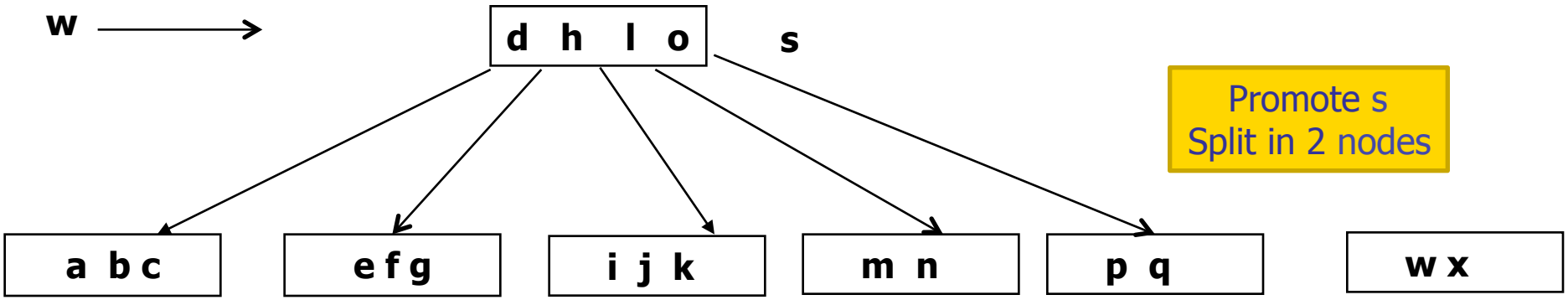


Arbre B insertion



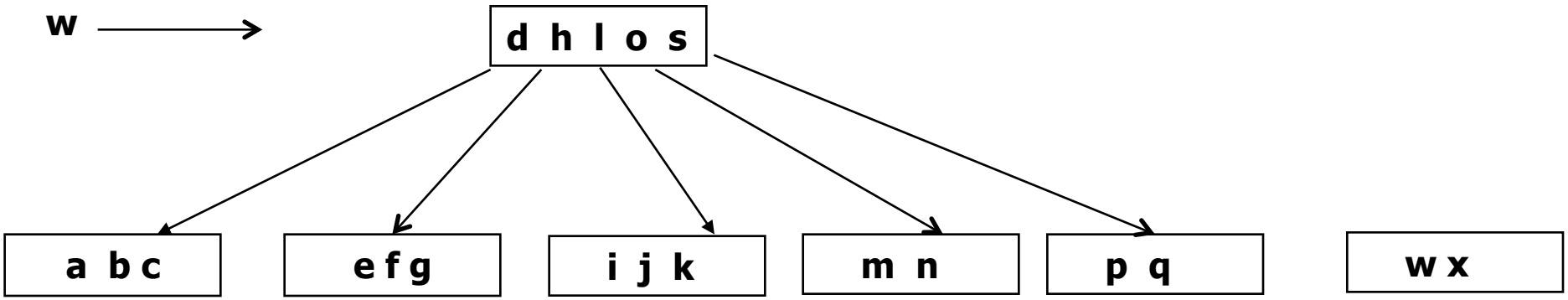
Promote s
Split in 2 nodes

Arbre B insertion



Promote s
Split in 2 nodes

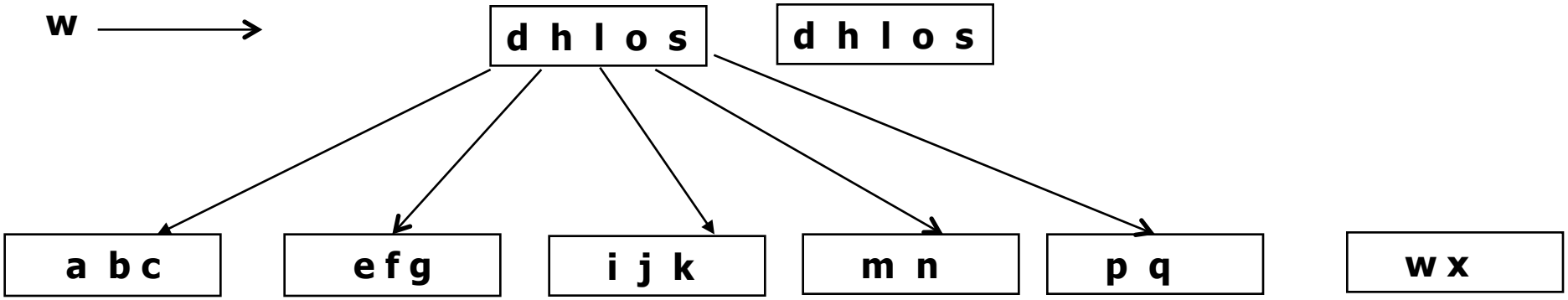
Arbre B insertion



Arbre B insertion

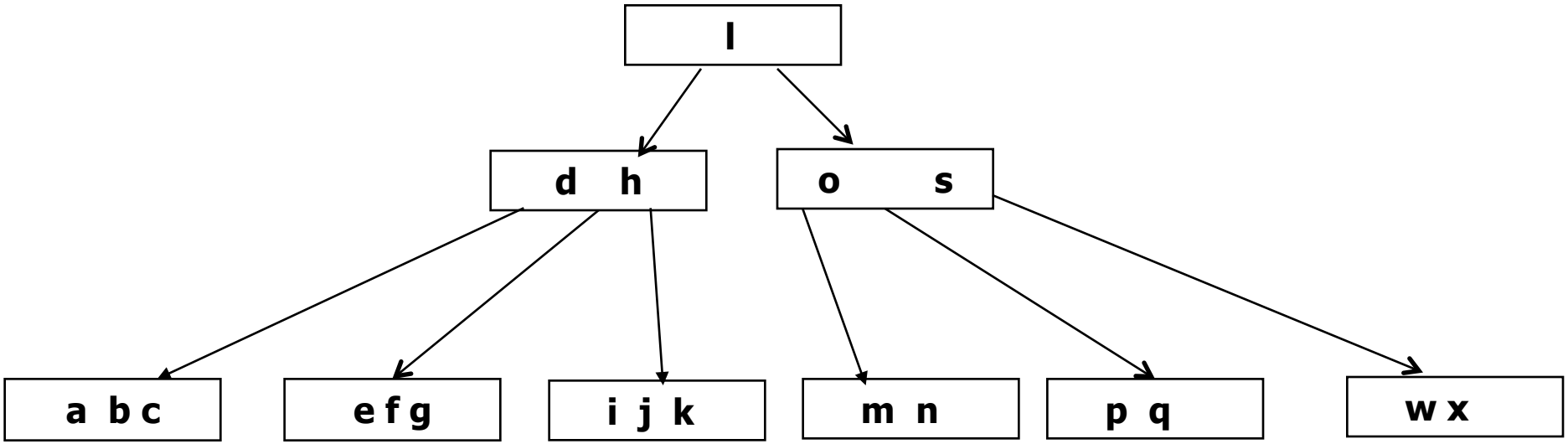
Promote l
Split in 2 nodes

w →

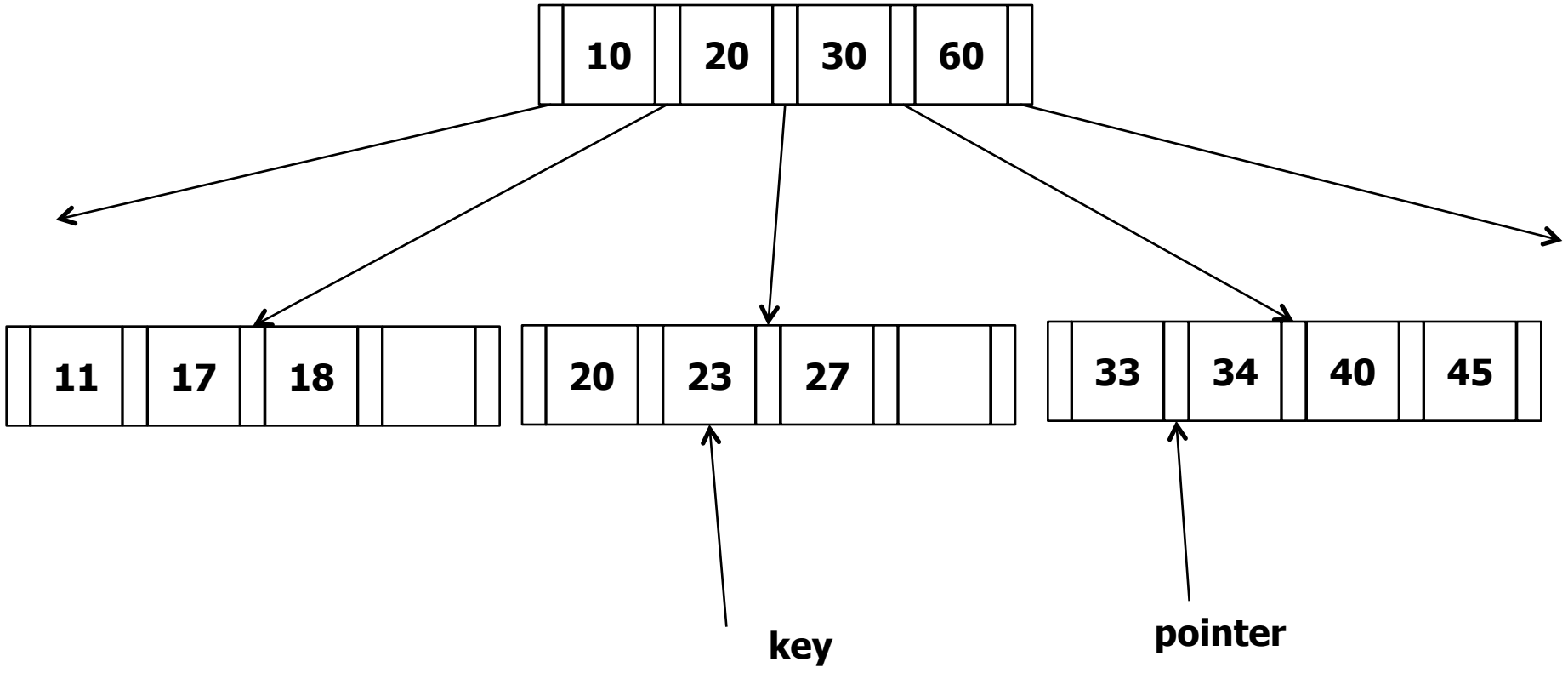


Arbre B insertion

Arbre B insertion



B-tree representation



Comer Article Representation

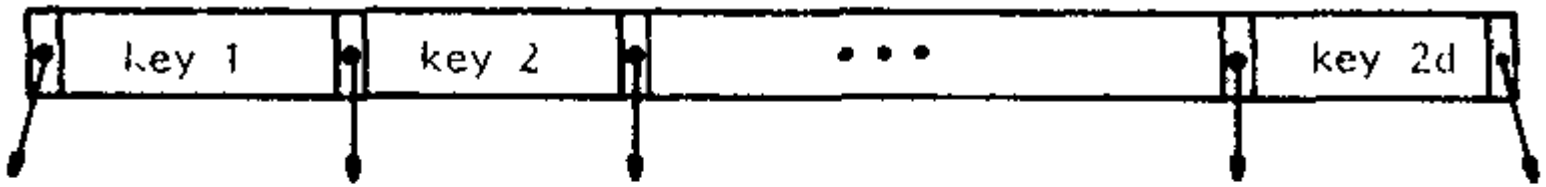
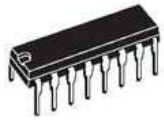


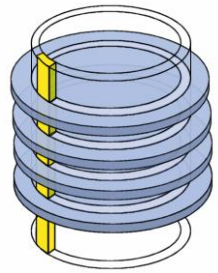
FIGURE 4. A node in a B-tree of order d with $2d$ keys and $2d + 1$ pointers.

BST vs B-tree

- BST : Binary Seach Tree
  - BST nodes contain only one key.
  - BST nodes have only Two children
  - In memory tree



- B-tree :
  - B-tree may have a variable number of keys and children
  - Multiway tree
  - Disk tree

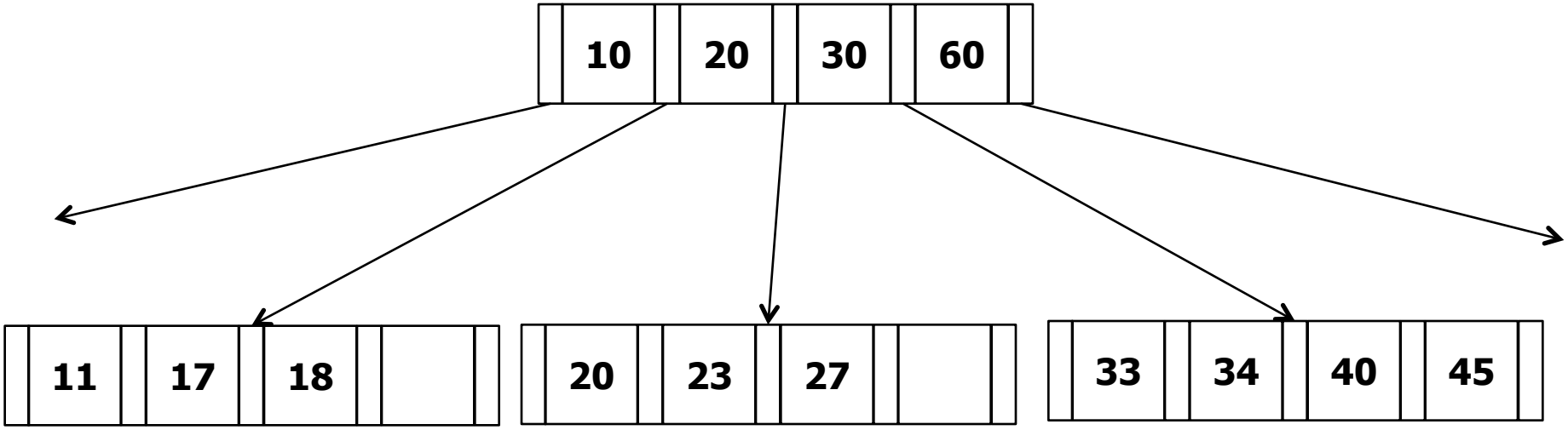


Height

- Tree search time is proportional to the height of the tree.
- BST height :
  - BST nodes contain only one key.
  - $log_2\ n$
    - Where *n* is the number of nodes in the tree.
- B-tree height
  - B-tree contains a lot of keys in each node so that the height of the tree keep small.
  - worst case : $O(log_b\ n)$.
    - Where b base of the logarithm depend on the Btree degree.

B-tree example

K children in the node.
k – 1 keys in the node.

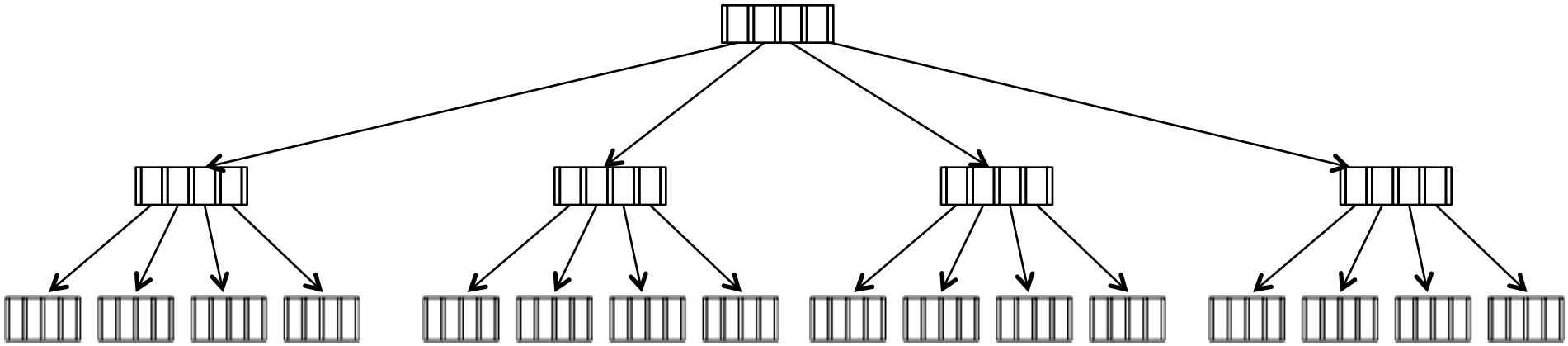


B-tree Height and Key Numbers

1 ⟶ **4** **m = 2**

2 ⟶ **4 x 4 = 16**

3 ⟶ **16 x 4 = 64**



B-tree Height and Key Numbers m = 2

Height	Keys #
1	4
2	16
3	64
4	256
5	1024
6	4096
7	16384
8	65536
9	262144
10	1048576

B-tree Height and Key Numbers and m

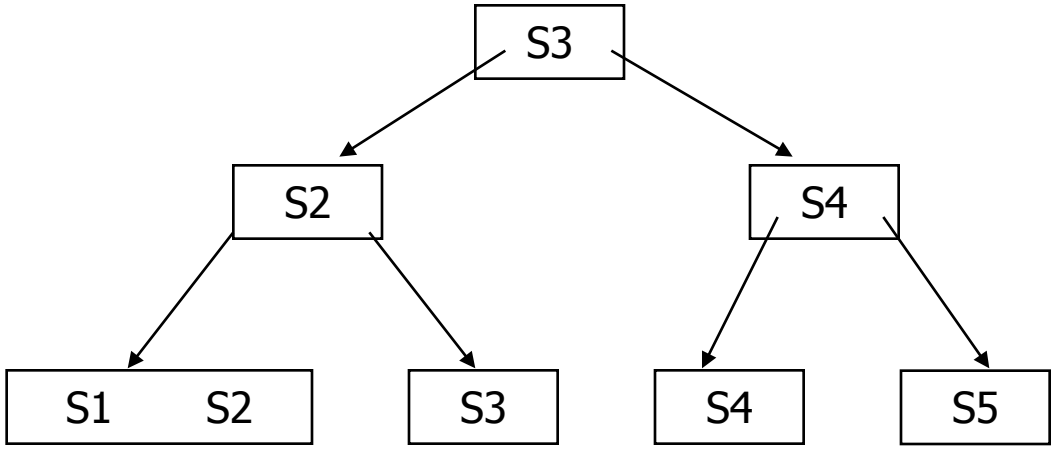
	2*m				
	2	4	6	8	10
Height	Keys #	Keys #	Keys #	Keys #	Keys #
1	2	4	6	8	10
2	4	16	36	64	100
3	8	64	216	512	1000
4	16	256	1296	4096	10000
5	32	1024	7776	32768	100000
6	64	4096	46656	262144	1000000
7	128	16384	279936	2097152	10000000
8	256	65536	1679616	16777216	100000000
9	512	262144	10077696	134217728	1000000000
10	1024	1048576	60466176	1073741824	10000000000

B-tree Height and Key Numbers and m

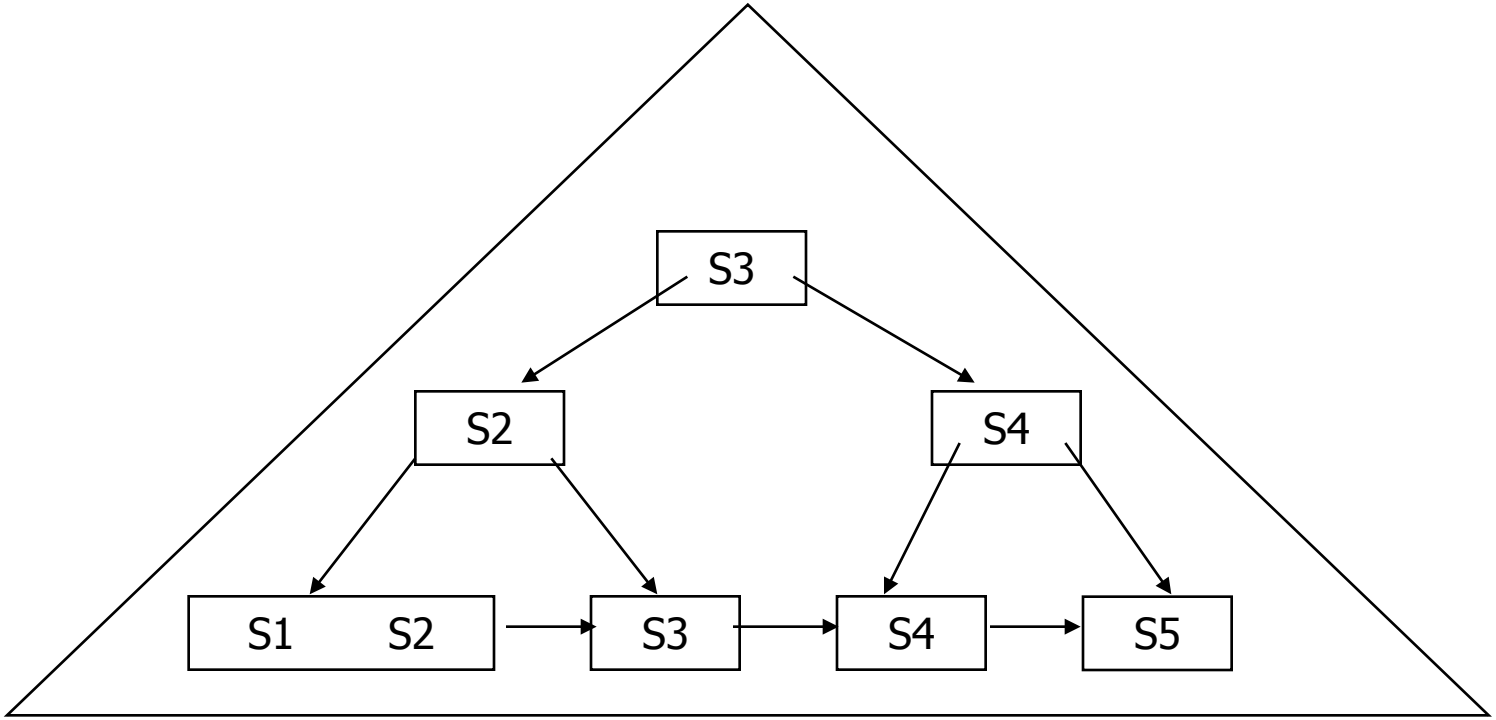
- Keys Number :: Tree Height at the power of 2 x m.
- Tree Height :: $\log_{(2 \times m)}(\#\ \text{Keys})$

Order of Magnitude

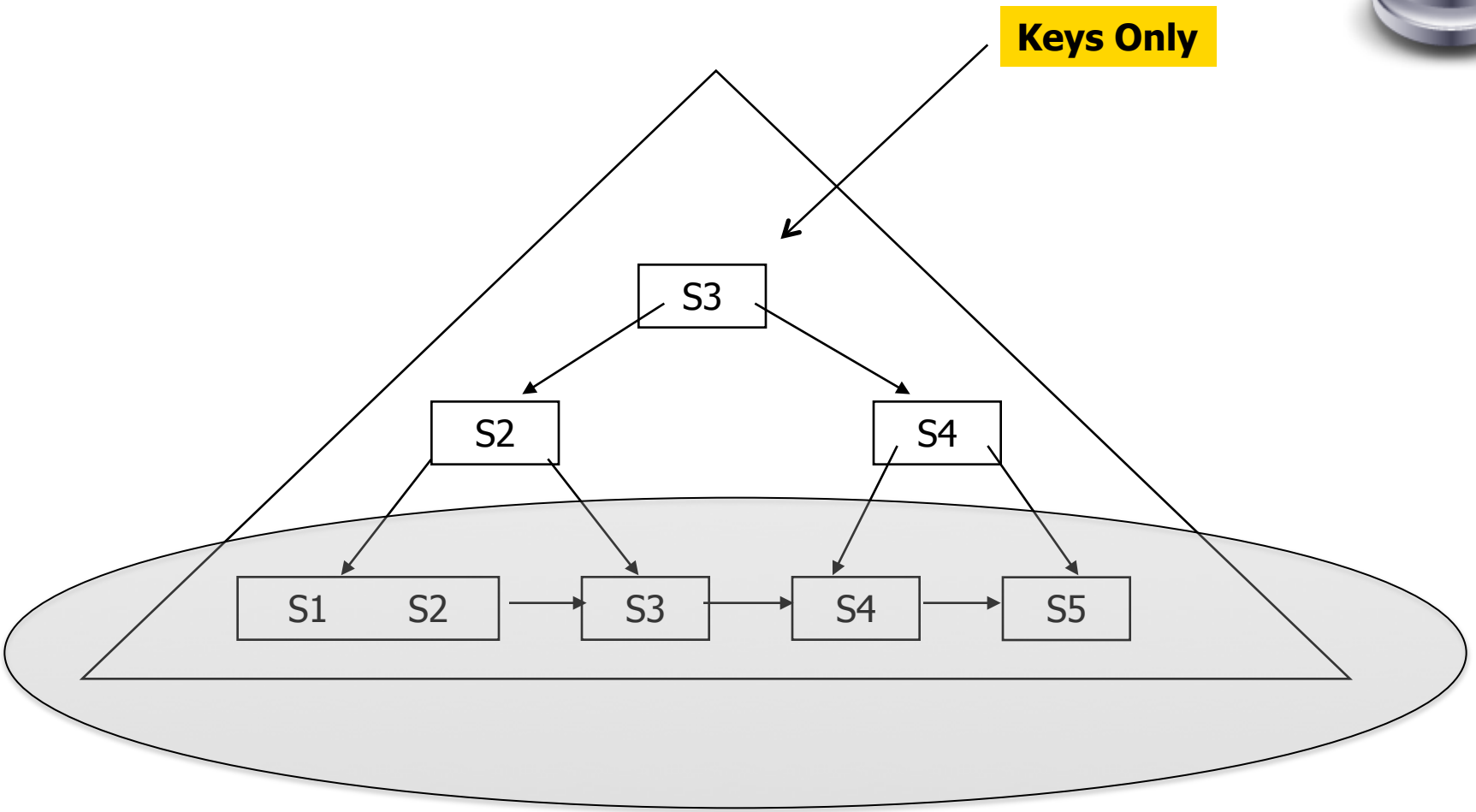
B-tree Index



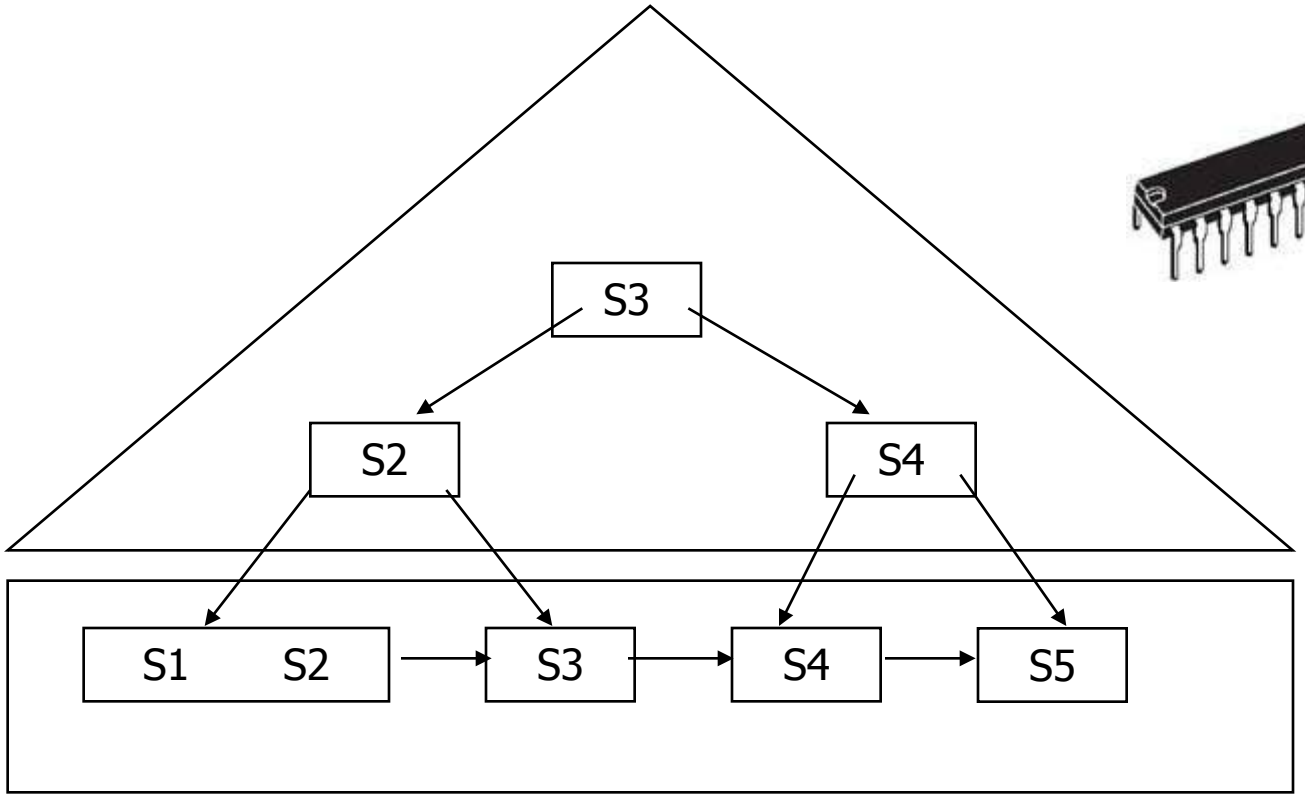
B+tree

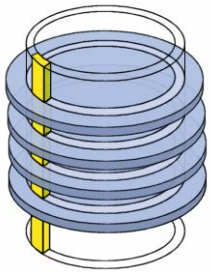


B+tree



B+tree Index Disk and Memory

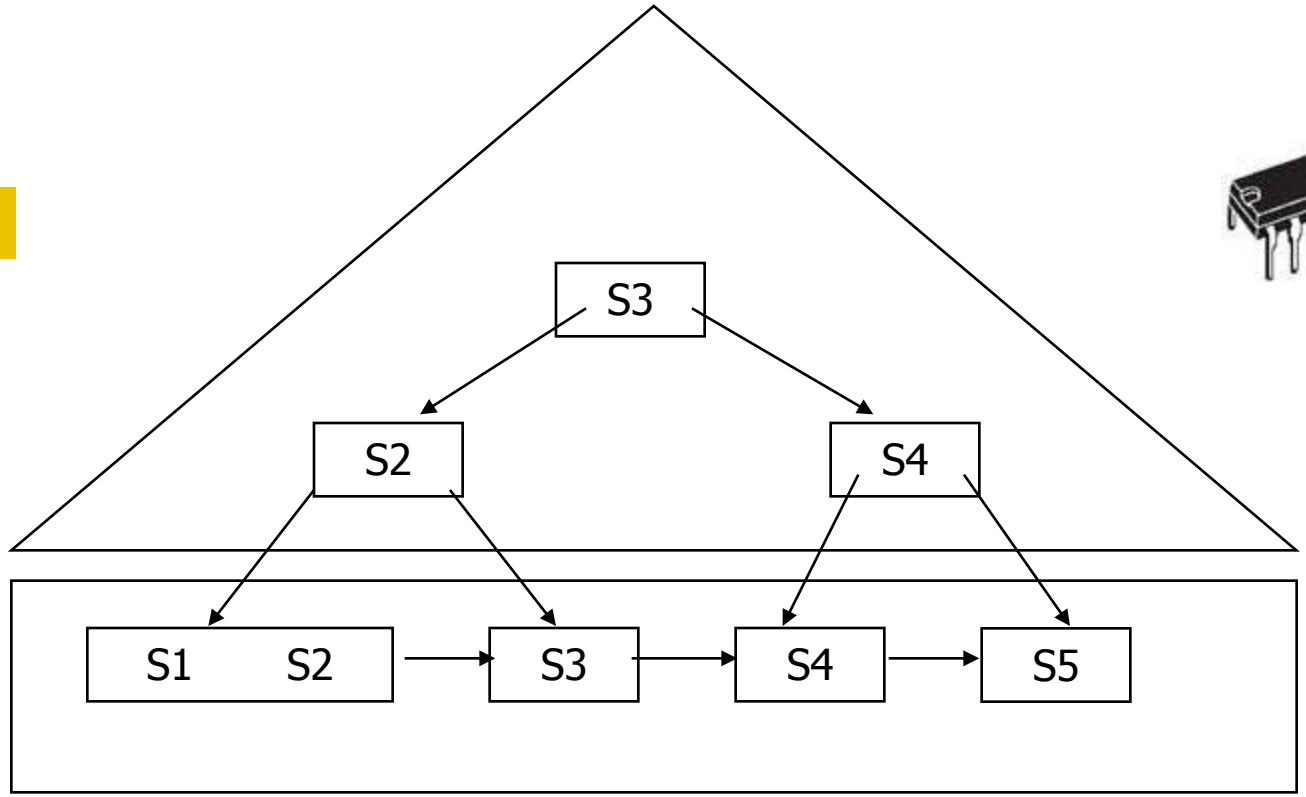


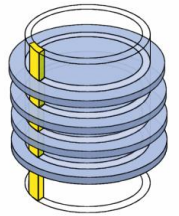


B+tree Index Disk and Memory

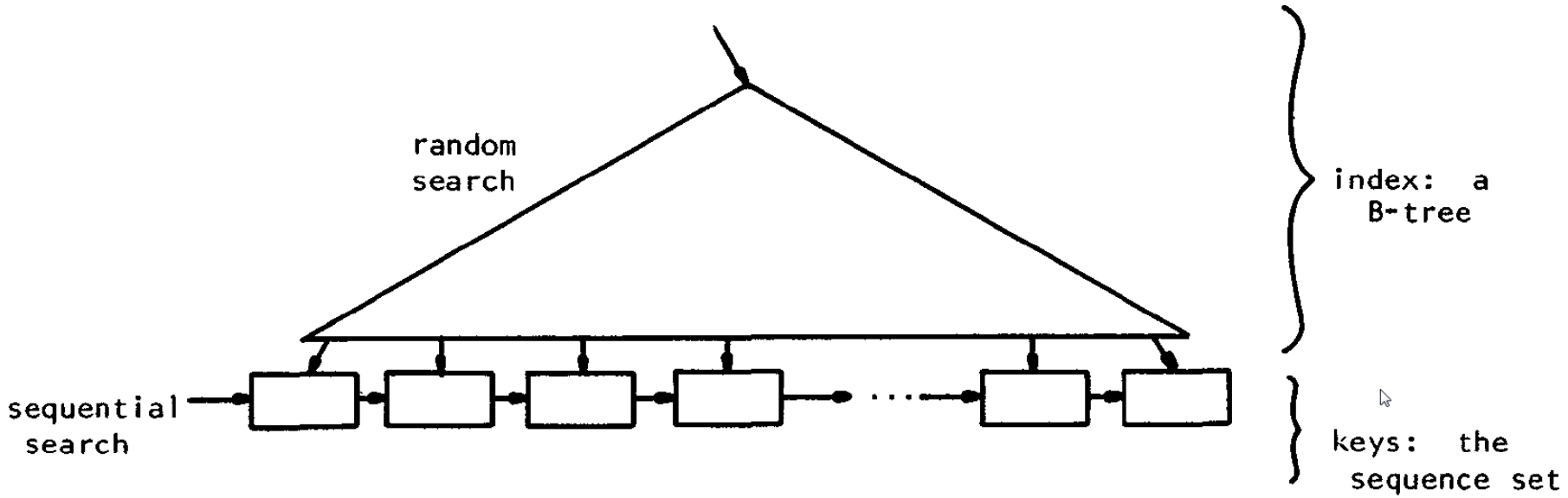
Index : B tree

Leafs : Data

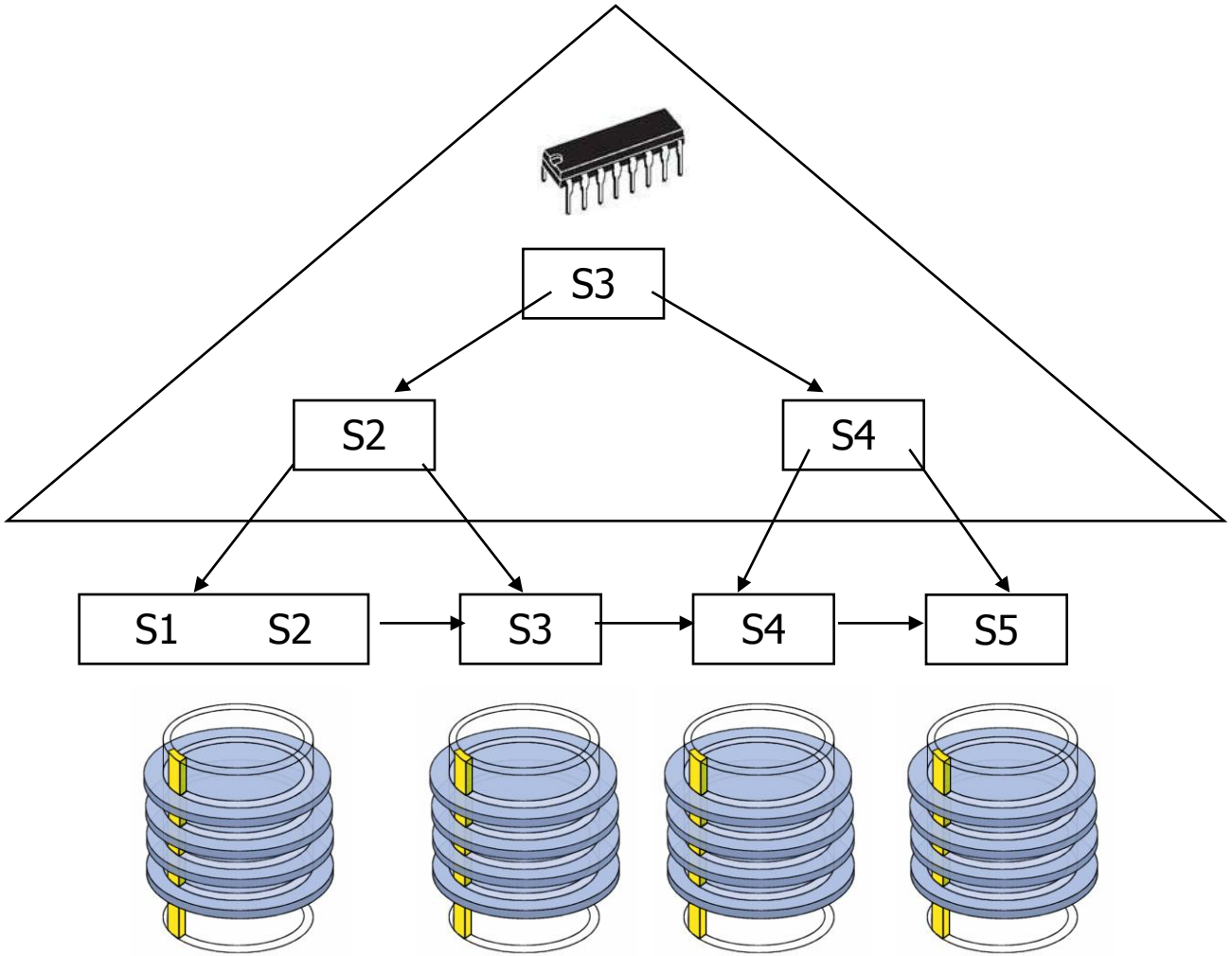




Comer Paper Representation



B+tree Index Disk and Memory



Memory VS disk