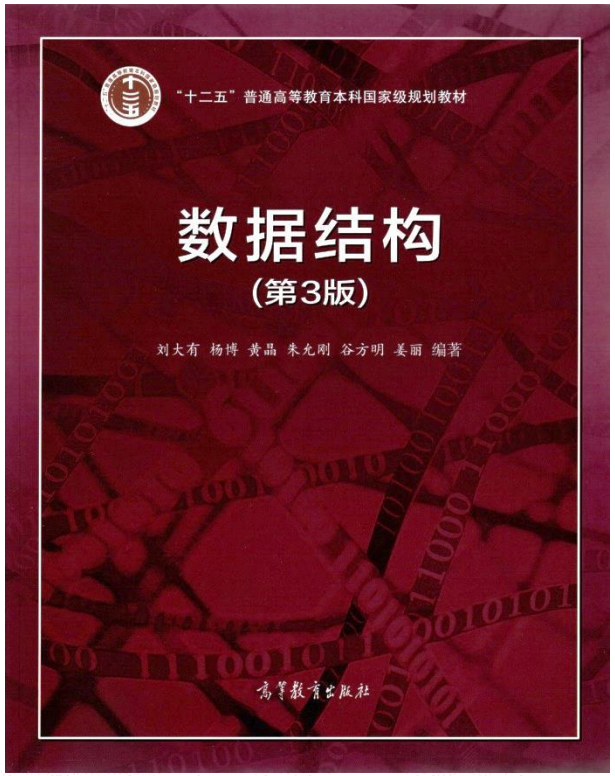


多叉查找树

- B树
- B+树简介
- 数字查找树/字典树

数据之法
结构之美
算法之道

zhuyungang@jlu.edu.cn

Last updated on 2023.12

我刚接触电脑就发现电脑的妙处，电脑远没有人那么复杂。如果你的程序写得好，就可以和电脑处好关系，就可以指挥电脑干你想干的事，这个时候你是十足的主宰。每当你坐在电脑前，你就是在你的王国里巡行，这样的日子简直就是天堂般的日子。

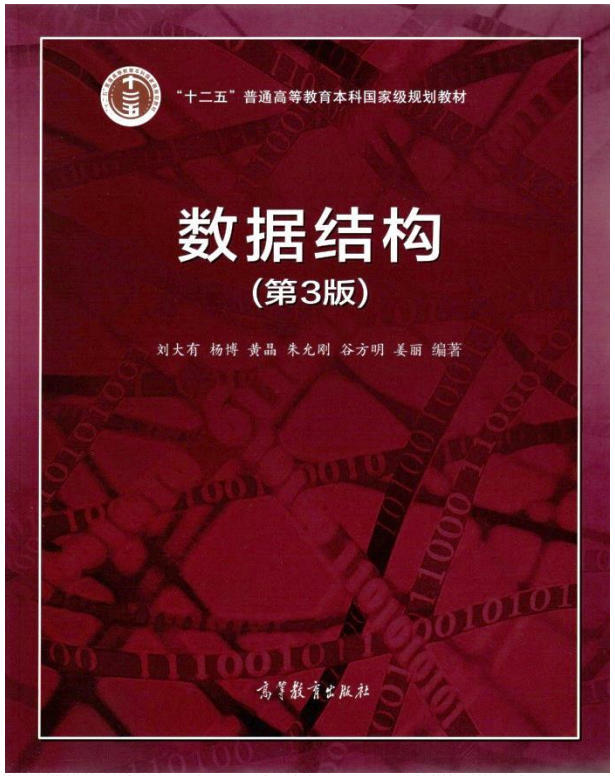
电脑里的世界很大，编程人是活在自己想象的王国里。你可以想象到电脑里细微到每一个字节、每一个比特的东西。

编程不仅仅是技术，也还是艺术。编程是技术活，才有可能大规模进行，才会有软件工程和软件工厂。也正是因为编程是艺术，才会有如此多的好产品，让大家如痴如醉。

——雷军

武汉大学学士

小米集团创始人、董事长兼CEO



多叉查找树

- **B树**
- B+树简介
- 数字查找树/字典树

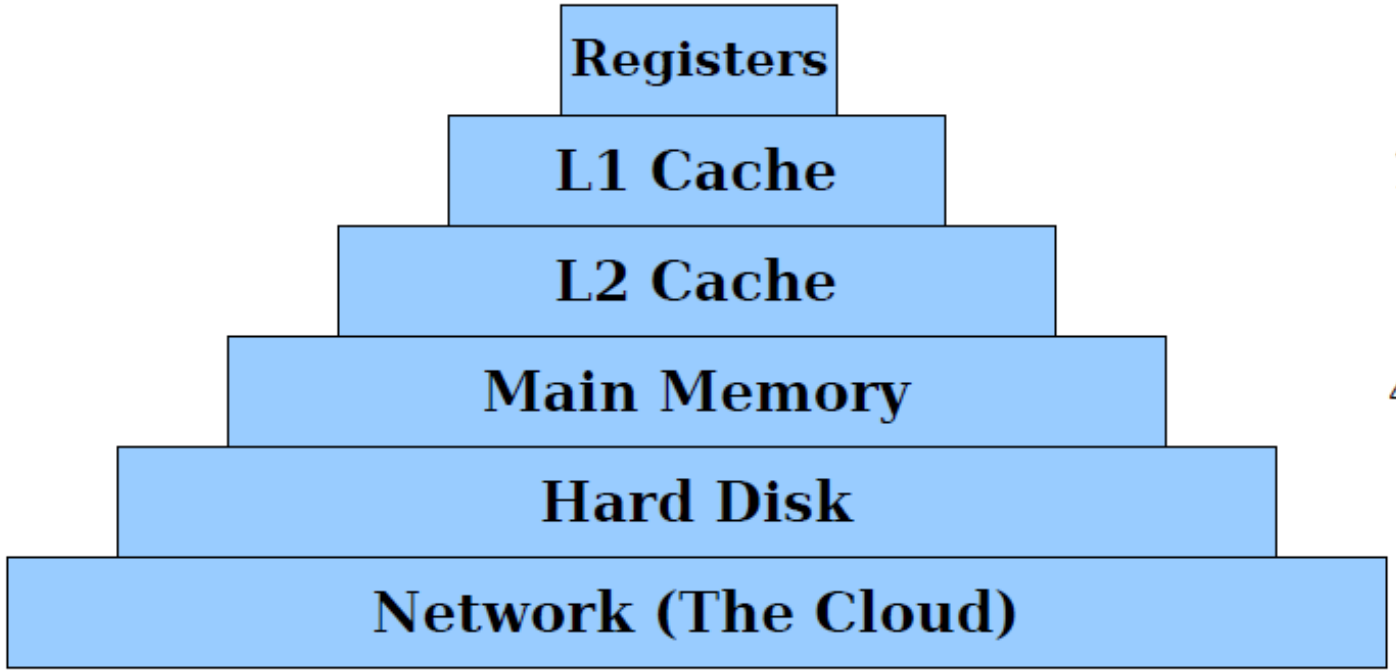
数据之法
结构之美
算法之道

zhuyungang@jlu.edu.cn

Last updated on 2023.12

计算机存储体系

C

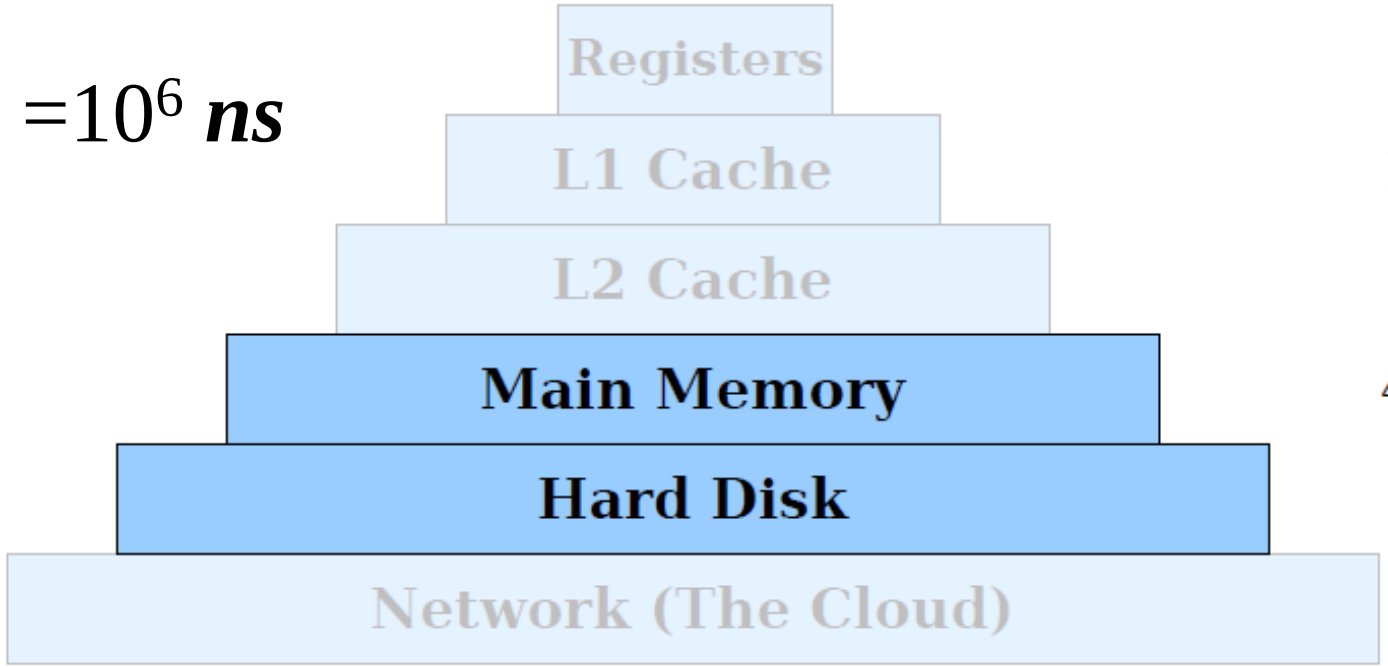


	存储容量	访问速度
Registers	256B - 8KB	0.25 - 1ns
L1 Cache	16KB - 64KB	1ns - 5ns
L2 Cache	1MB - 4MB	5ns - 25ns
Main Memory	4GB - 256GB	25ns - 100ns
Hard Disk	1TB+	3 - 10ms
Network (The Cloud)	*Lots*	10 - 2000ms

计算机存储体系

C

$1ms = 10^6\ ns$



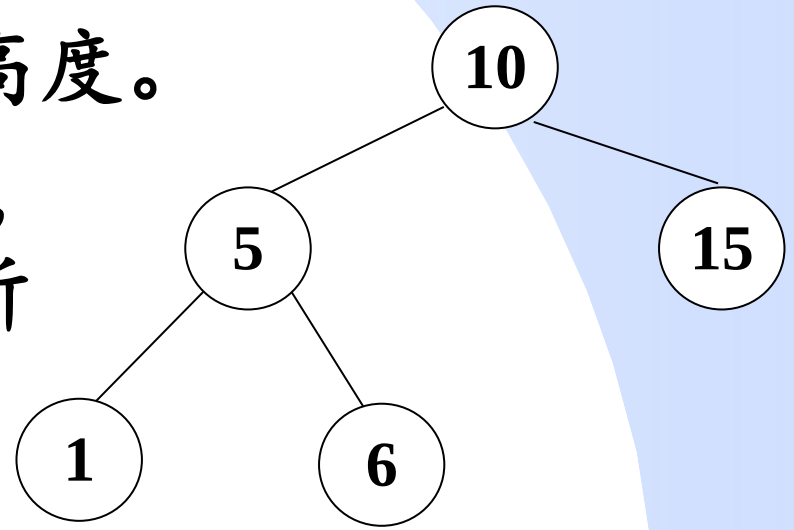
	存储容量	访问延迟
Registers	256B - 8KB	0.25 - 1ns
L1 Cache	16KB - 64KB	1ns - 5ns
L2 Cache	1MB - 4MB	5ns - 25ns
Main Memory	4GB - 256GB	25ns - 100ns
Hard Disk	1TB+	3 - 10ms
Network (The Cloud)	*Lots*	10 - 2000ms

- 外存与内存的访问速度：相差10万倍
- 若一次内存访问需要1秒，则一次外存访问就相当于1天
- 为避免1次外存访问，我们宁愿做10次、100次、甚至更多次内存访问
- 当从一个存放在外存上的文件中查找信息时，希望尽可能减少访问外存的次数。

动机

A

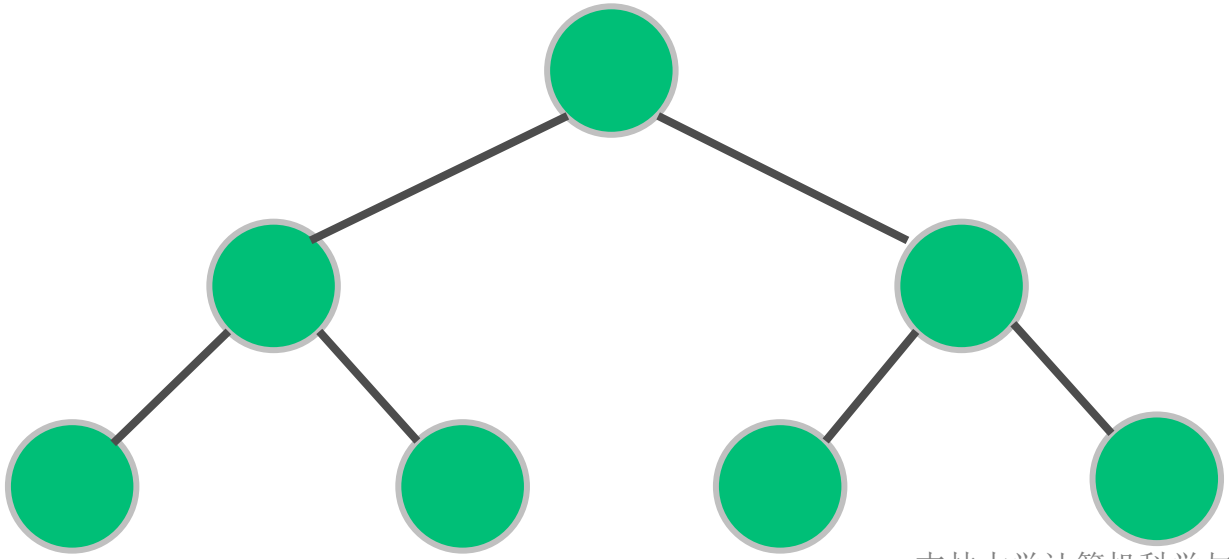
- 前面介绍的树形查找方法，主要属于内查找方法，适用于被检索文件能完整地被放入计算机内存中的情况。
- 如果文件很大，以至于计算机内存容纳不下时，则必须放在硬盘等外存储器上，即查找树的结点存储在外存。
- 采用树形表示存于外存上的大文件，此时指针域已不是内存地址，而是磁盘存储器的地址。
- 时间复杂度 ↔ 外存访问次数 ↔ 查找树高度。
- 如果一个文件包含n个记录，若$n=2\times10^{10}$，且该文件组织成一棵二叉查找树。查找所需平均外存访问次数约为$\log_2 n \approx 35$ 次。

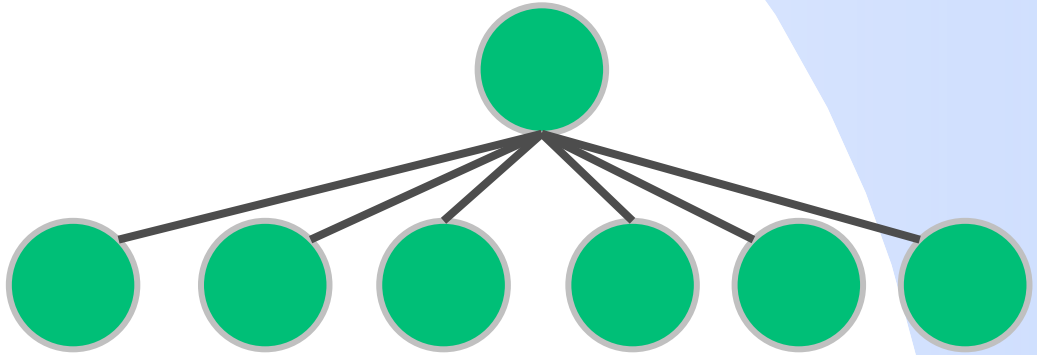


动机

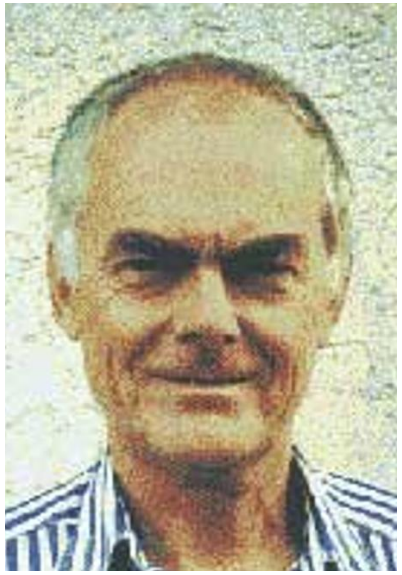
A

- 外存访问次数取决于查找树高度。
- 希望：查找树的高度尽可能矮，且一次访问外存时多读一些数据。
- 解决方案：一个结点存多个关键词，形成多叉查找树来代替二叉查找树，降低查找树高度。
- B树：*m*叉平衡查找树。





B树（B-Tree）



Rudolf Bayer
波音公司高级研究员
慕尼黑工业大学教授

Edward McCreight
波音公司研究员

I am occasionally asked what the **B** in B-Tree means. We wanted the name to be short, quick to type, easy to remember. It honored our employer, **B**oeing Airplane Company. It suggested **B**alance. Rudolf **B**ayer was the senior researcher of the two of us.

We had been admiring the elegant natural balance of AVL Trees, but for reasons clear to American English speakers, the name BM Tree was a non-starter.

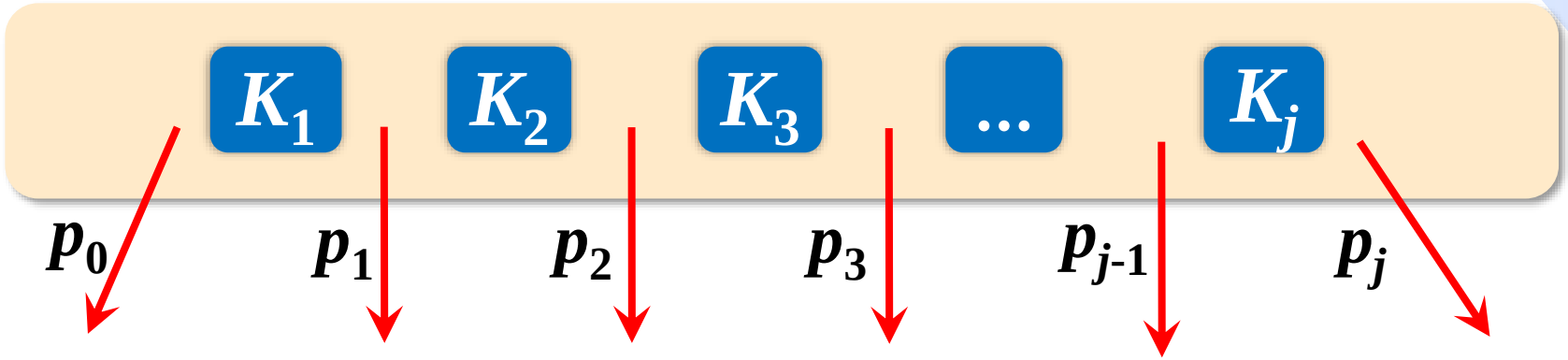
——Edward McCreight

The more you think about what the **B** could mean, the more you learn about B-Trees, and that is good.

——Rudolf Bayer

B树结点结构

- 一个包含j个关键词和$j+1$个指针的B树结点如图所示。结点之关键词满足 $K_1 < K_2 < \ldots < K_j$，第i个指针p_i（$0 \le i \le j$）指向的子树所包含关键词都在K_i和K_{i+1}之间。
- 希望一个结点里存的记录不能太多，也不能太少。



B树的结点

m阶B树的定义

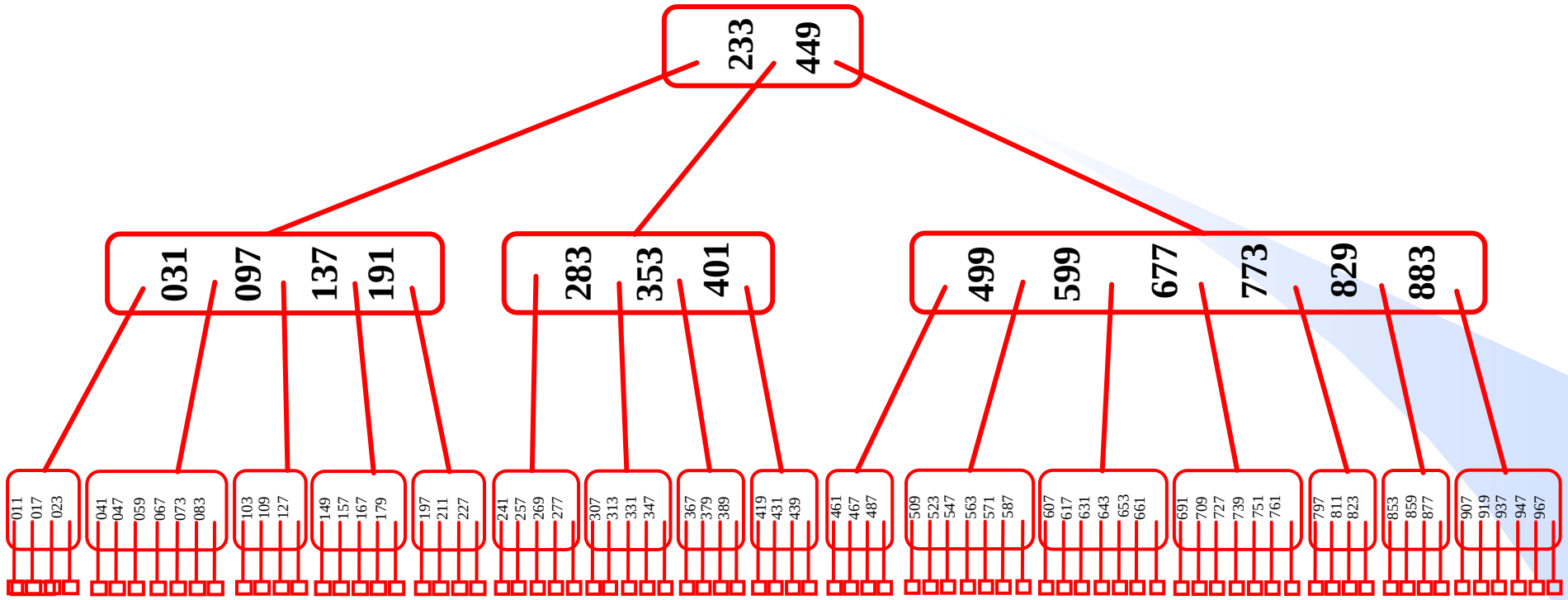
A

① 每个结点至多有m个子结点；
② 除根结点外，每个非叶结点至少有$\lceil m/2 \rceil$个子结点；
③ 若根结点不是叶结点，则至少有2个子结点
④ 有k个子结点的结点恰好包含$k-1$个递增有序的关键词；
⑤ 所有的叶结点在同一层，且不带有信息。

根以外的非叶结点有$\lceil m/2 \rceil$～m个孩子

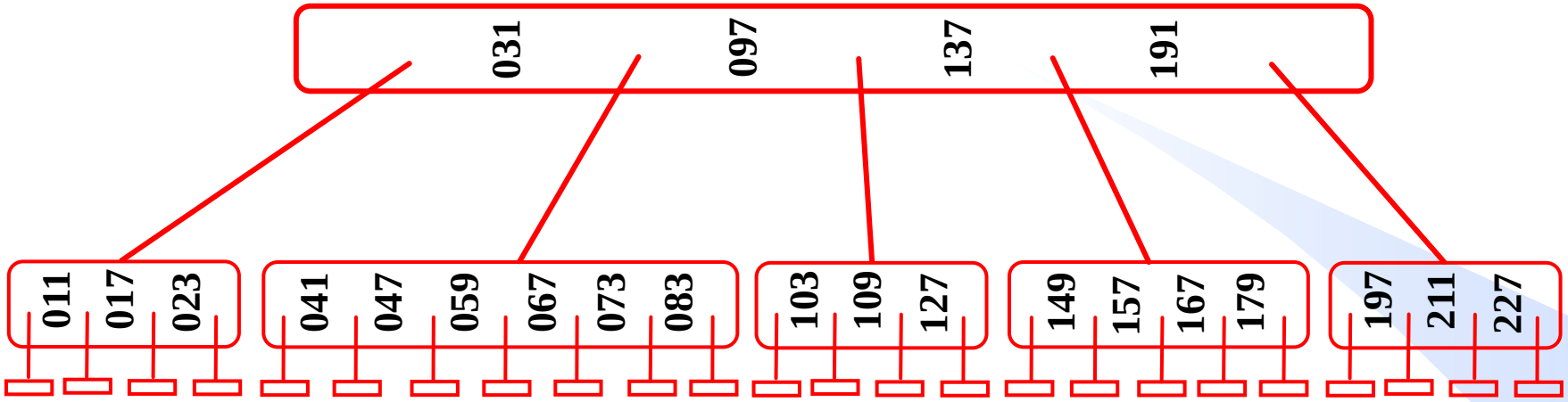
根结点有2～m个孩子

一个 7 阶B树示例：m=7



- 除根外的非叶结点有⌈m/2⌉～ m个孩子，即4 ～ 7个孩子，即包含3 ～ 6个关键词；
- 根结点有2 ～ 7个孩子，即包含1 ～ 6 个关键词；
- 每个结点中的关键词从左到右递增排列。

关于叶结点



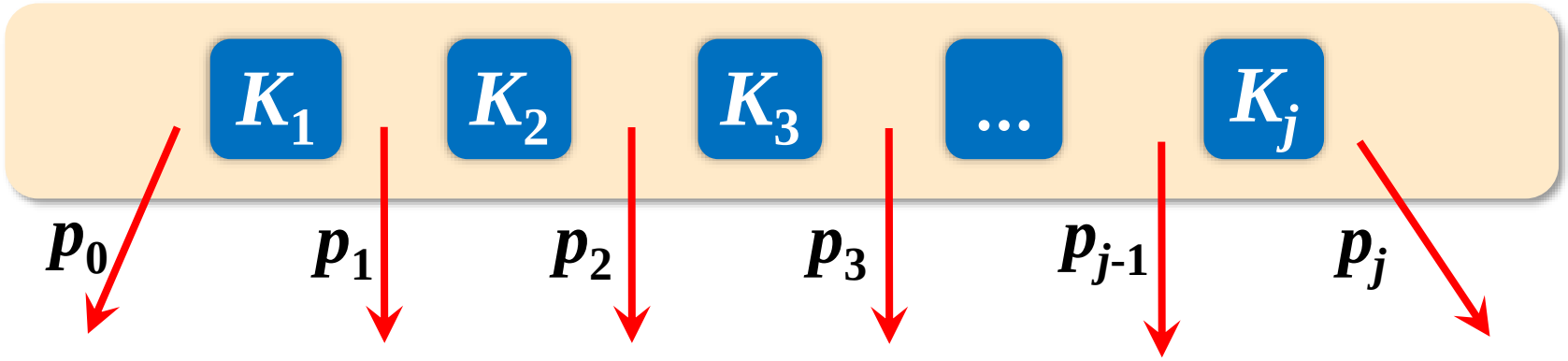
- 叶结点是虚拟的结点，不包含数据。
- 指向叶结点的指针是空指针。
- 叶结点有时可以不画。

B树的查找

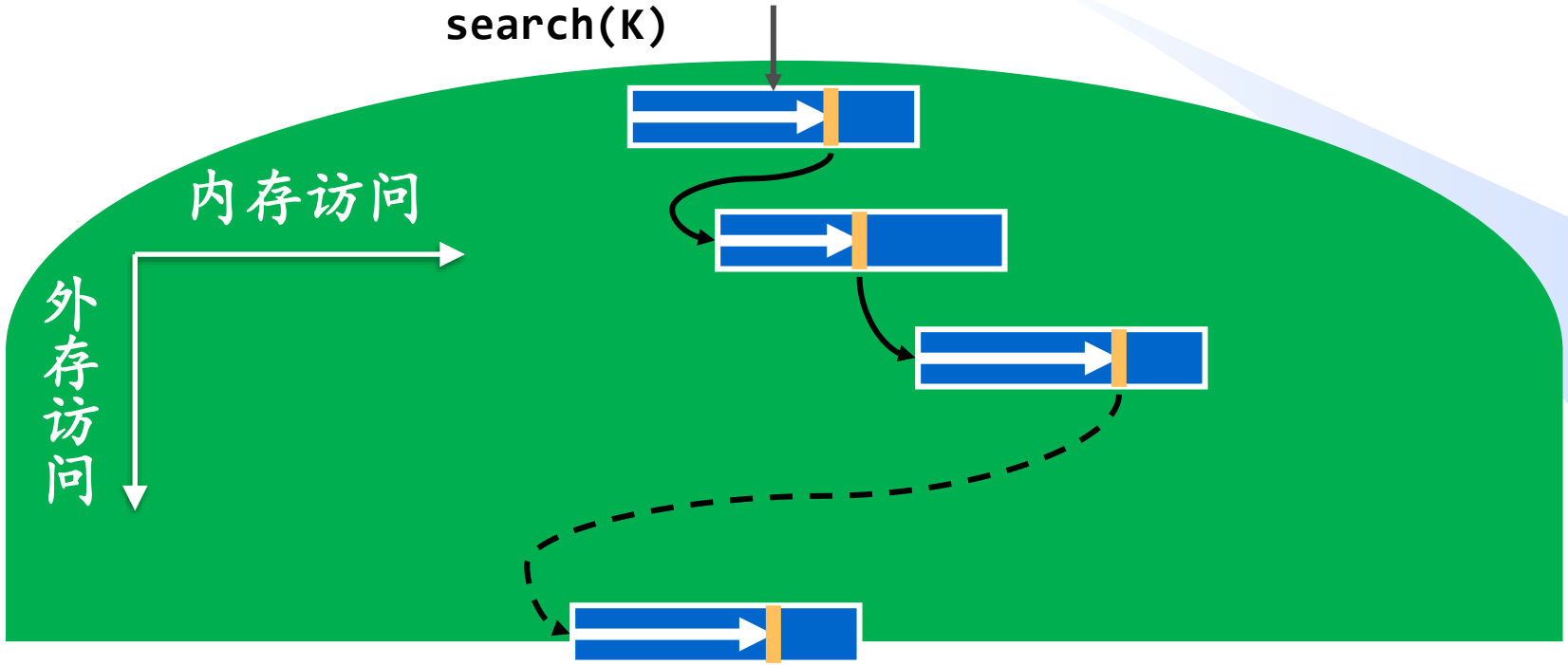
查找给定关键词K

① 在根结点内查找K，即在其所包含的关键词$K_1,\ldots,K_j$中查找K（可采用顺序查找或对半查找），找到则查找成功；

② 否则，确定要查的关键词是在某个 K_i 和 K_{i+1} 之间，于是在指针p_i所指向的子树里继续查找K，即将 p_i 所指向的结点读入内存，继续查找；如果 p_i 为空，则查找失败。



B树的查找

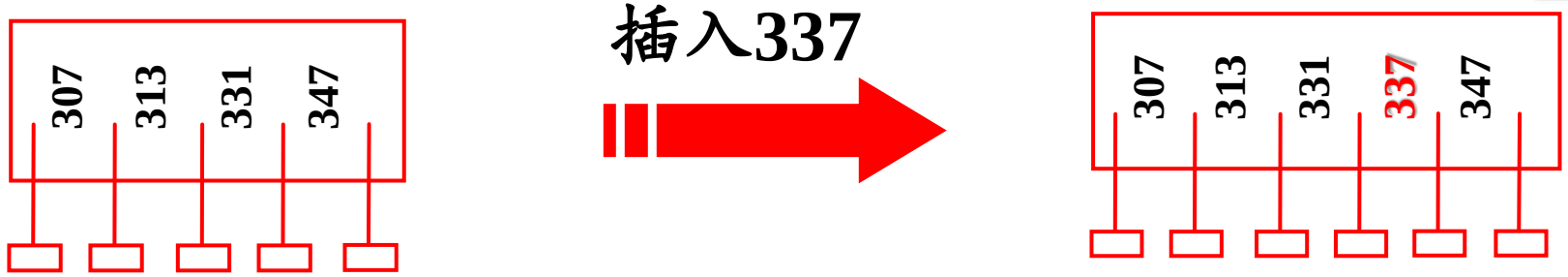


B树的插入

A

若在一个包含 $j < m-1$ 个关键词（子结点数$< m$）的结点中插入一个新关键词，则插入过程将局限于该结点自身（把新关键词直接插入该结点中即可）。如在下图所示的7阶B树中插入关键词337，只须改变一个结点即可。

7阶B树



每个结点包含最多m-1个关键词

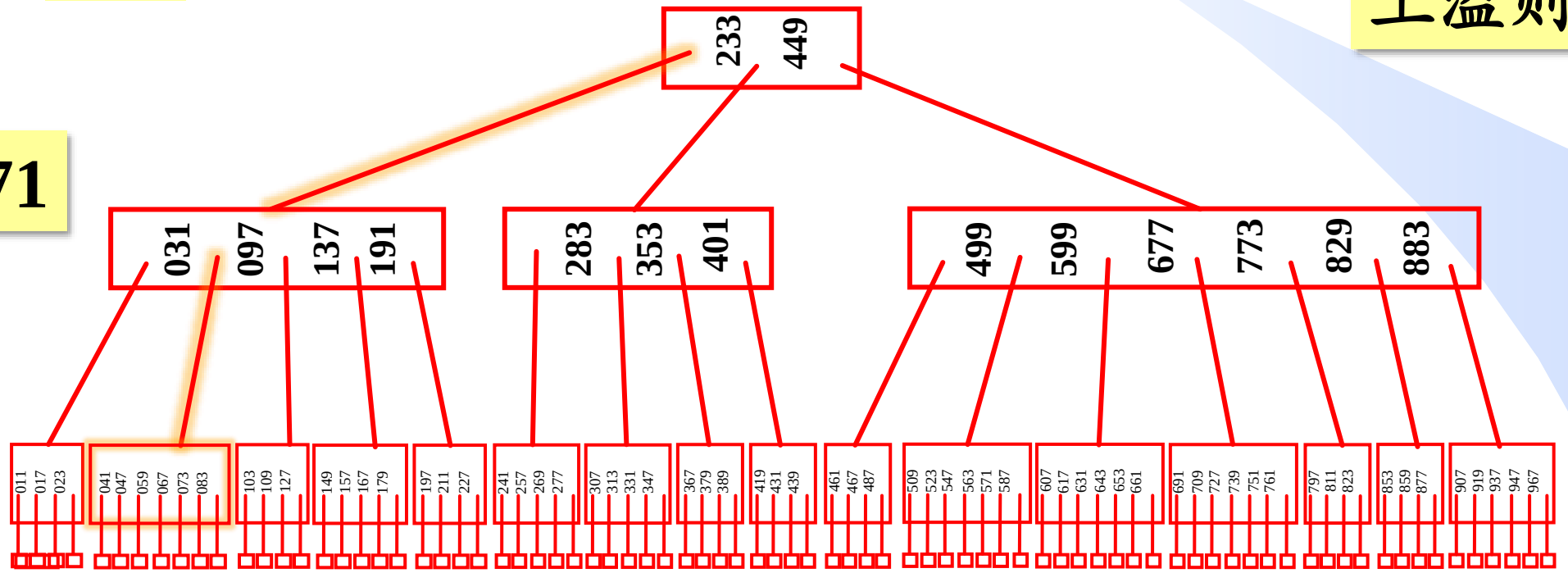
B树的插入

A

若把一个新关键词插入一个已包含$m-1$个（m为B树的阶）关键词的结点（结点已满），则插入将造成此结点关键词上溢，需进行分裂操作。

上溢则分裂

插入71

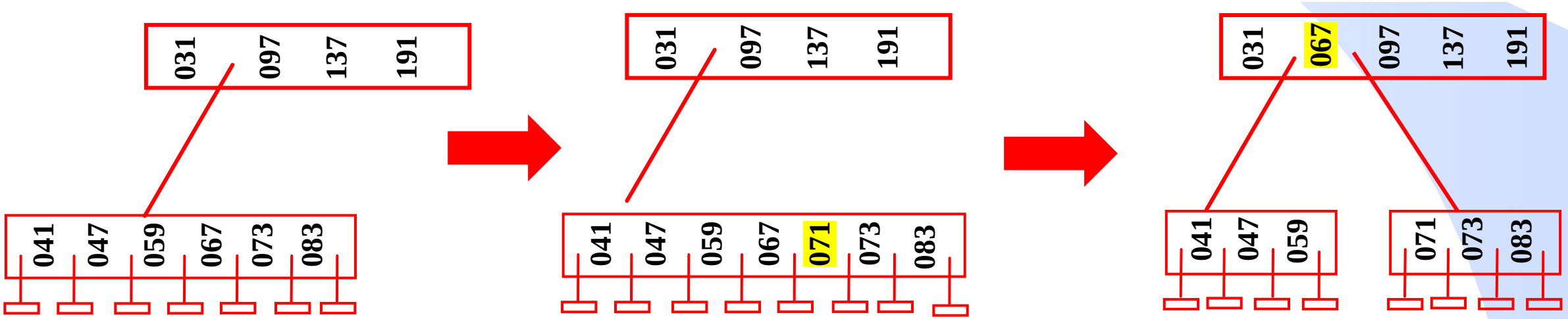


7阶B树

上溢的处理——分裂操作

A

- 结点内关键词$K_{\lceil m/2 \rceil}$提升到上一层结点中，$K_{\lceil m/2 \rceil}$左侧的关键词形成1组，分裂成一个结点。$K_{\lceil m/2 \rceil}$左侧的关键词形成另1组，分裂成另一个结点。
- $K_{\lceil m/2 \rceil}$提升到上一层结点，若导致上一层结点上溢，则须进一步分裂，即分裂操作可能会向上传播，直至根结点。



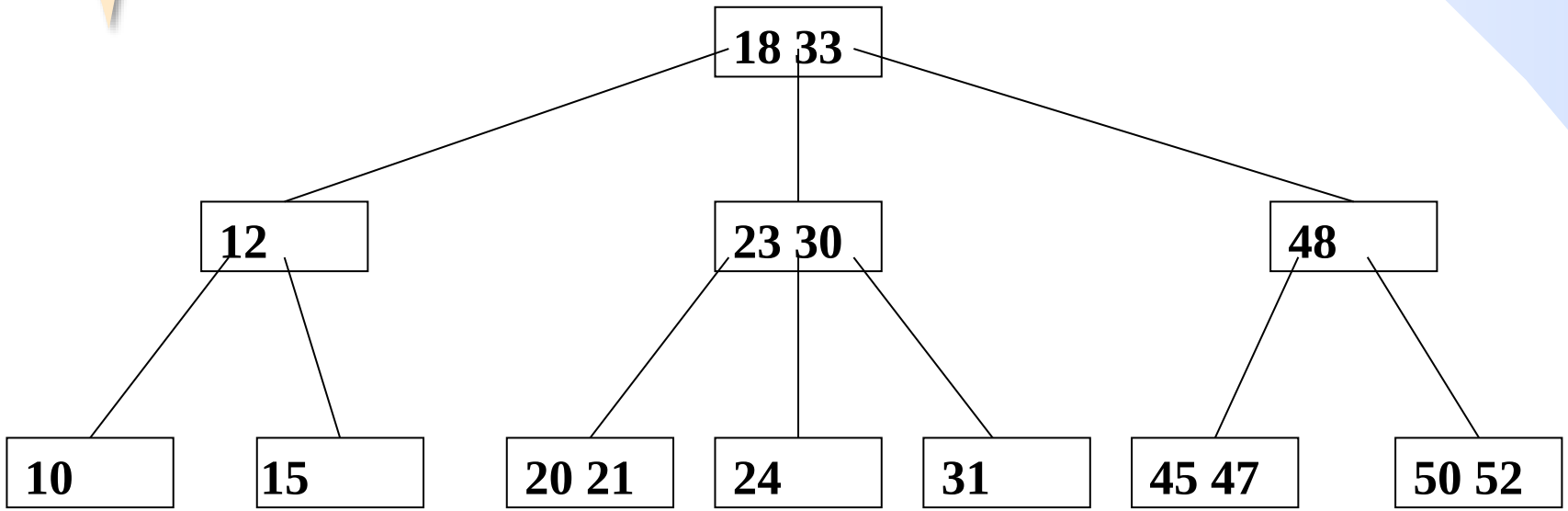
7阶B树每个结点最多包含6个关键词

例1：3阶B树插入14

A

插入14

3阶B树每个结点最多包含2个关键词



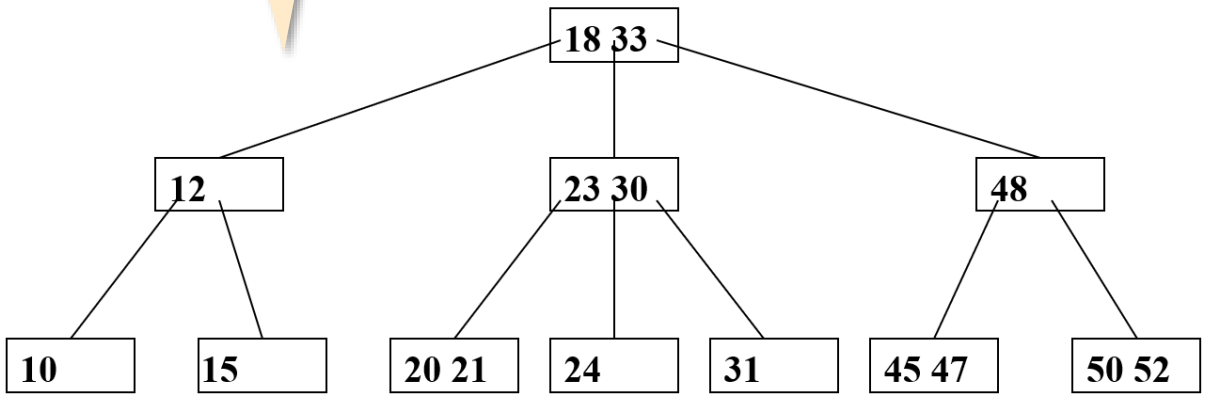
14

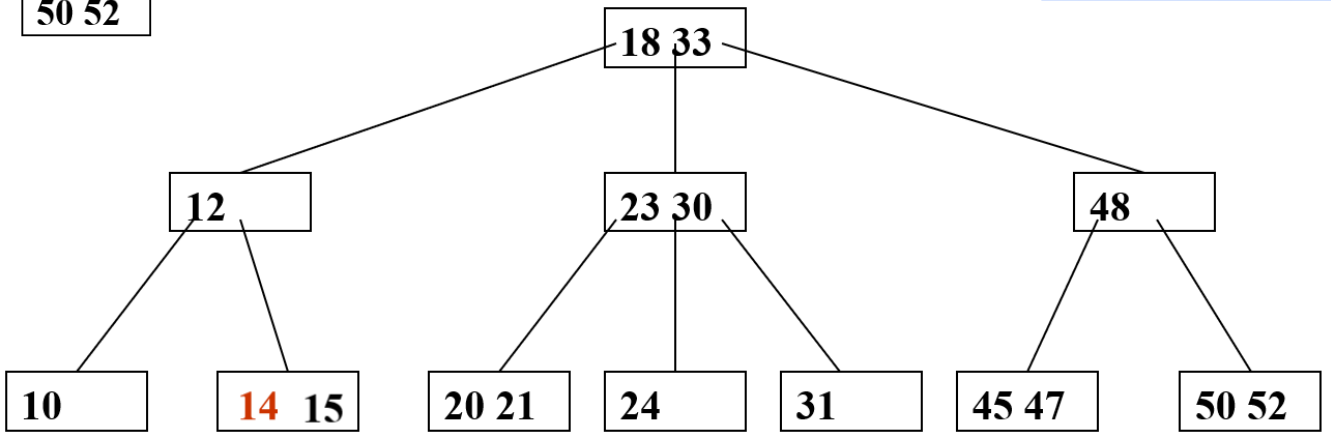
例1：3阶B树插入14（静态图）

A

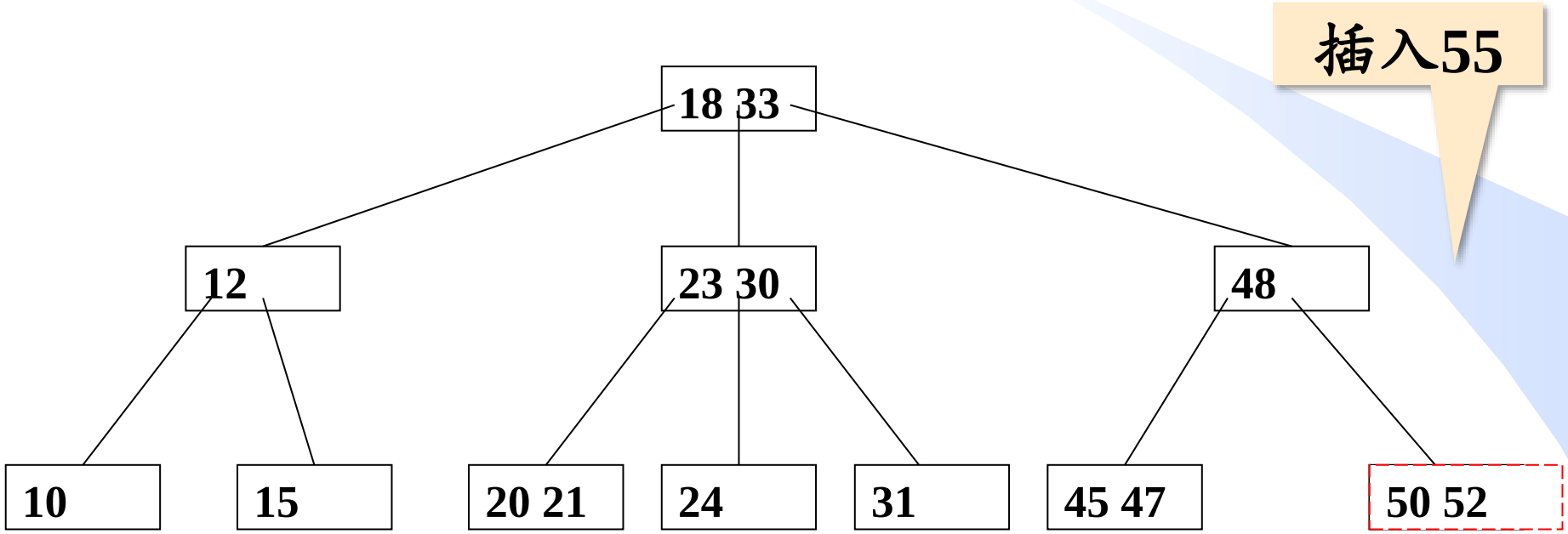
插入14

3阶B树每个结点最多包含2个关键词





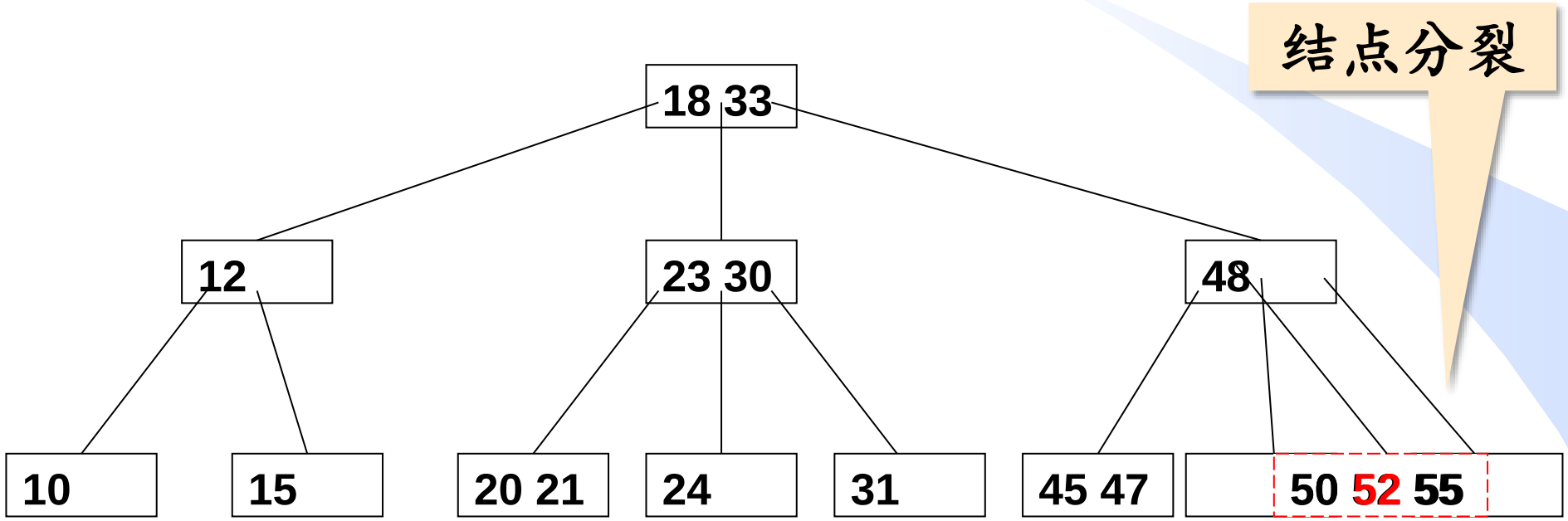
例2：3阶B树插入55



3阶B树每个结点最多包含2个关键词

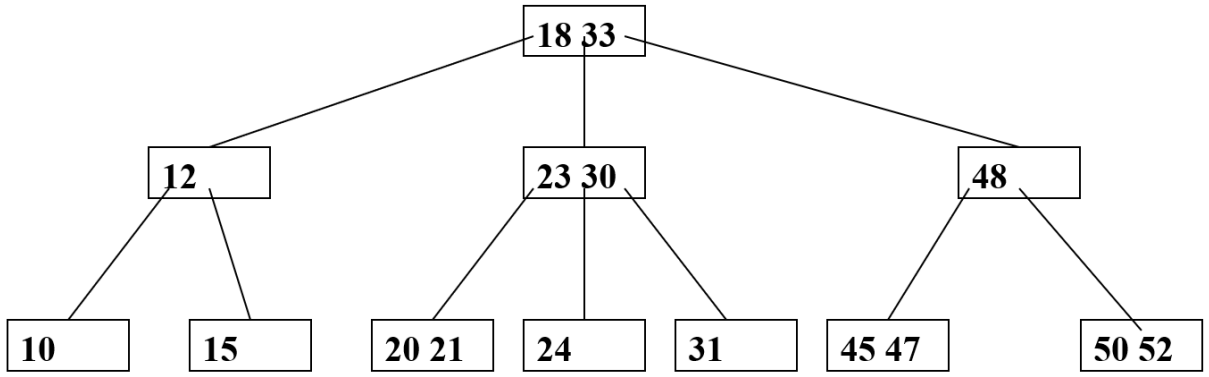
55

叶结点分裂，把52提升到父结点

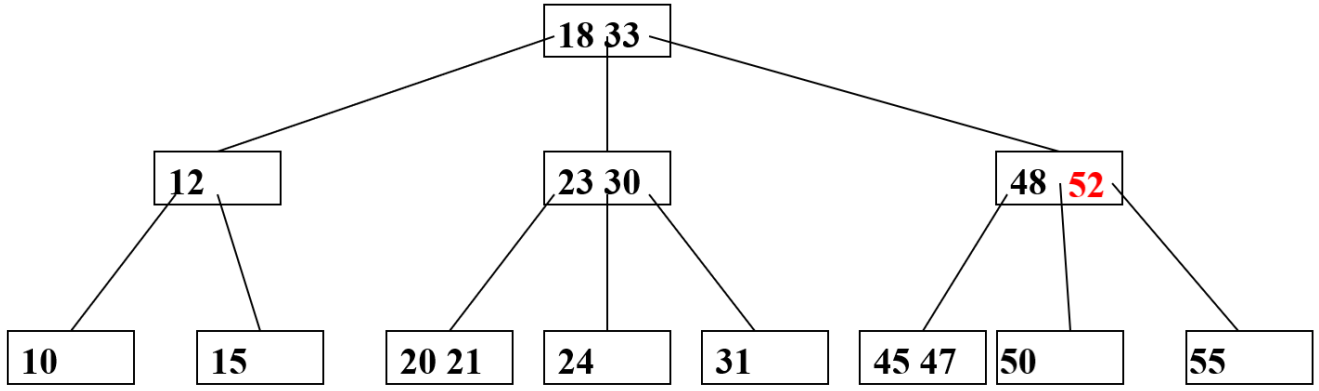


3阶B树每个结点最多包含2个关键词

例2：3阶B树插入55（静态图）

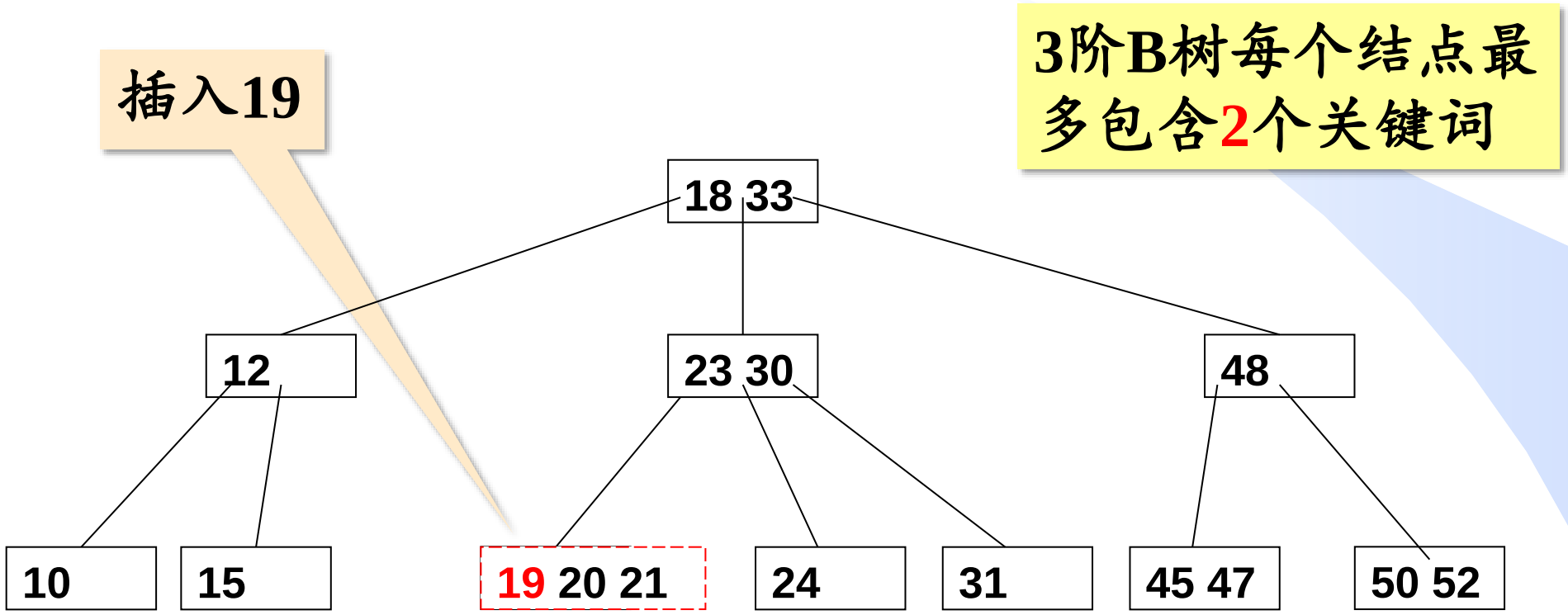


3阶B树每个结点最多包含2个关键词



例3：3阶B树插入19

A

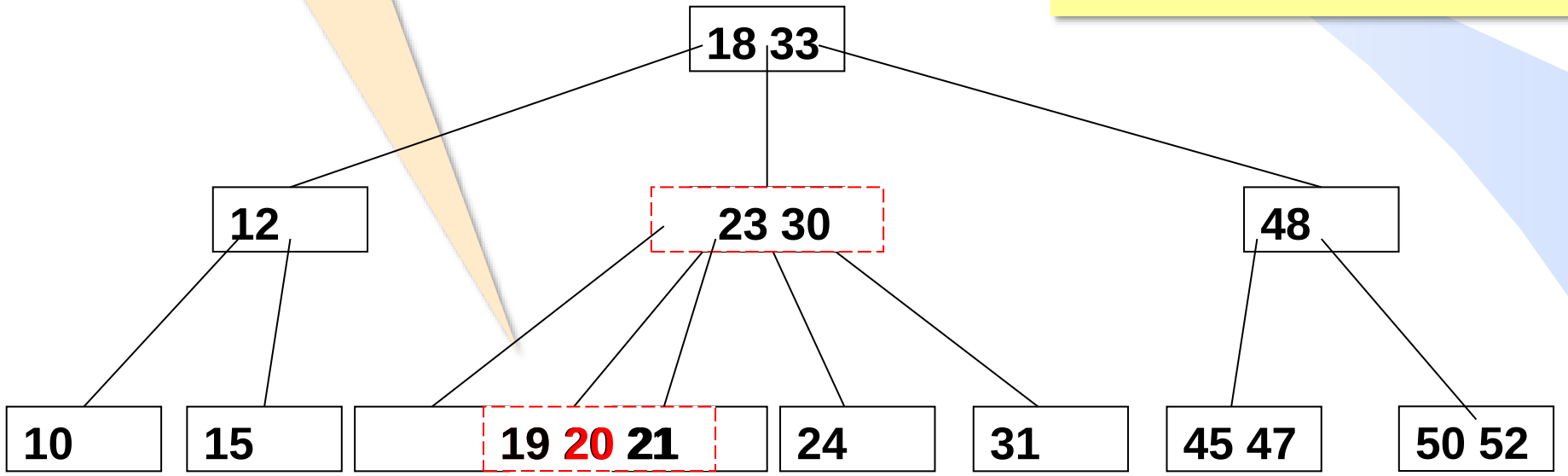


19

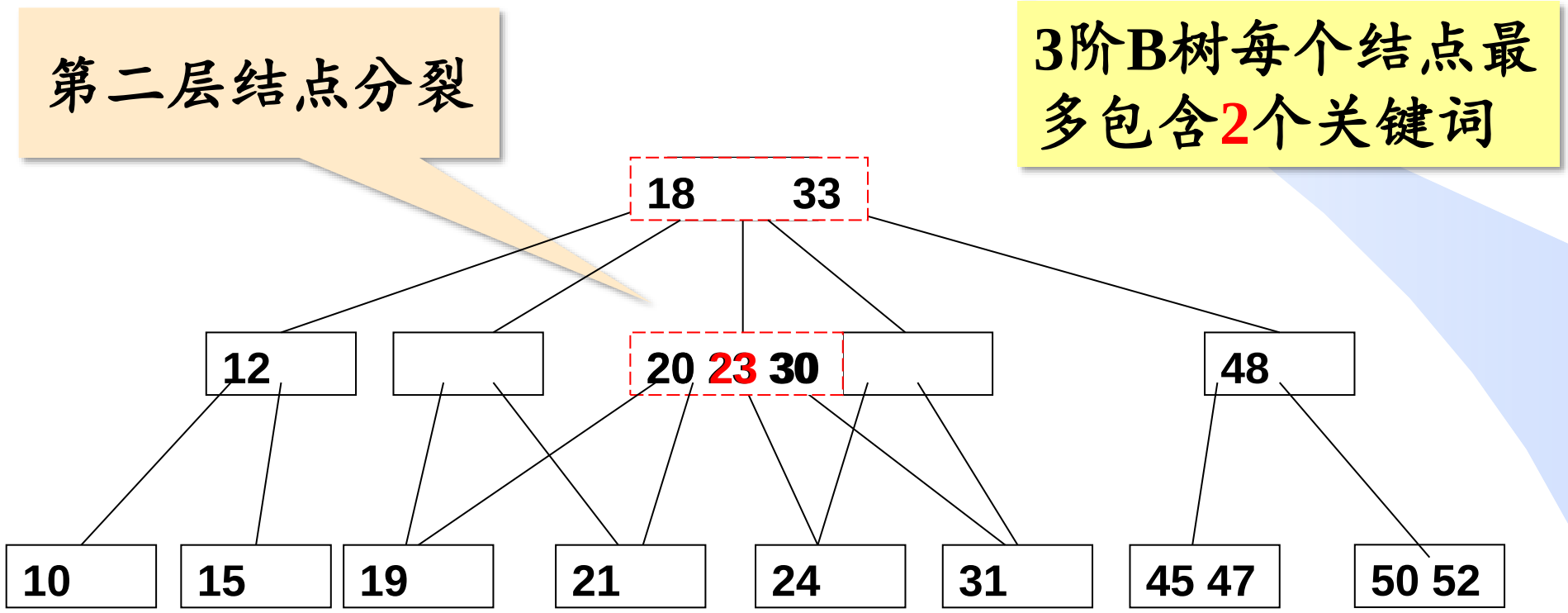
例3：3阶B树插入19

叶结点分裂

3阶B树每个结点最多包含2个关键词



例3：3阶B树插入19

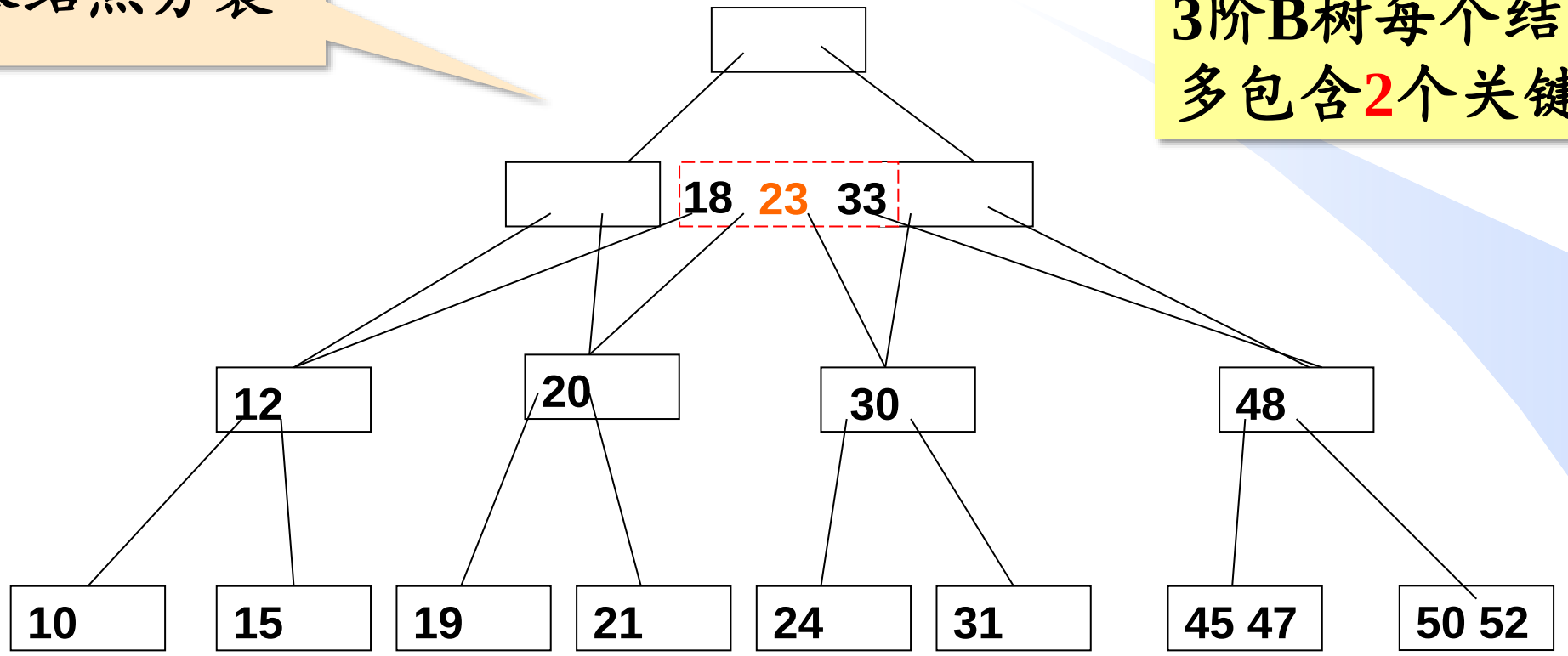

第二层结点分裂
3阶B树每个结点最多包含2个关键词
18 33
12
20 23 30
48
10
15
19
21
24
31
45 47
50 52

例3：3阶B树插入19

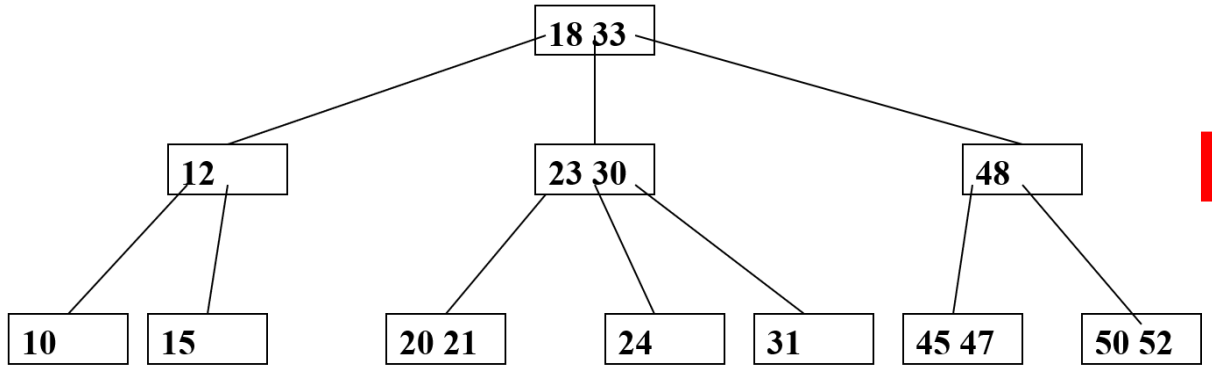
A

根结点分裂

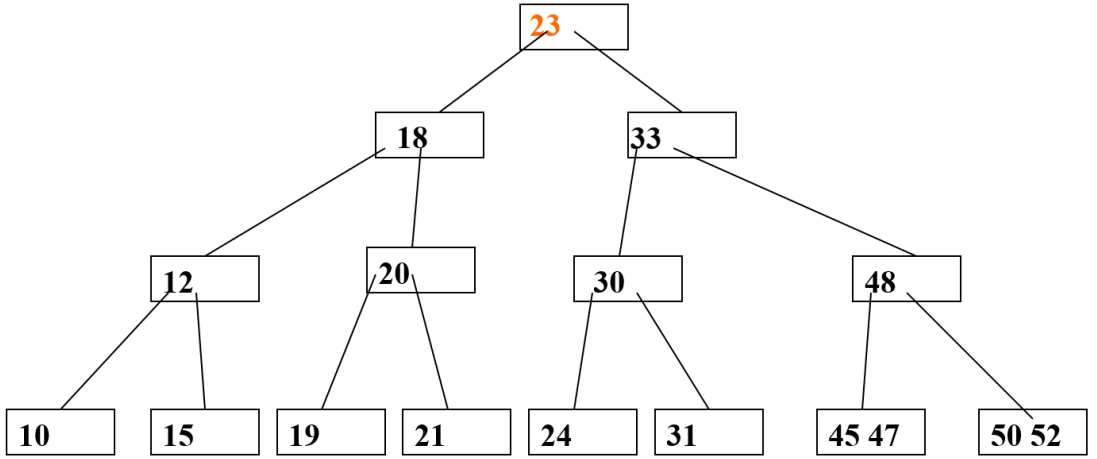
3阶B树每个结点最多包含2个关键词



例3：3阶B树插入19（静态图）



3阶B树每个结点最多包含2个关键词



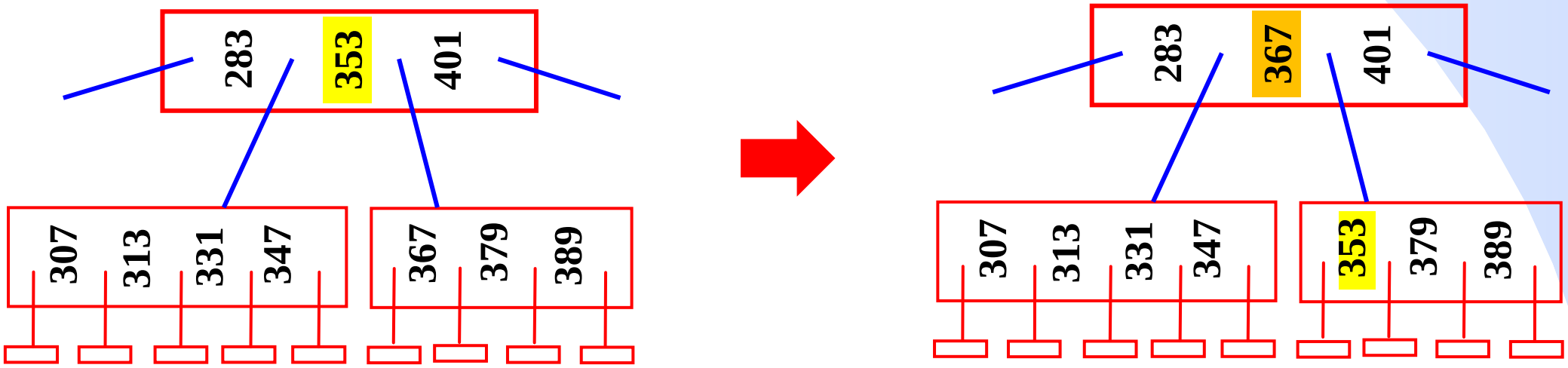
B树的删除

A

- 查找要删除的关键词K的位置。
- 换：若K不在最底层，将K与其“右子树”最小关键词交换，然后再删K，即真正的删除只发生在最底层。

如下图删除关键词 353 的情形。

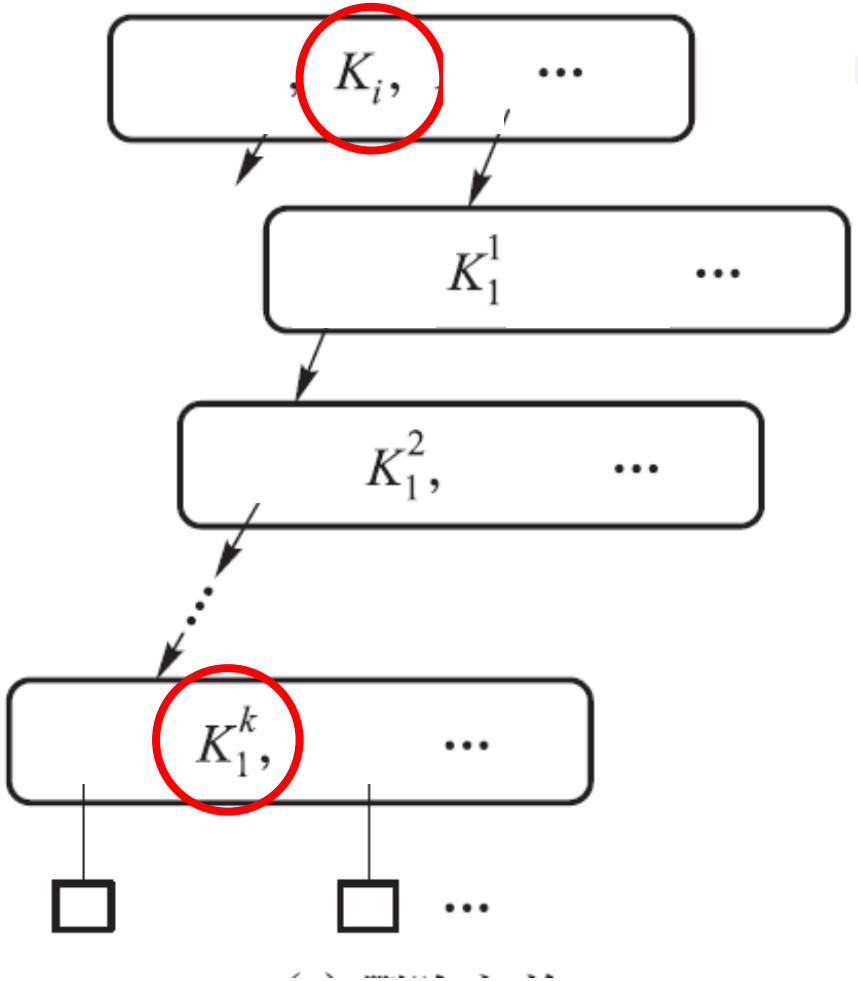
先换后删



B树的删除

A

将K与其“右子树”最底层最小关键词交换。



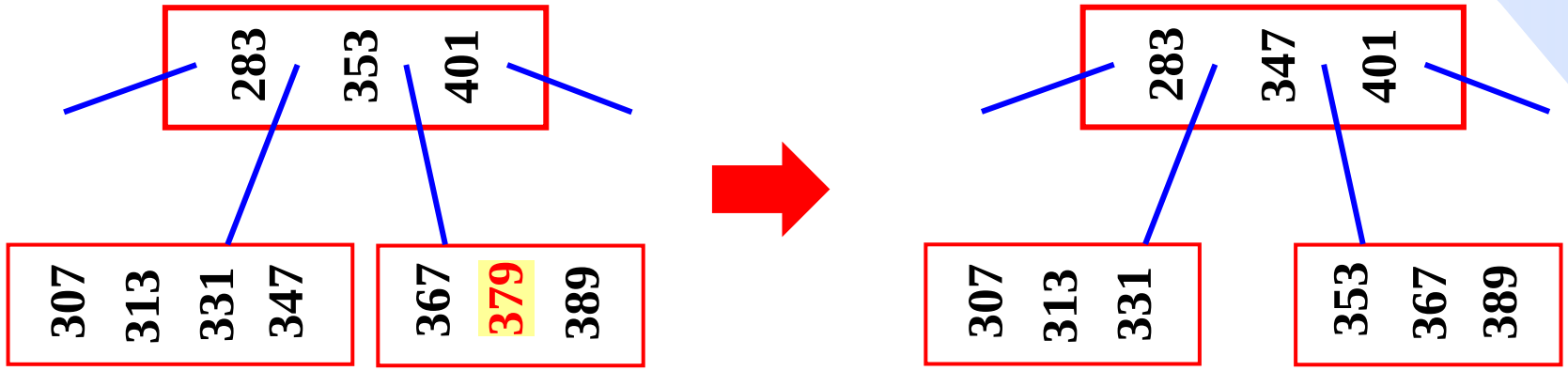
B树的删除

A

- 借：当删除关键词后，若该结点目前包含的关键词个数小于$\lceil m/2 \rceil - 1$（称为下溢）。则从左（右）兄弟结点中借最大（最小）关键词。
- 不是直接从兄弟结点借，而是通过父结点中转。
- 借的一般顺序：左顾右盼（优先从左兄弟借）。

下溢借位

如下图删除关键词 379 的情形。



7阶B树非根结点最少包含3个关键词

B树的删除

A

- 若借位不是发生在最底层：从左（右）兄弟结点中借最大（最小）关键词的同时，须连同左（右）兄弟结点最右（左）方的子树一起借。稍后通过示例说明。

B树的删除

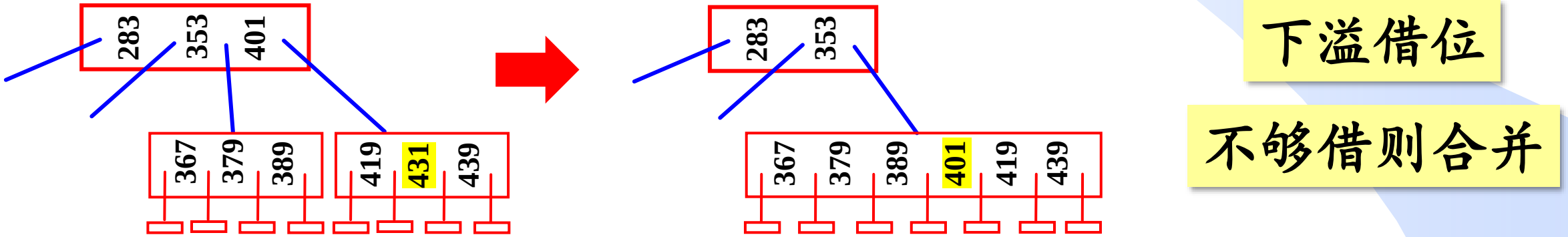
A

- 若兄弟结点不够借，即兄弟结点包含的关键词个数等于$\lceil m/2 \rceil-1$，则执行合并操作：把两个兄弟结点的关键词、父结点中指向这两个结点的指针之间的关键词按递增顺序合并到一个新结点中。
- 合并的顺序：左顾右盼（优先与左兄弟合并）。
- 例如删去关键词431。

先换后删

下溢借位

不够借则合并



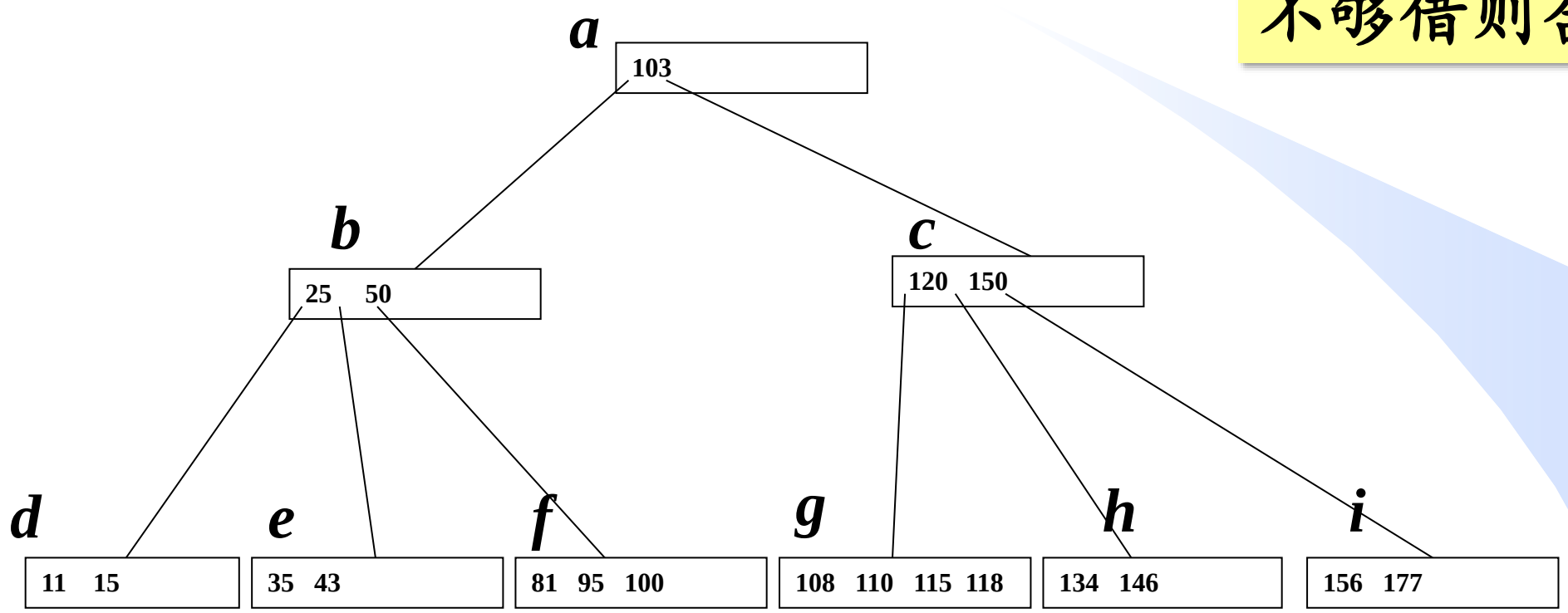
- 若“合并”操作使上一层结点（父结点）所包含的关键词个数出现不够（下溢）的情况，则对上一层结点继续“借”或“合并”。
- “合并”可能会导致上一层发生“合并”，从而可能使“合并”不断向上传播，直至根结点，进而使整个B树减少一层。

7阶B树非根结点最少包含3个关键词

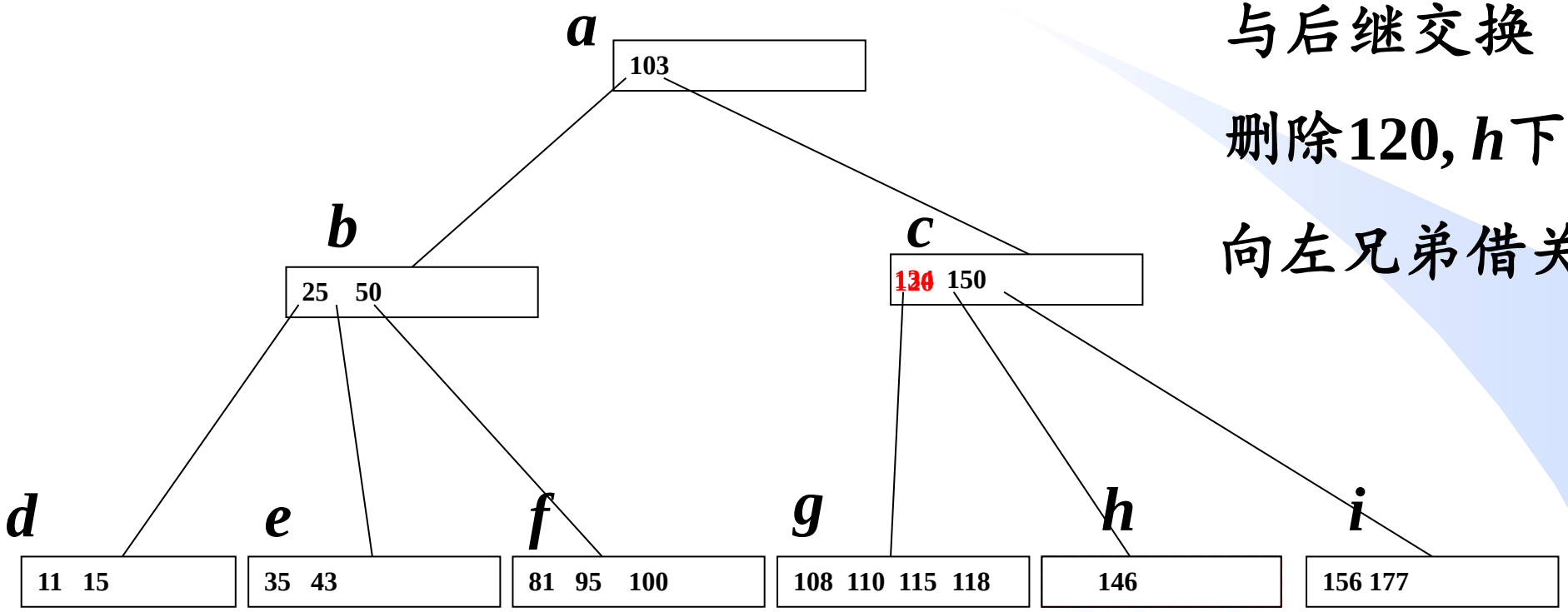
5阶B树删除120示例

A

不够借则合并



5阶B树非根结点最少包含2个关键词



与后继交换

删除120, h下溢

向左兄弟借关键词

5阶B树非根结点最少包含2个关键词

A

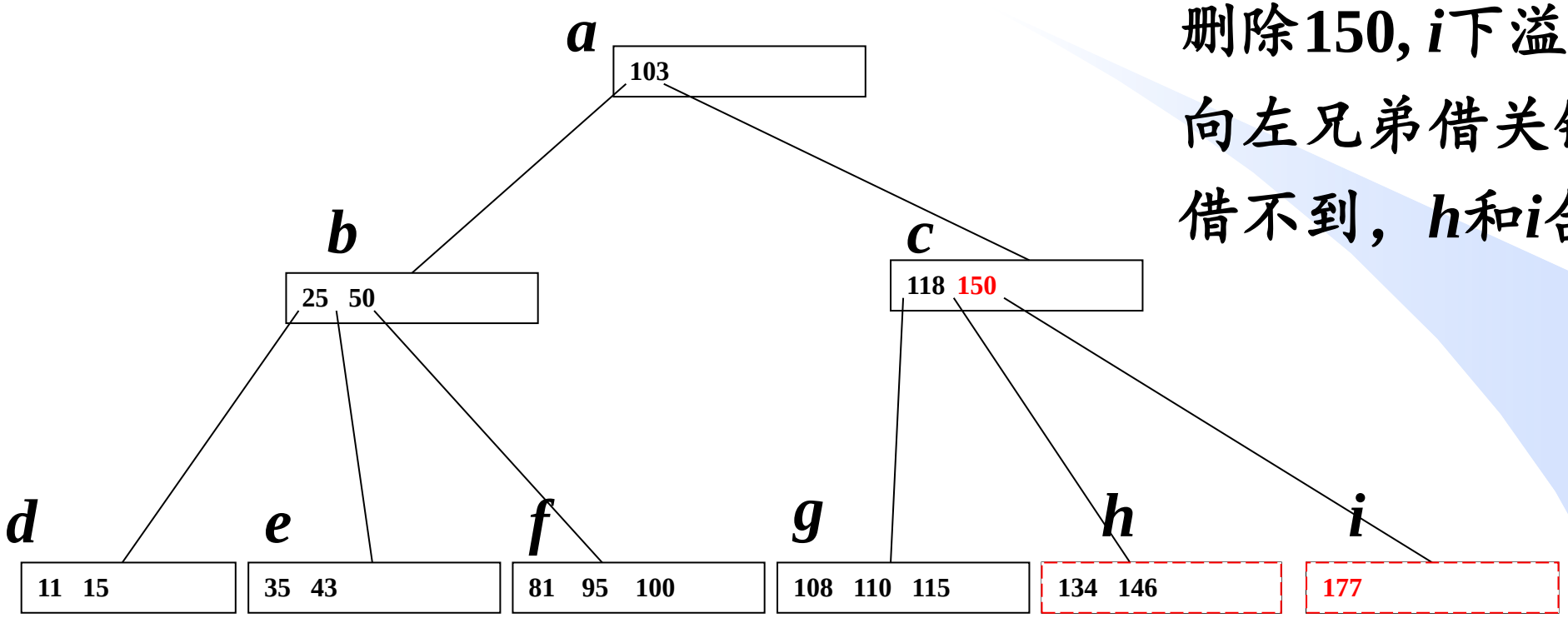
继续删除150

与后继交换

删除150, *i*下溢

向左兄弟借关键词

借不到，*h*和*i*合并



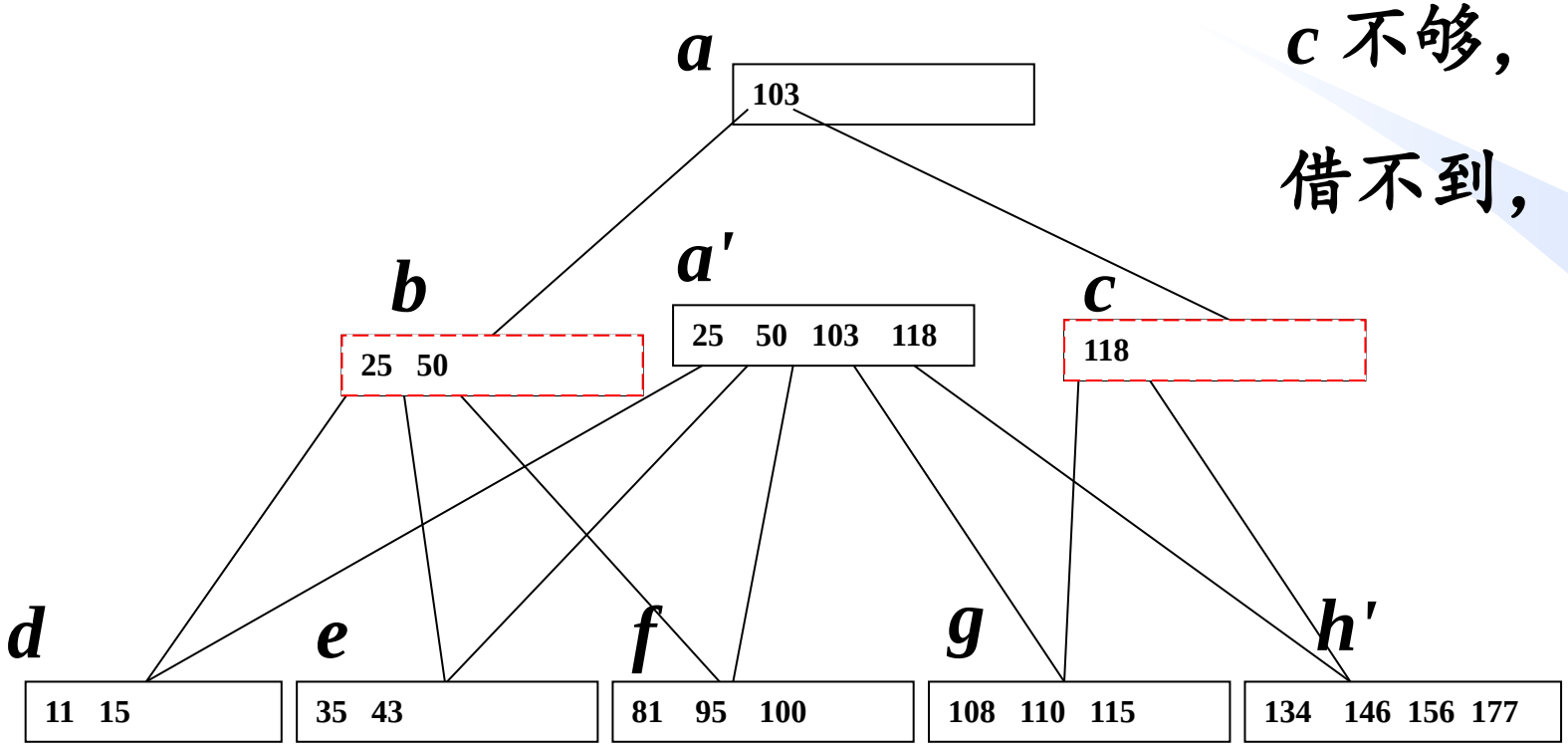
5阶B树非根结点最少包含2个关键词

A

h, i合并为h'

c 不够，向左兄弟借关键词

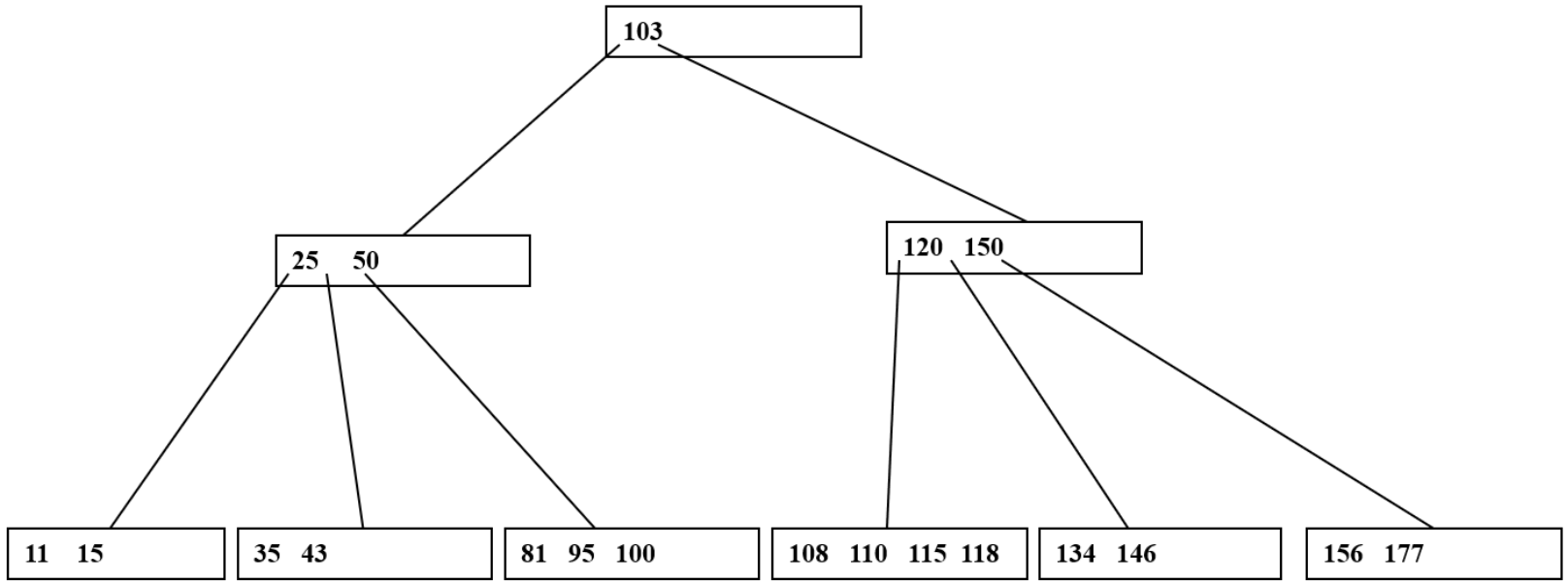
借不到，b和c合并

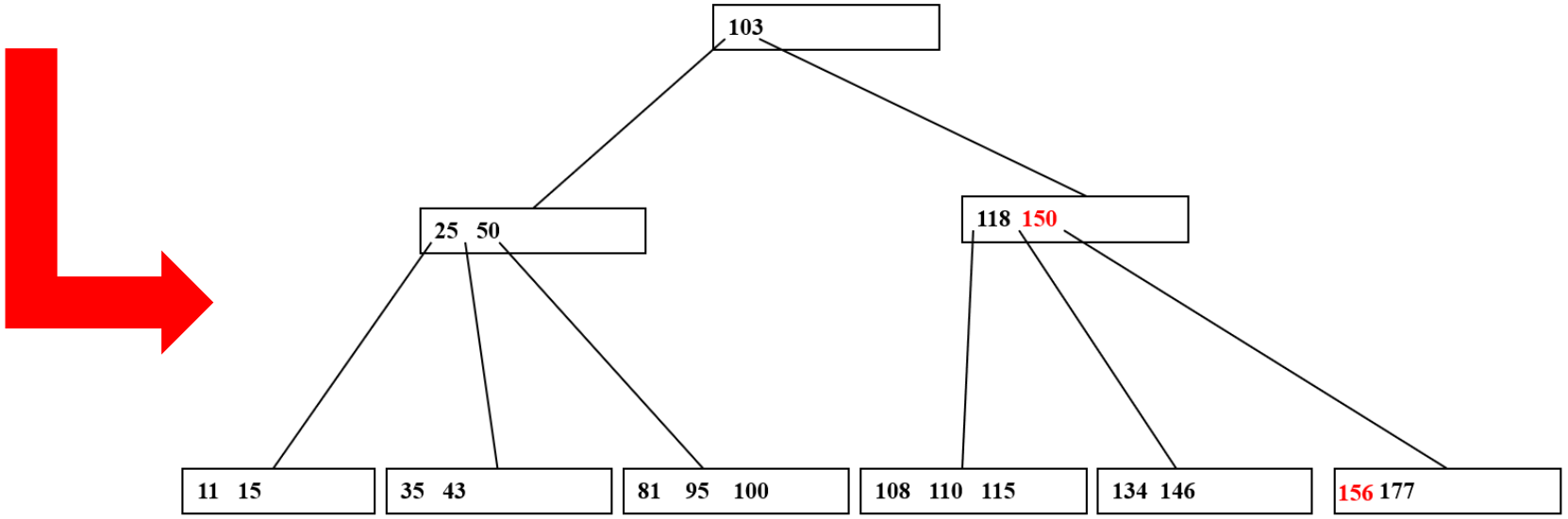


5阶B树非根结点最少包含2个关键词

5阶B树删除120示例（静态图）

A

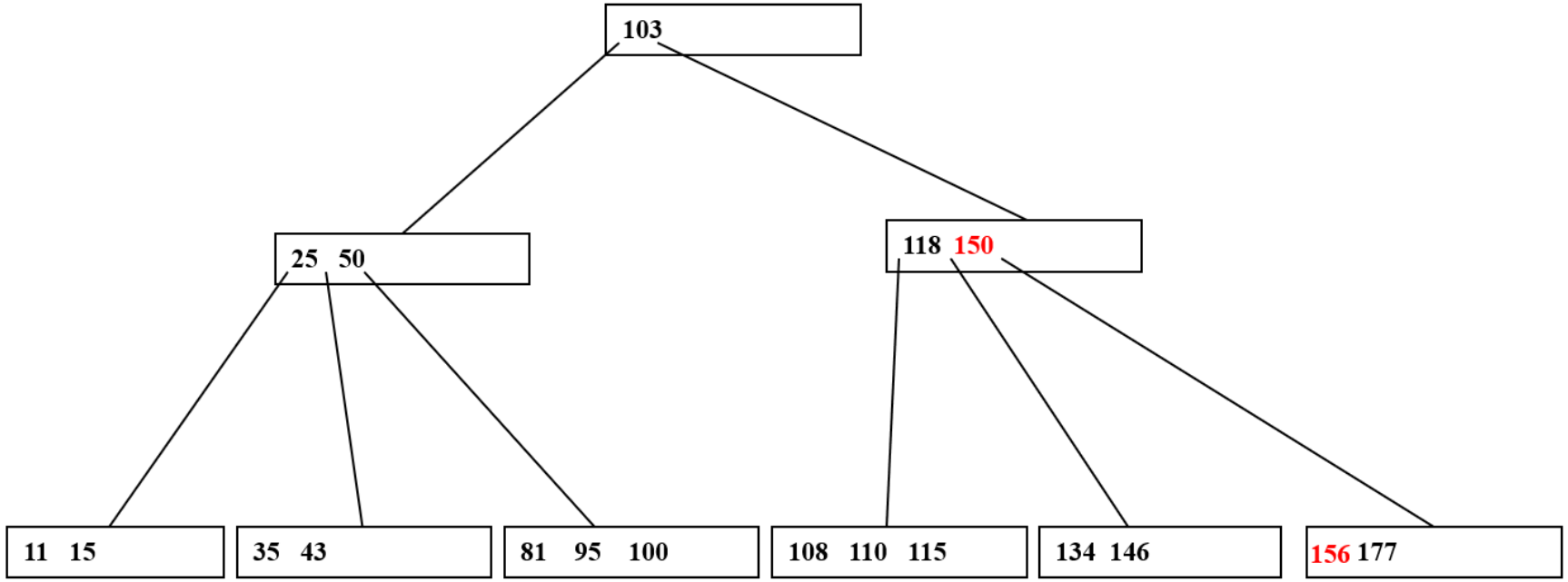




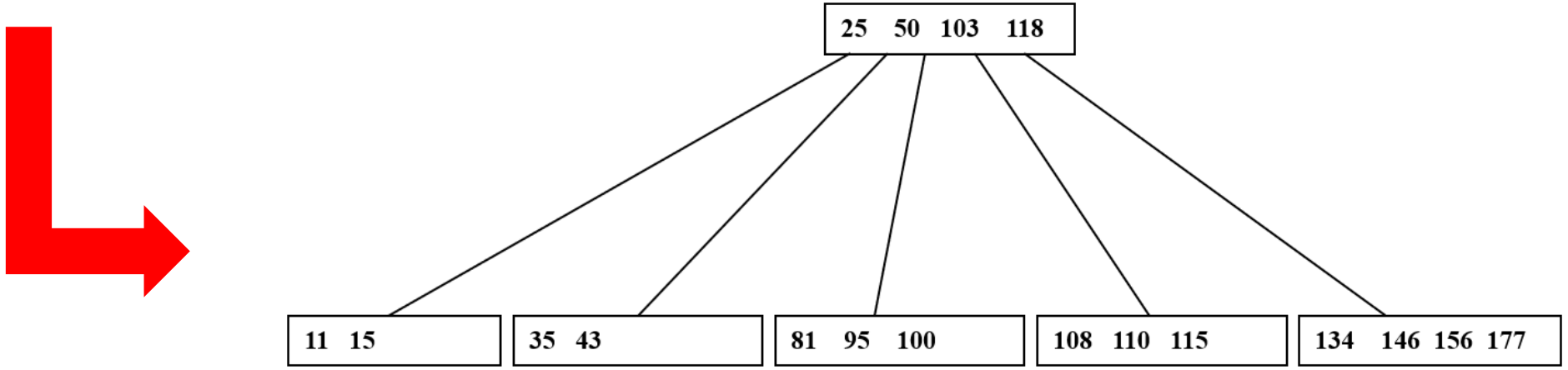
5阶B树非根结点最少包含2个关键词

5阶B树删除150示例（静态图）

A



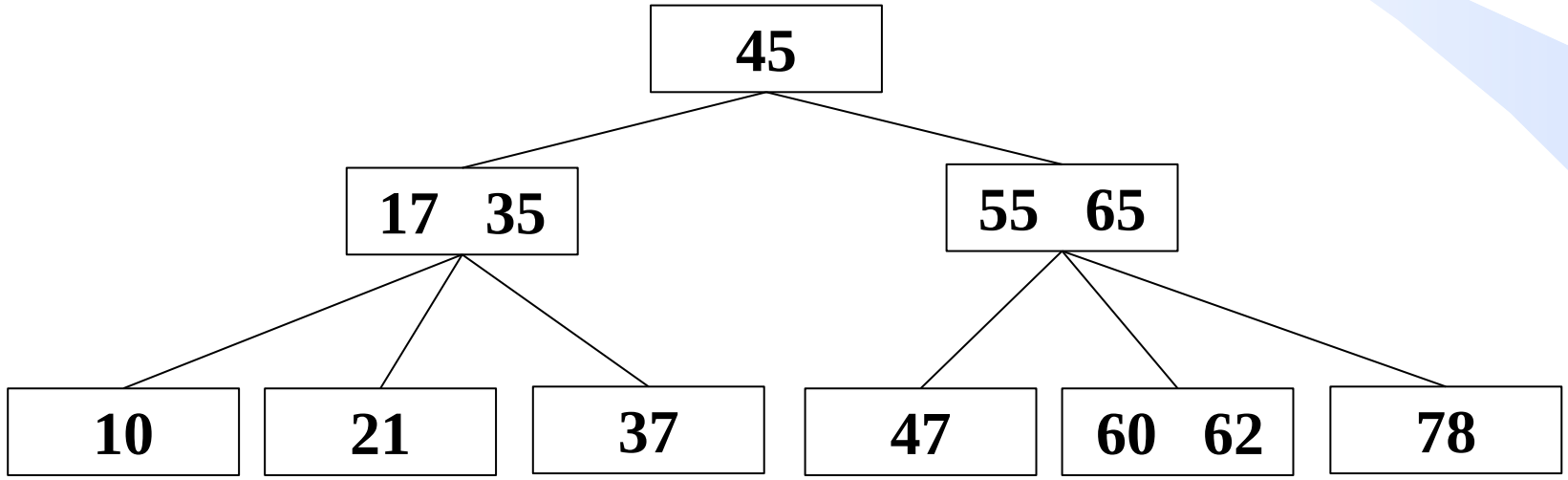
5阶B树非根结点最少包含2个关键词



练习

A

已知一棵3阶B树如下图所示，删除关键字78后得到一棵新B树，其最右叶结点中的关键字是__65__。【考研题全国卷】



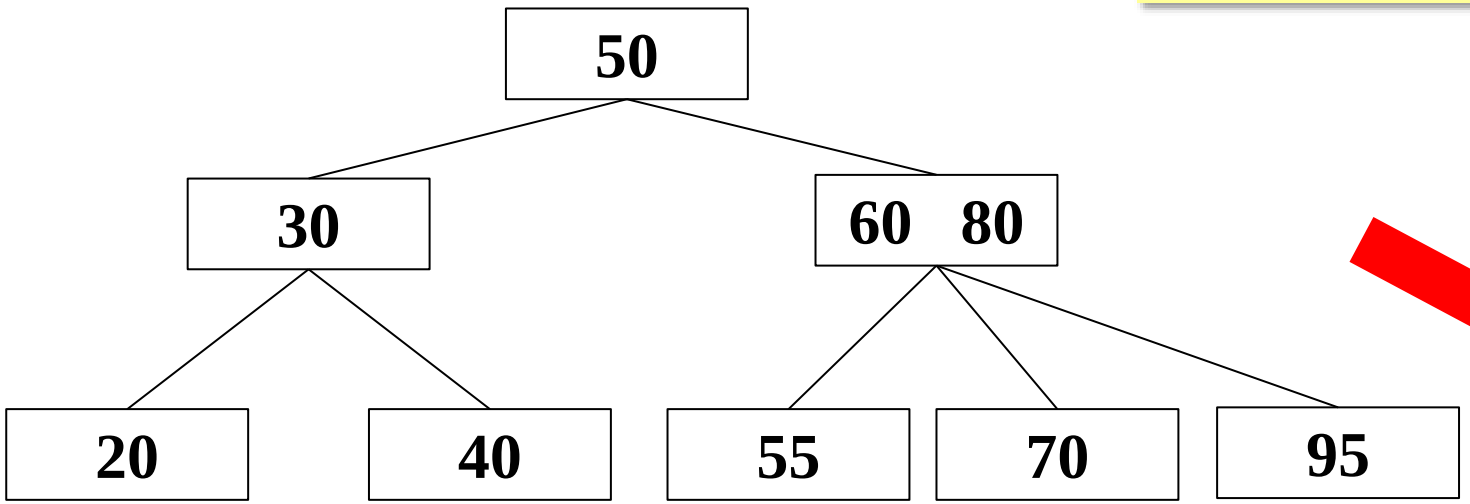
3阶B树非根结点最少包含1个关键词

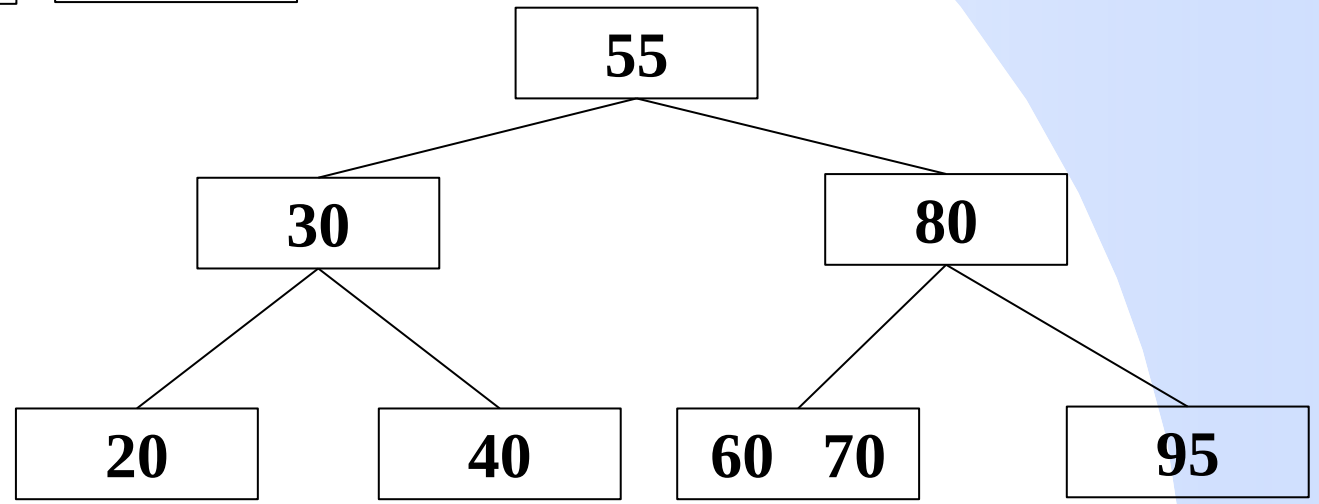
练习

A

对如下3阶B树删除关键字50

3阶B树非根结点最少包含1个关键词



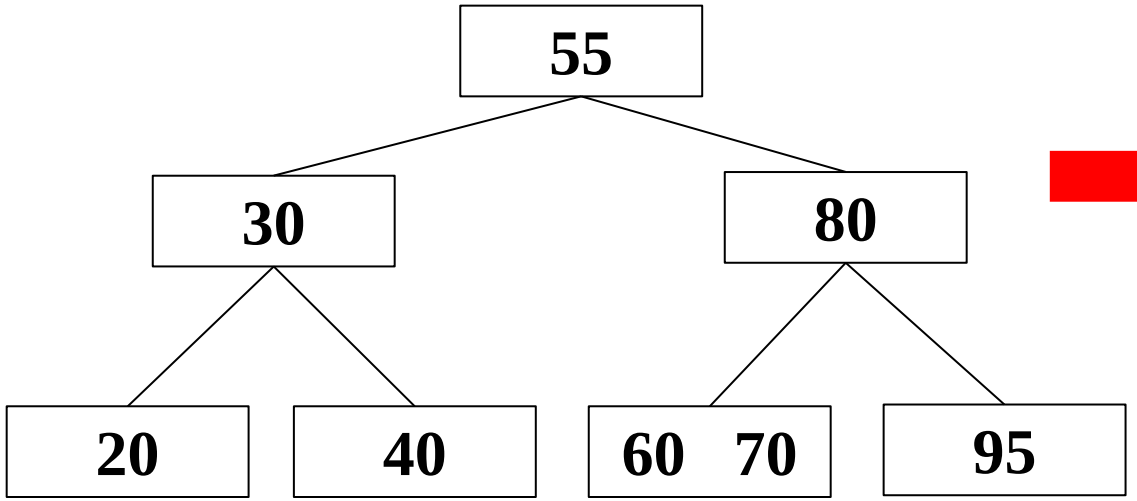


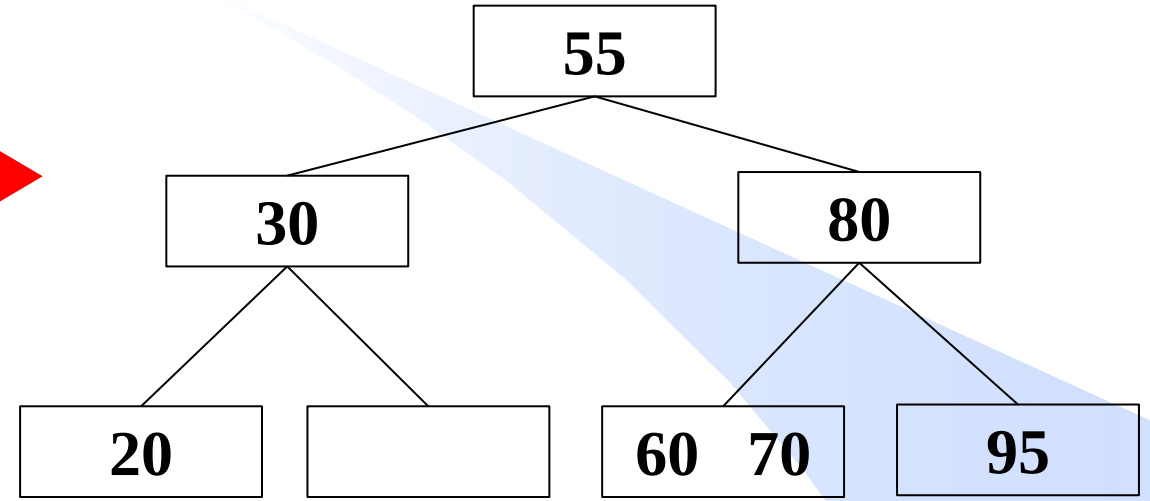
练习

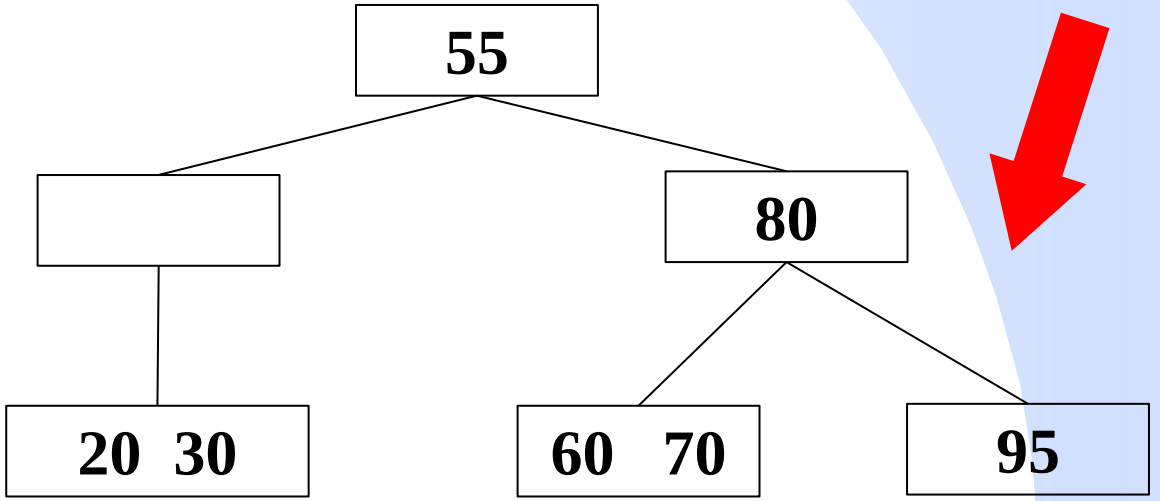
A

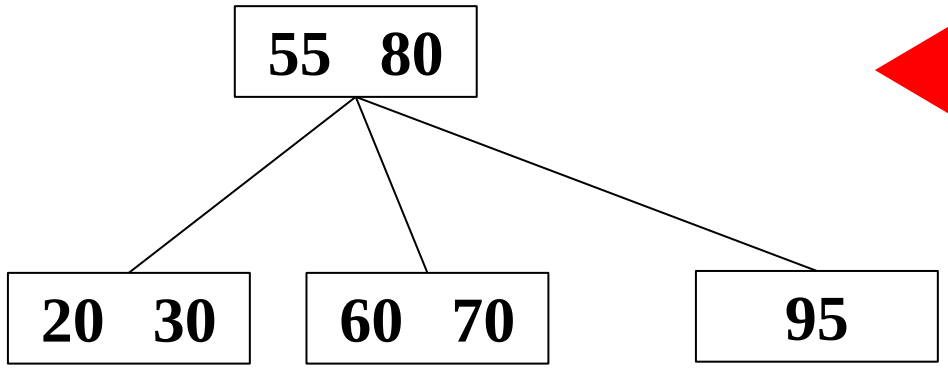
对如下3阶B树删除关键字40

3阶B树非根结点最少包含1个关键词







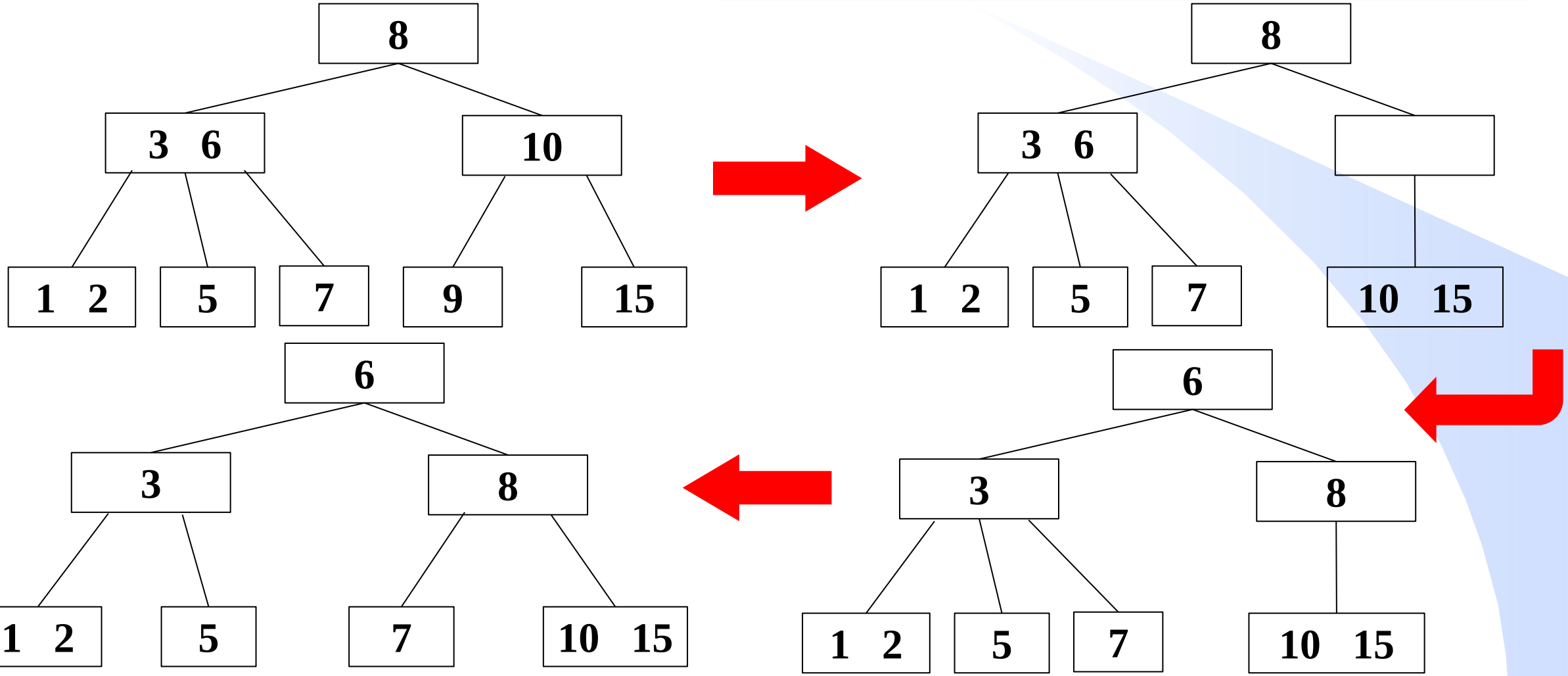


练习题

A

对如下3阶B树删除关键字9

3阶B树非根结点最少包含1个关键词



回顾：B树的删除

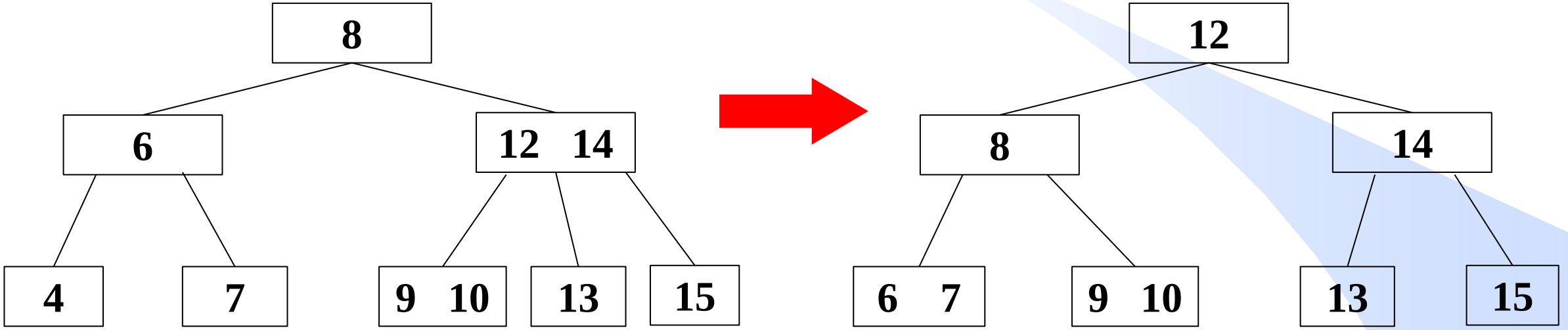
A

- 若借位不是发生在最底层：从左（右）兄弟结点中借最大（最小）关键词的同时，须连同左（右）兄弟结点最右（左）方的子树一起借。

课下练习

A

对如下3阶B树删除关键字4



3阶B树非根结点最少包含1个关键词

B树的高度

B

- B树的高度由叶结点的深度决定。
- 包含n个关键词的B树，有$n+1$个叶结点。

  n个关键词 $\leftrightarrow n$ 种成功查找 $\leftrightarrow n+1$种失败查找 $\leftrightarrow n+1$个叶结点
- 假设叶结点在第k层
- 各层的结点数目
  - ✓第0层为根，第1层至少2个结点，
  - ✓第2层至少 $2\cdot\lceil m/2\rceil$ 个结点
  - ✓第3层至少 $2\cdot\lceil m/2\rceil^2$个结点
  - ✓第4层至少 $2\cdot\lceil m/2\rceil^3$个结点
  - ✓……
  - ✓第k层至少 $2\cdot\lceil m/2\rceil^{k-1}$个结点，

$$n+1\geq 2\cdot\lceil m/2\rceil^{k-1},$$

$$k\leq 1+\log_{\lceil m/2\rceil}\left(\frac{n+1}{2}\right)$$

$$=O(\log_m n)$$

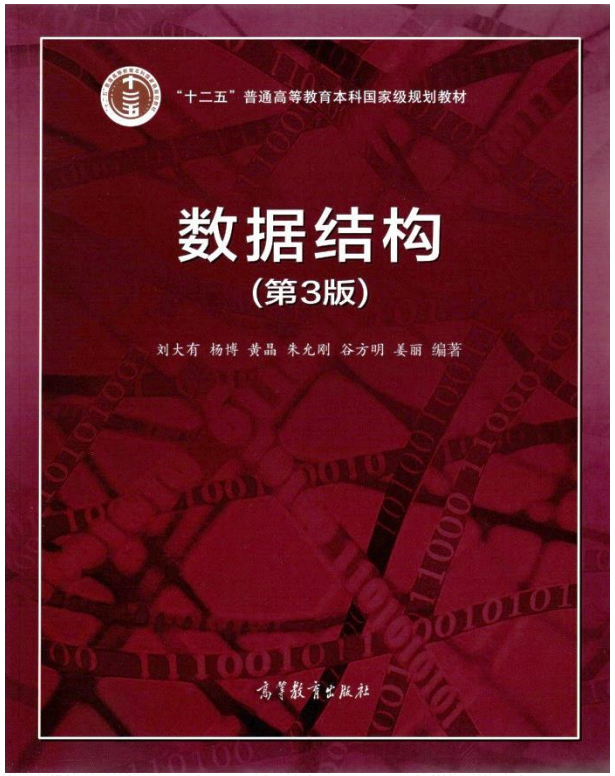
m多大为宜

B

- 磁盘块是操作系统读写外存的最小单位。m取值与磁盘块大小相关，尽量让B树每个结点的大小接近一个磁盘块的大小。
- 一般多数数据库系统采用 m=200~300。
- 例如，若关键词个数$n=2\times10^{10}$ 且 m=300 时，k最多是5。意味着在最坏的情况下，一次查找至多需要5次外存访问。

$$k \le 1 + \log_{\lceil m/2 \rceil}\left(\frac{n+1}{2}\right)$$

- 若将文件组织成一棵二叉查找树，查找所需平均外存访问次数约为 $\log_2 n \approx 35$ 次。



多叉查找树

- B树
- **B+树简介**
- 数字查找树/字典树

数据之法
结构之美
算法之道

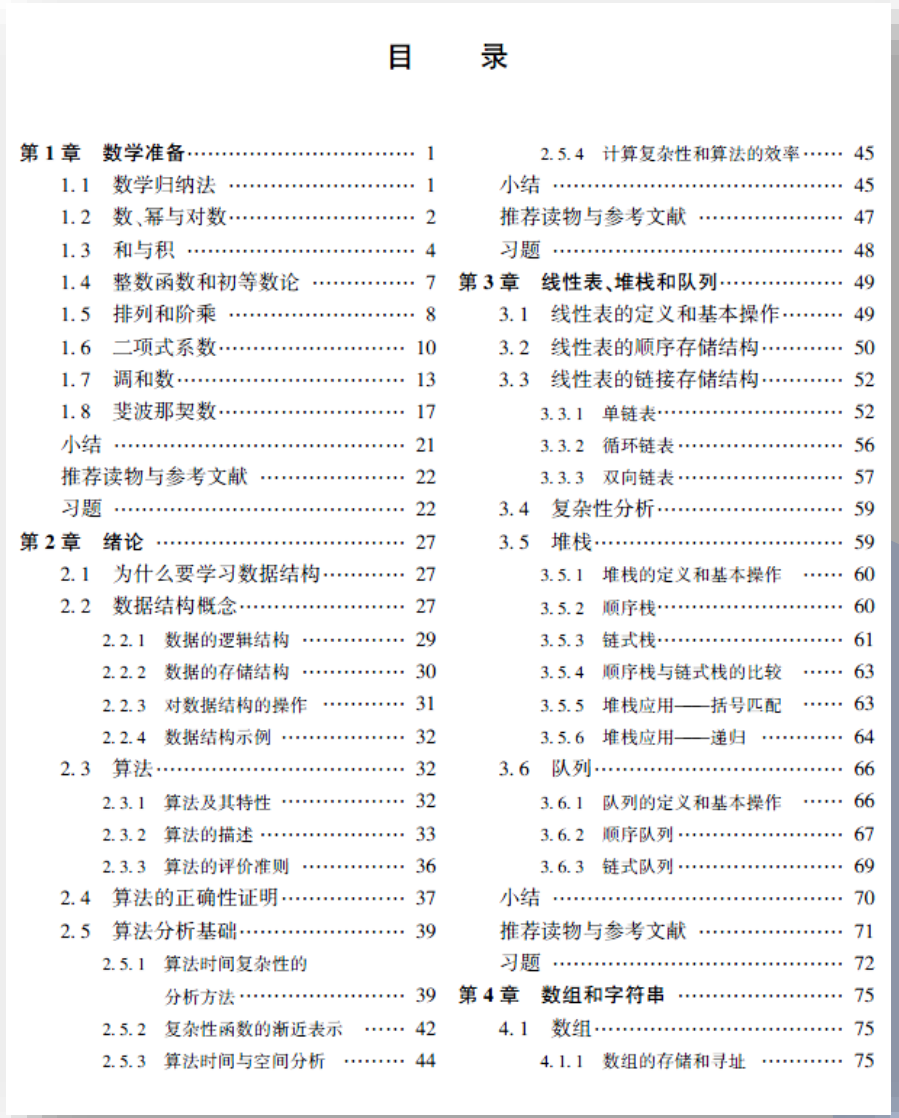
zhuyungang@jlu.edu.cn

Last updated on 2023.12

B+ 树

C

- 目标：能否进一步减少外存访问次数？
- 策略：引入索引



目　录

B+ 树

C

一棵m阶B+树须满足下列条件：

① 每个结点至多有m个子结点；

② 除根结点外，其它每个非叶结点至少有$\lceil m/2 \rceil$个子结点；

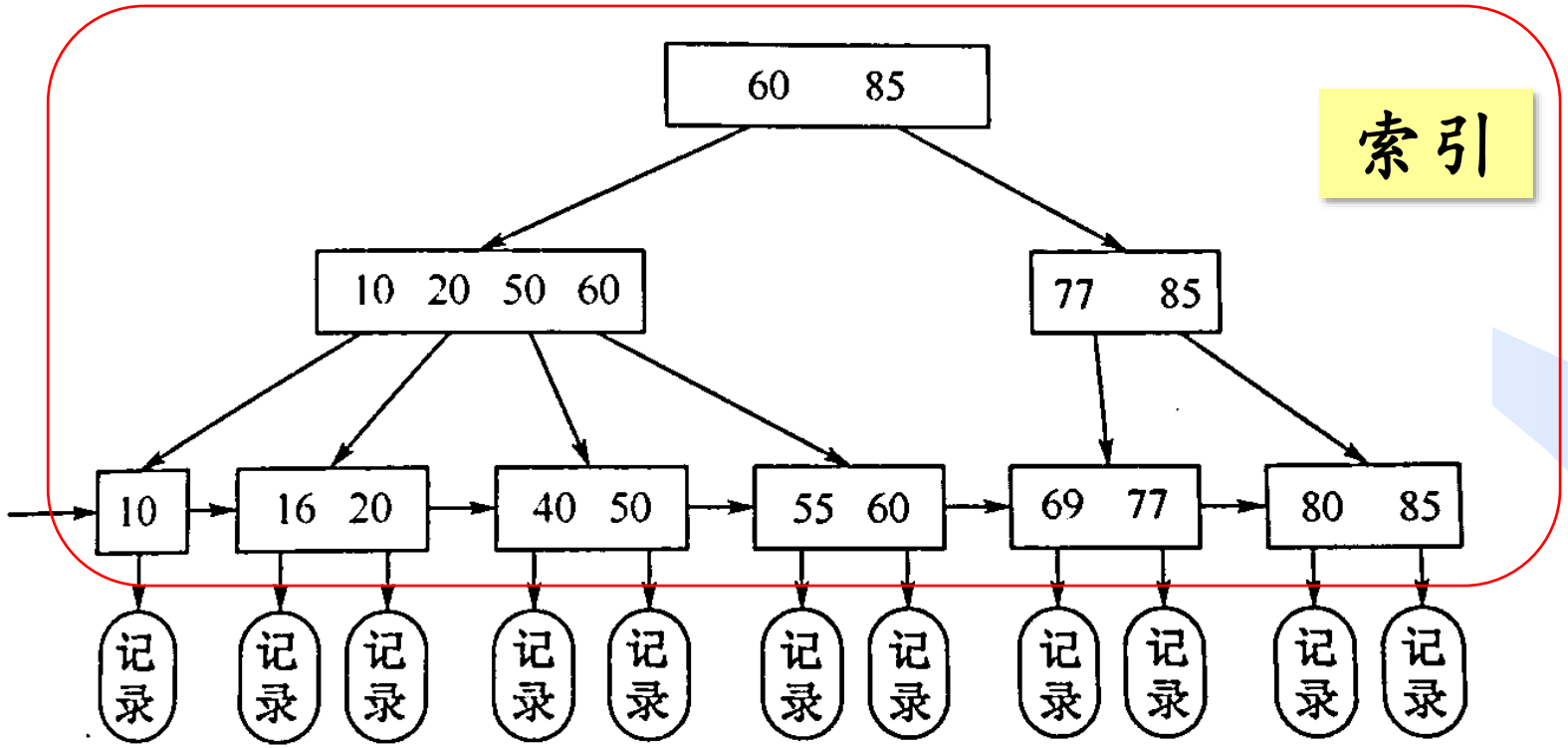
③ 根结点至少有两个子结点；

④ 包含n个孩子的结点有n个关键词（即每个关键词对应一棵子树）；

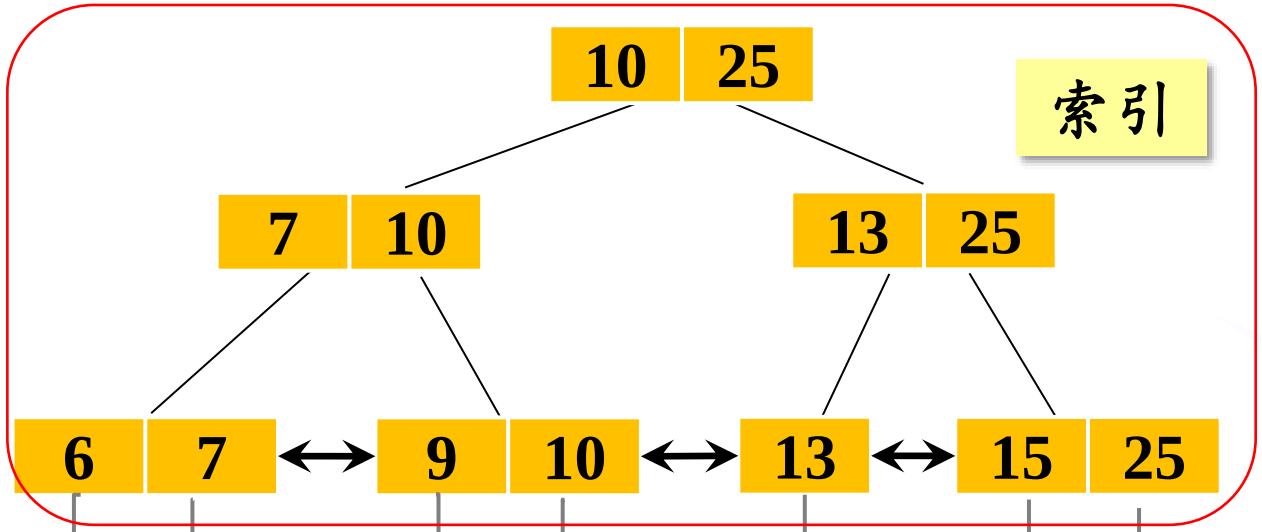
⑤ 每个非叶结点仅包含各孩子的最大关键词及指向各孩子的指针（每个非叶结点不存放实际的数据记录，可看成索引）；

⑥ 所有叶结点包含全部关键词及指向相应数据记录的指针，且叶结点按关键词递增的顺序连成一个链表（每个叶结点的指针指向磁盘上实际数据文件中的记录，也可看成索引）。

一棵4阶B+ 树

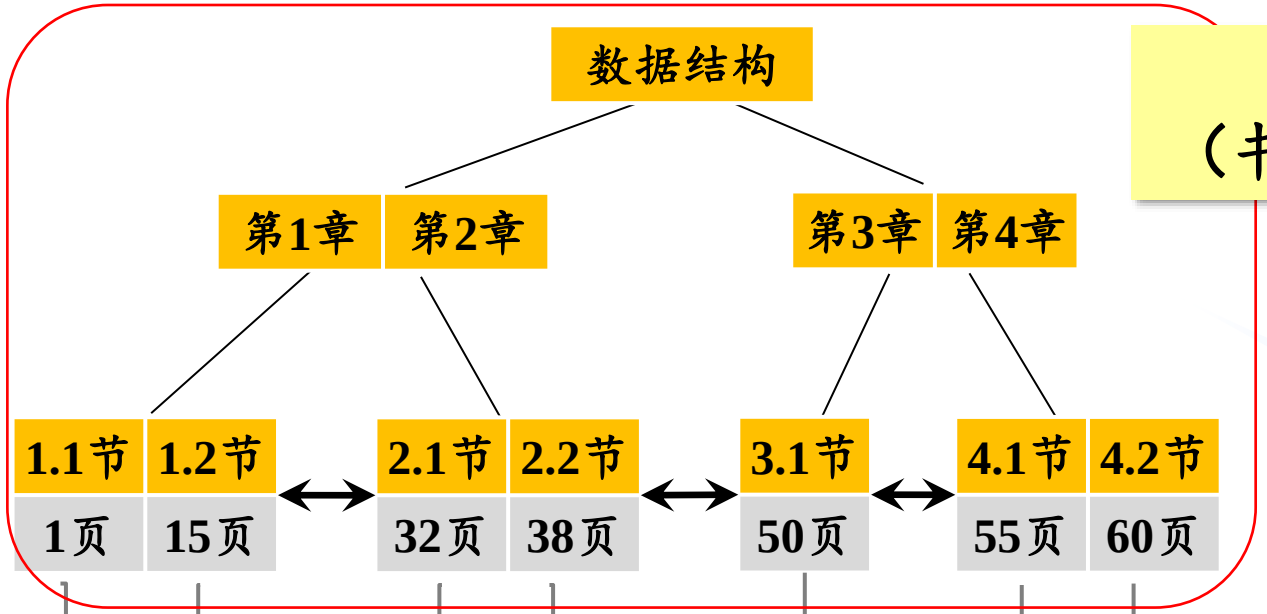


- 每个关键词对应一棵子树；
- 每个非叶结点仅包含各孩子的最大关键词及指向各孩子的指针，不存放实际的数据记录。
- 所有叶结点包含全部关键词及指向相应数据记录的指针，且叶结点按关键词递增顺序连成链表（便于顺序访问和区间查找）



存储在磁盘的
实际数据记录

ID	Name	Age	gender	…
6	Bob	18	Male	…
7	Sunny	15	Female	…
9	Mike	20	Male	…
10	Peter	9	Male	…
13	Proro	8	Male	…
15	John	12	Male	…
25	Mary	17	Female	…



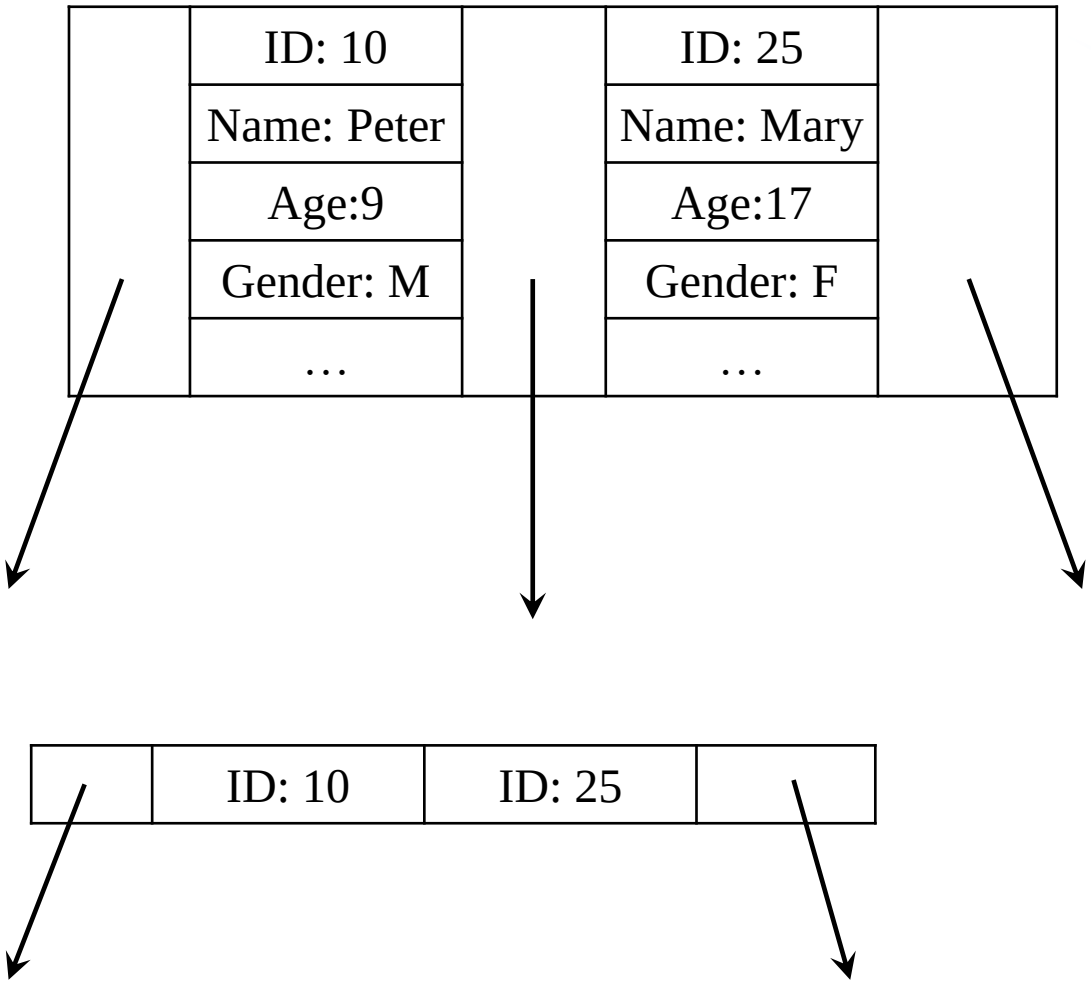
索引
（书的目录）

存储在磁盘的
实际数据记录
（书的正文）

节	内容
1.1	…
1.2	…
2.1	…
2.2	…
3.1	…
4.1	…
4.2	…

B树结点 vs B+树结点

C



B树：每个结点存储关键词及其对应的实际数据记录（或指针）

B+树：只存关键词

*m*阶B+树 vs *m*阶B树

C

	B+ 树	B树
有*n*棵子树的结点中含有关键词个数	*n*个	*n*–1
除根外的每个结点包含关键词个数	$[\lceil m/2 \rceil, m]$	$[\lceil m/2 \rceil - 1, m-1]$
根结点包含关键词个数	$[1, m]$	$[1, m-1]$
结点与关键词	所有叶结点包含了全部关键词，即非叶结点中的关键词都包含在叶结点中	叶子结点包含的关键词与其它结点包含的关键词是不重复的
头指针个数	两个：一个指向根结点；另一个指向最小关键词的叶结点，所有叶结点链接成一个链表。	一个：根结点指针
记录存储	非叶结点只含有对应孩子的最大关键词和指向该孩子的指针，不含有该关键词对应记录的存储地址。	每个非叶结点存储关键词及其对应的实际数据记录

B树 vs B+树

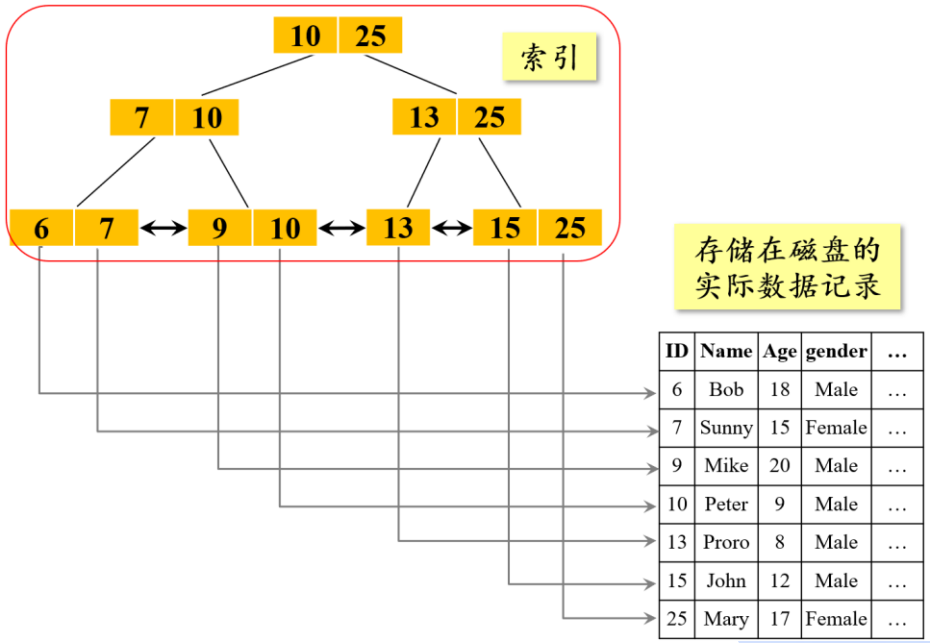
C

- B+树进行两种查找运算：一种是从最小关键词开始进行顺序查找，另一种是从根结点开始进行树形查找。
- 在B+树上进行随机查找、插入和删除操作的过程基本上与B树类似。只是在查找时，如果非叶结点上的关键词等于给定值，并不终止，而是继续向下直到叶结点。因此，在B+树中，不管查找成功与否，每次查找都是走了一条从根到叶结点的路径。

B树 vs B+树

C

- 若把索引表放入内存，经过对索引的查找后确定数据记录的实际存储地址，再经过 1 次磁盘访问，读取记录，就可以完成查找。
- 若索引很大，也可放在磁盘。由于索引结点只存关键词，在结点所占空间相等的情况下，B+每个结点存的关键词个数多于B树，使得B+树更宽更矮，磁盘访问次数更少。



ID	Name	Age	gender	…
6	Bob	18	Male	…
7	Sunny	15	Female	…
9	Mike	20	Male	…
10	Peter	9	Male	…
13	Proro	8	Male	…
15	John	12	Male	…
25	Mary	17	Female	…

B树/B+树的应用场景

D

- 文件系统
  - ✓Windows NTFS
  - ✓Apple HFS+/APFS
  - ✓Linux EXT4
  - ✓……
- 数据库系统
  - ✓Oracle
  - ✓SQL Server
  - ✓DB2
  - ✓MySQL
  - ✓……

APFS
Apple File System

NTFS

ORACLE
DATABASE

Ext4
File System

IBM
DB2

MySQL

Microsoft
SQL Server

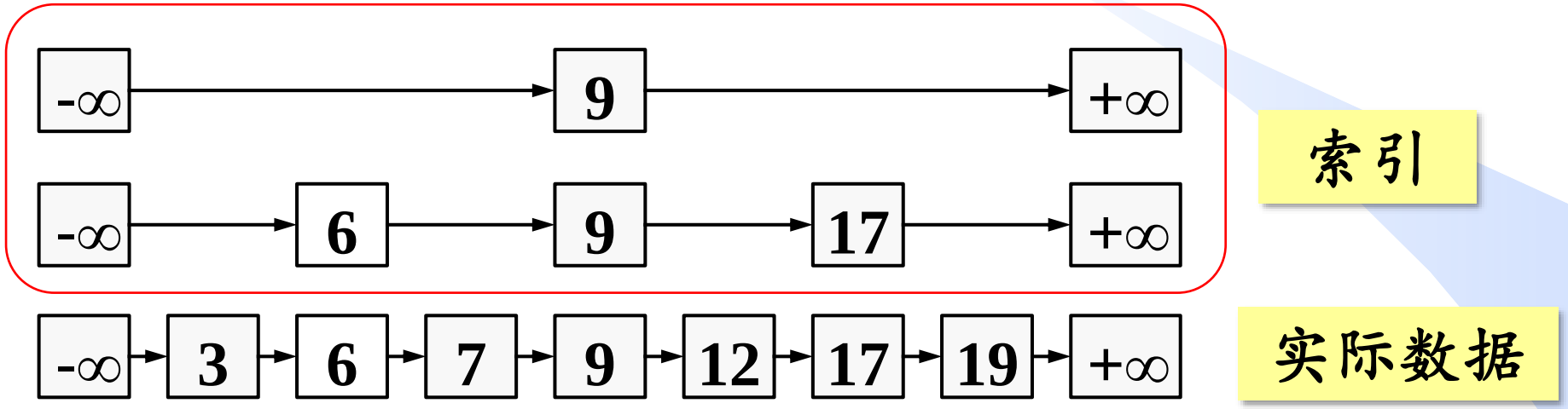
练习题

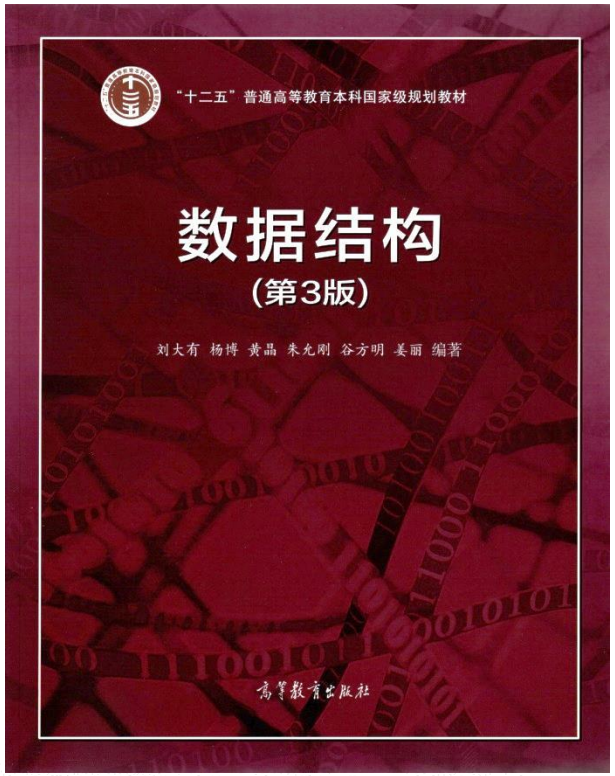
C

B+树不同于B树的特点之一是__A__。【考研题全国卷】

A. 能支持顺序查找
B. 结点中含有关键字
C. 根结点至少有两个分支
D. 所有叶结点都在同一层上

跳跃表 vs B+树





多叉查找树

- B树
- B+树简介
- **数字查找树/字典树**

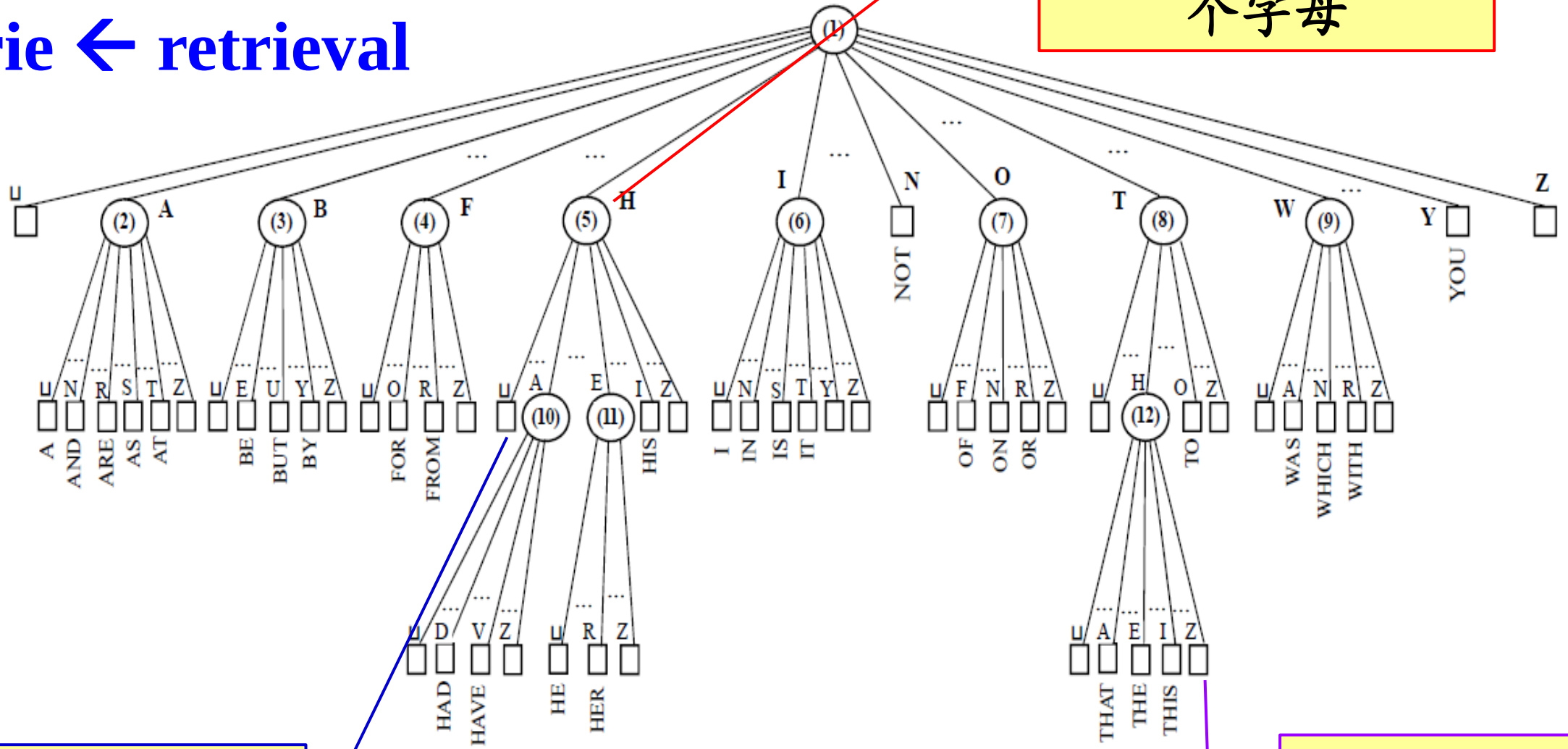
数据之法
结构之美
算法之道

zhuyungang@jlu.edu.cn

Last updated on 2023.12

字典树/Trie树

Trie ← retrieval



第1层对应K的第1个字母

第2层对应K的第2个字母

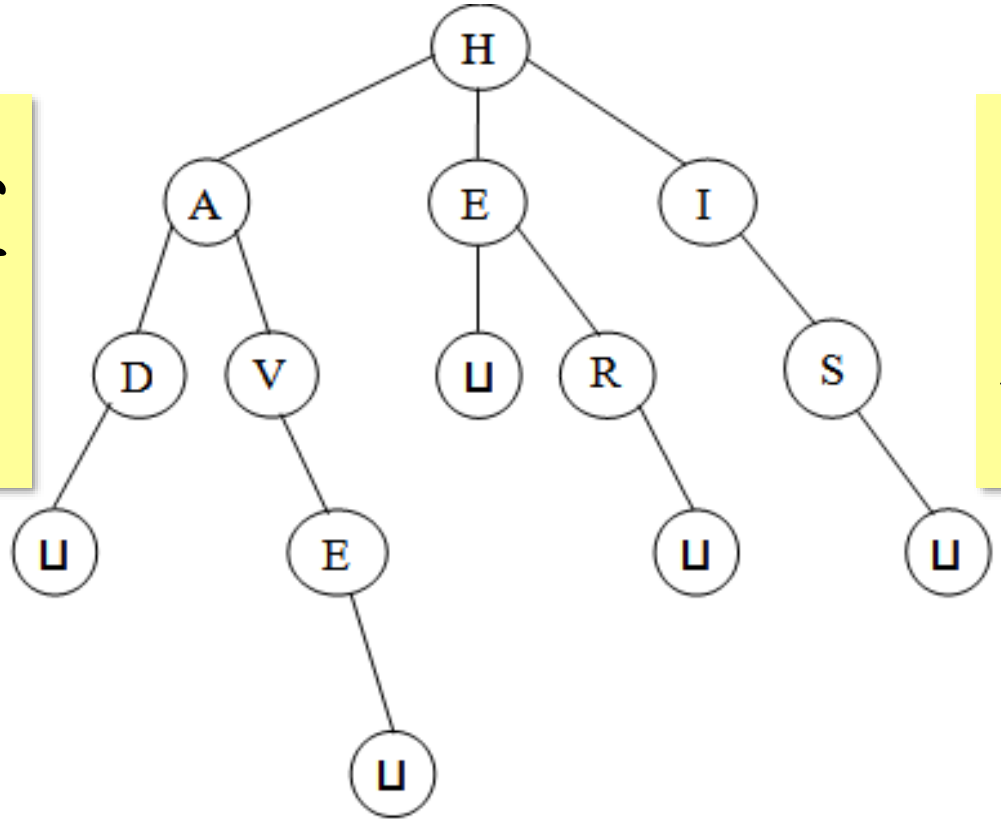
第3层对应K的第3个字母

若关键词K表示成字母序列

C

C

关键词不是结点，而是从根到叶的路径。



空格符“␣”作为关键词的结束符。

如果关键词是字母序列，称字典树/Trie树
如果关键词是数字序列，亦称数字查找树

典型应用场景——字典

C

- 常被用于保存、统计和排序大量的字符串。

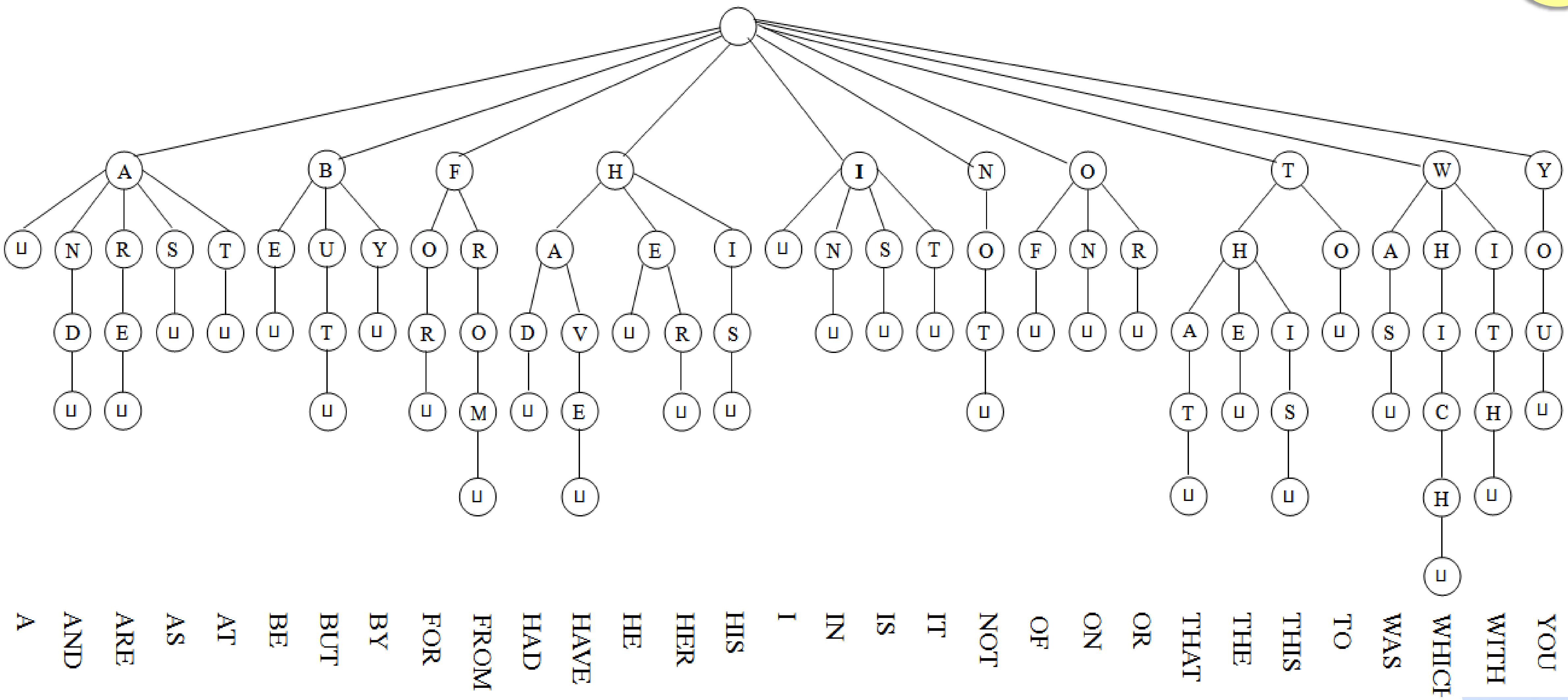
- 查找方法为：

(1)从根结点开始查找；

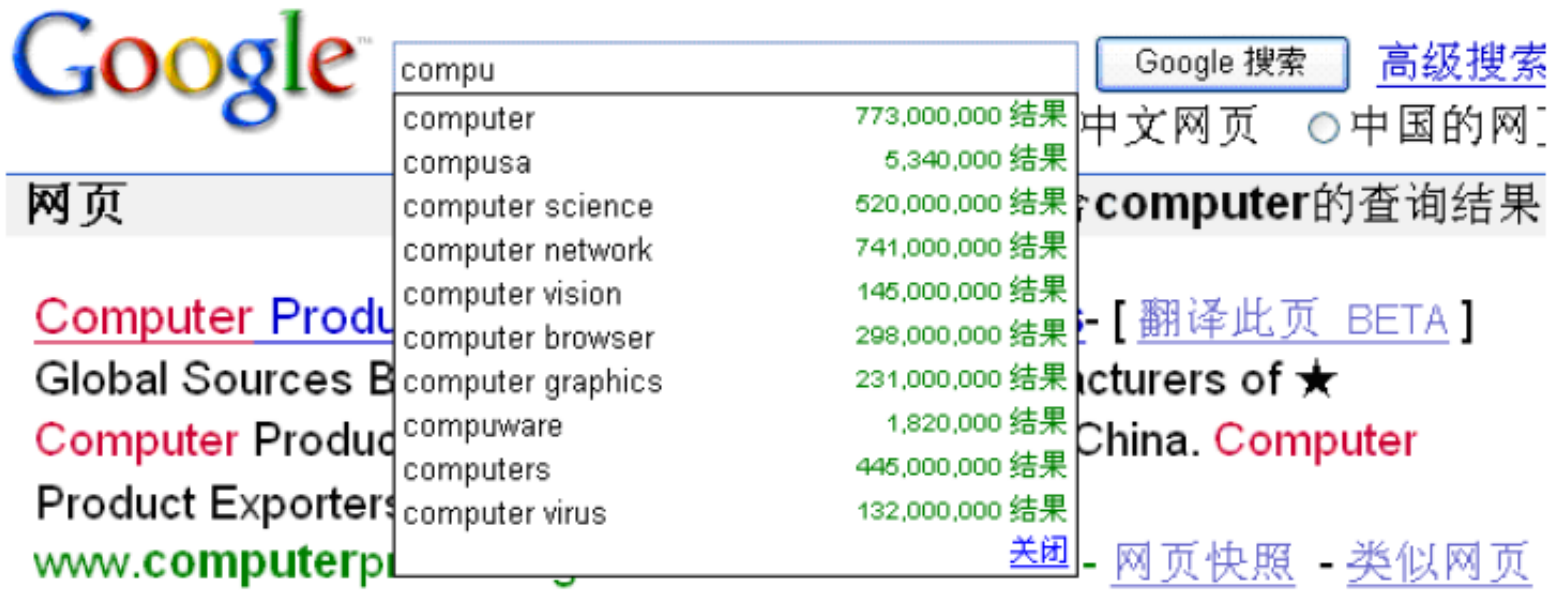
(2)取得要查找关键词的第1个字母，并根据该字母选择对应的子树并转到该子树继续进行查找；

(3)在相应的子树上，取得要查找关键词的第2个字母，并进一步选择对应的子树进行检索。

(4)迭代上述过程……直至在某个结点处，关键词的所有字母已被取出且该结点对应的字母是某一单词的最后一个字母，则完成查找。

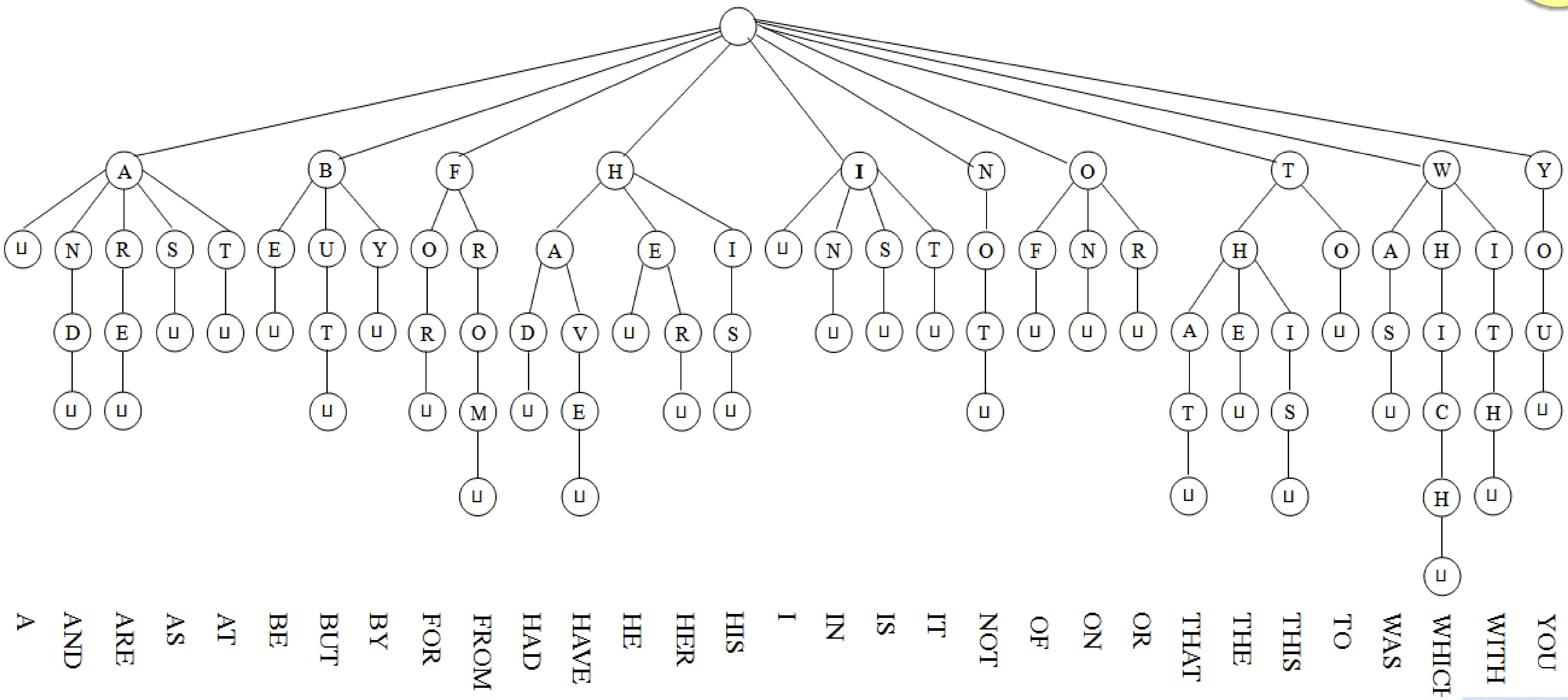

A
AND
ARE
AS
AT
BE
BUT
BY
FOR
FROM
HAD
HAVE
HE
HER
HIS
I
IN
IS
IT
NOT
OF
ON
OR
THAT
THE
THIS
TO
WAS
WHICH
WITH
YOU

搜索引擎的自动补全功能



字符串的前缀匹配

搜索引擎的自动补全功能



搜索引擎的自动补全功能

C

搜索引擎的自动补全功能

C

新闻 网页 贴吧 知道 音乐 图片 视频 地图 百科 文库 更多>>

shua

百度一下

双色球开奖结果
双色球
双色球走势图
双视影院
刷机精灵
双视
刷新3+7