B树（B⁻树）——平衡多路树

① 物理层面

⇓

实现高效I/O

i> 平衡二叉树查找效率高——通过降低树的深度

若数据量太大导致平衡二叉树深度太大而造成查找效率下降。

ii> 由于AVL的深度太大导致在进行磁盘I/O时读写频繁

② 适用场景：通常适用于数据库索引机制，或文件系统在读写相对较大的数据块的存储系统中。

B 树特点：

A> 是一个平衡的多叉树。（平衡多叉树 / 每个结点可有多于 2 个子树）

B> m 阶 B 树（树性：跳叶子）

i> m 阶：每个结点最多可以有 m 个子结点。

ii> 每个非叶子结点（除根）至少有 $\lceil \frac{m}{2} \rceil$ 个子结点。

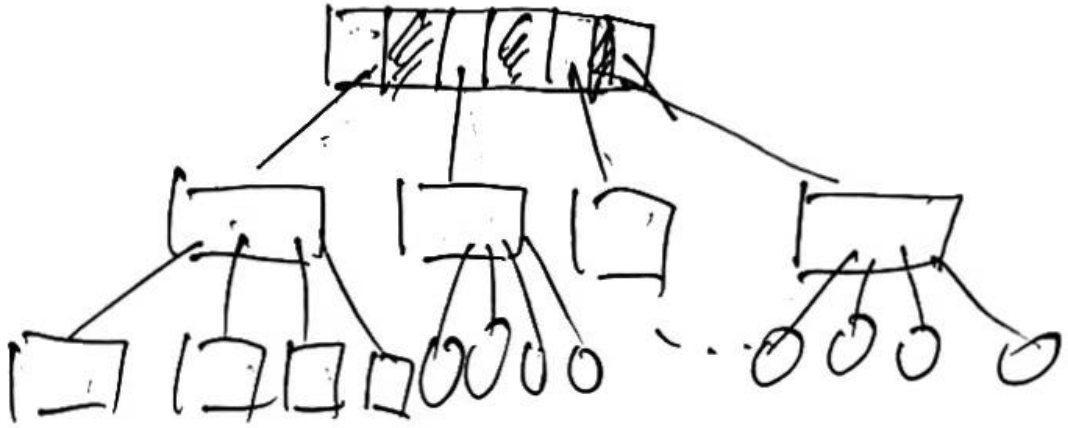
iii> 若根不是叶子，至少有 2 个子结点

iv> 具有 k 个子结点的非叶结点包含 k-1 个键。

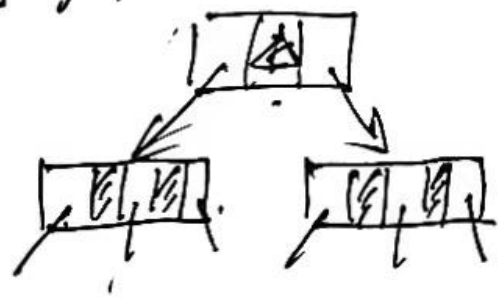
v> 所有叶子结点均处在同一层

☆ 每个结点的 k-1 个关键字有序排列

4阶：最多



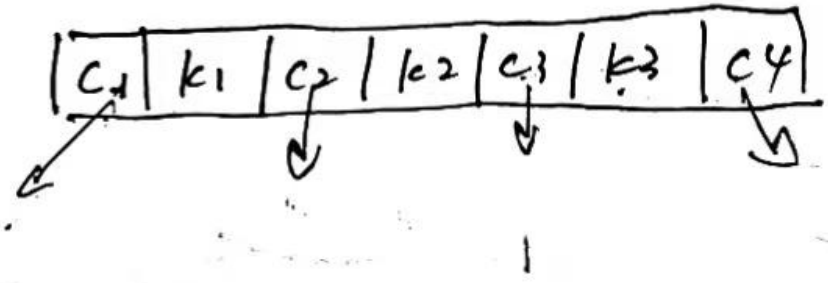
最少：



k_i：关键字

c_i：指向子树

结点构成：



m 阶 B 树

① 根结点：
子结点个数：$2 \le M \le m$
关键字个数：$1 \le key \le m-1$

② 非根非叶结点（内部结点）：
子结点个数：$\lceil \frac{m}{2} \rceil \le M \le m$
关键字个数：$\lceil \frac{m}{2} \rceil - 1 \le key \le m-1$

③ 叶结点：
子结点个数：0
关键字个数：$\lceil \frac{m}{2} \rceil - 1 \le key \le m-1$

B 树的插入：

① 若结点关键字个数小于 m-1 个，直接插入；

② 若结点关键字个数大于 m-1 个，进行结点分裂，

⇒ 向中间元素插入到父结点中。

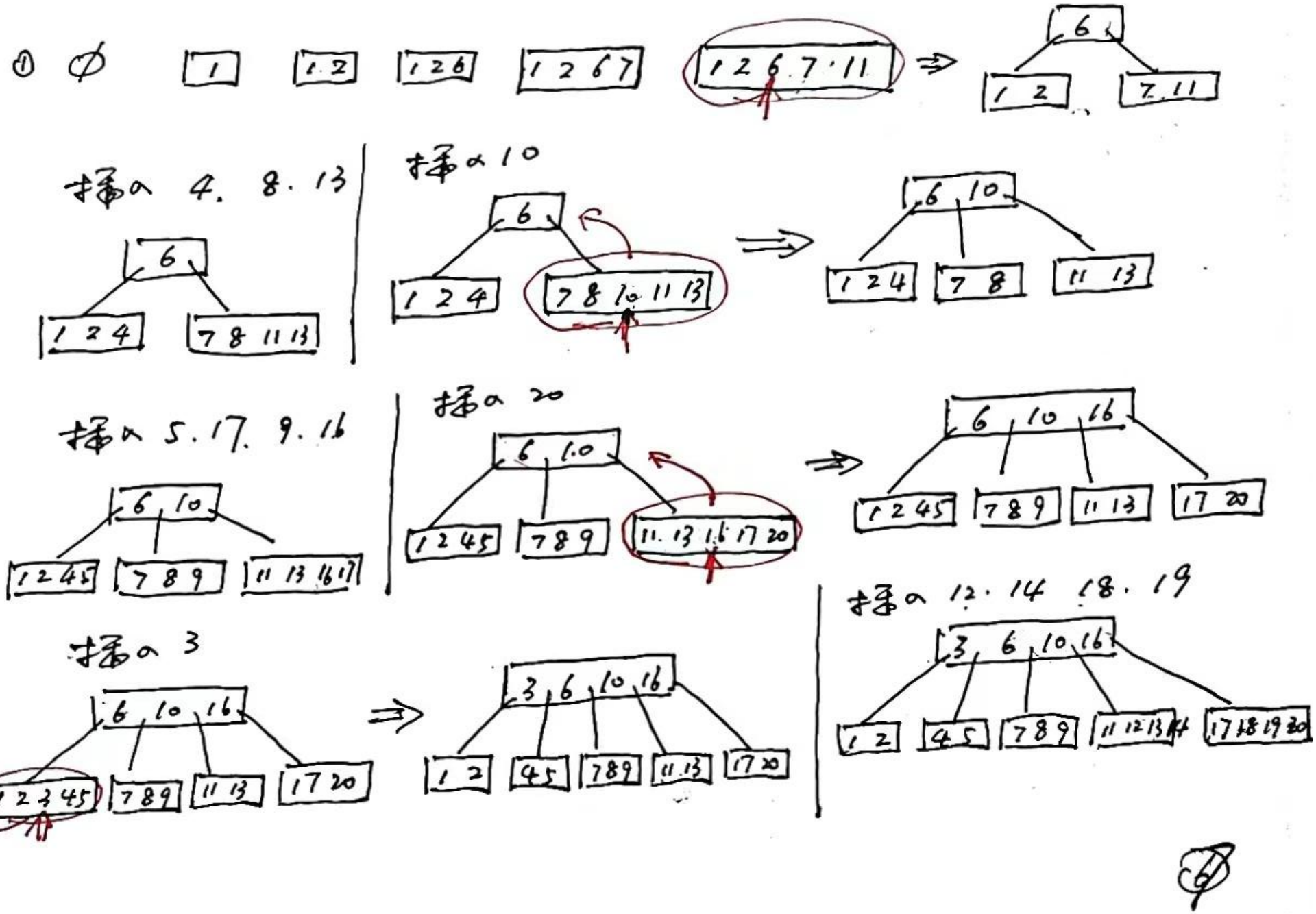
☆ 若分裂后的 B 树，中间元素，可以是靠左的最后一个或后者的第一个

☆ 最坏情况下，需向上进行多次分裂。

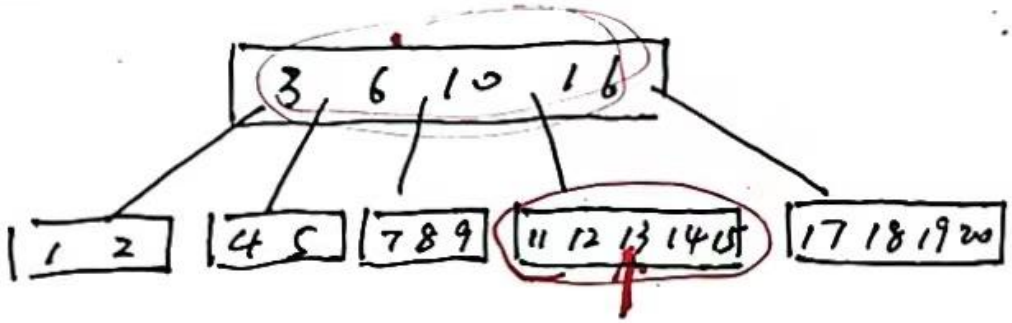
例：{1. 2. 6. 7. 11. 4. 8. 13. 10. 5. 17. 9. 16. 20. 3. 12.

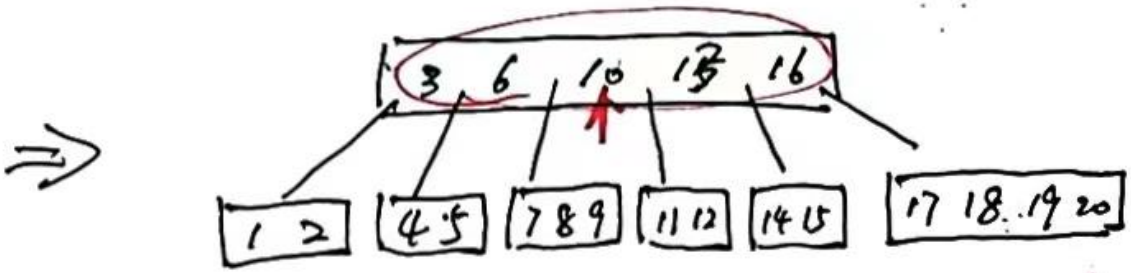
14. 18. 19. 15.} 构造一棵5阶B树.

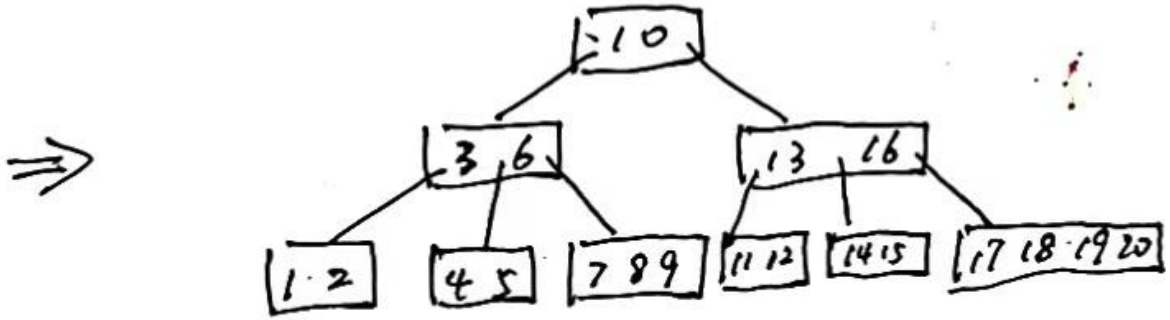
5阶B树
- 根：$2 \le M \le 5$，$1 \le key \le 4$
- 内部：$3 \le M \le 5$，$2 \le key \le 4$
- 叶子：$M = 0$，$2 \le key \le 4$


① ∅
1
1 2
1 2 6
1 2 6 7
1 2 6 7 11
6
1 2
7 11
插入 4. 8. 13
6
1 2 4
7 8 11 13
插入 10
6
1 2 4
7 8 10 11 13
6 10
1 2 4
7 8
11 13
插入 5. 17. 9. 16
6 10
1 2 4 5
7 8 9
11 13 16 17
插入 20
6 10
1 2 4 5
7 8 9
11 13 16 17 20
6 10 16
1 2 4 5
7 8 9
11 13
17 20
插入 3
6 10 16
1 2 3 4 5
7 8 9
11 13
17 20
3 6 10 16
1 2
4 5
7 8 9
11 13
17 20
插入 12. 14 18. 19
3 6 10 16
1 2
4 5
7 8 9
11 12 13 14
17 18 19 20

插入 15


3 6 10 16
1 2
4 5
7 8 9
11 12 13 14 15
17 18 19 20


3 6 10 13 16
1 2
4 5
7 8 9
11 12
14 15
17 18 19 20

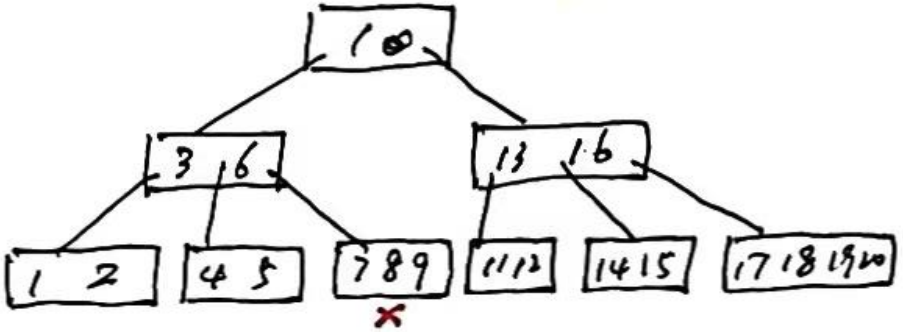

10
3 6
13 16
1 2
4 5
7 8 9
11 12
14 15
17 18 19 20

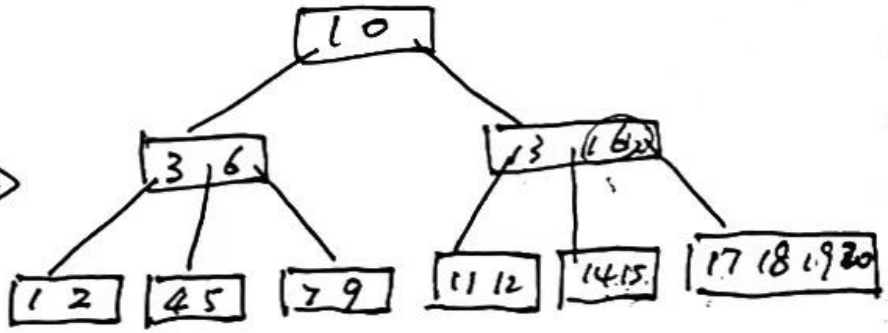
B 树删除（关键字）

① 若删除后，依然保持 B 树形态。

那么直接删除（叶子结点中的关键字） $\lceil \frac{m}{2} \rceil - 1$

例：删除 8。





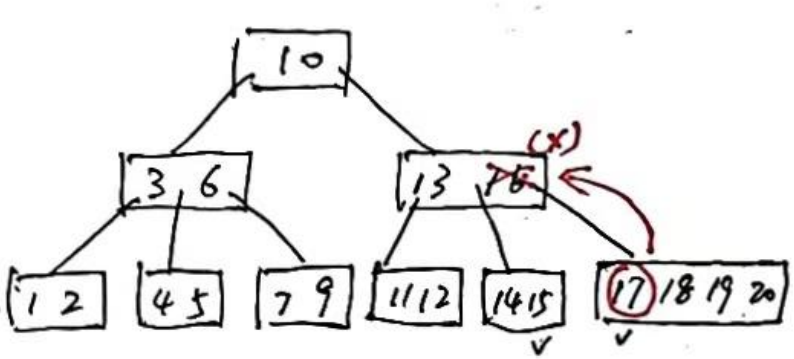
② 删 16。

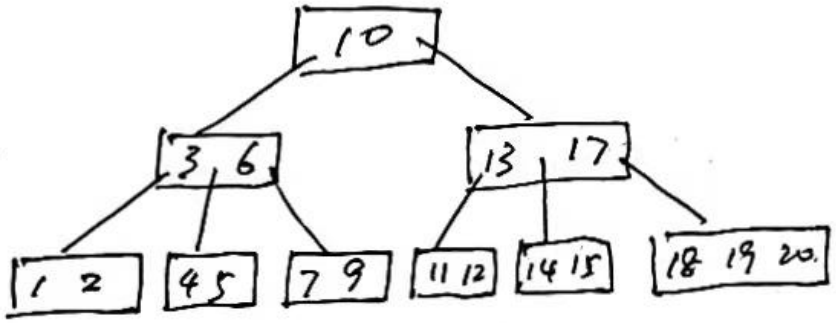
删除非叶子结点中的关键字

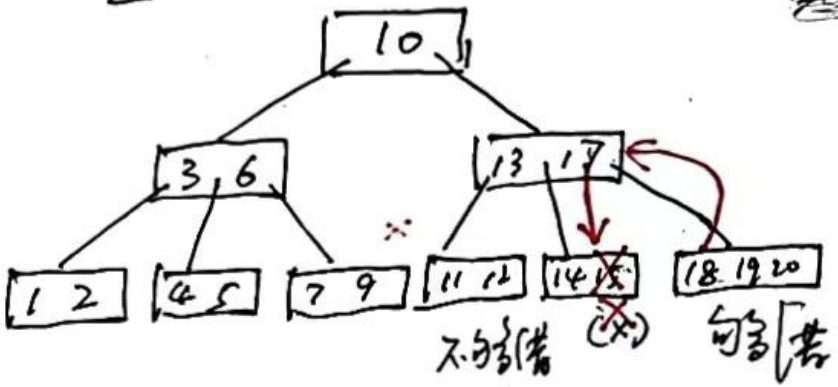
— 寻找关键字的直接前驱或直接后继。

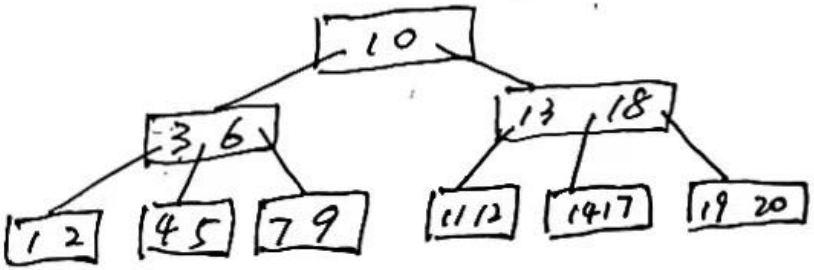
提：删 22 来结点。

☆ 先结点中关键字多于最少关键字个数的结点。


10
3 6
13 16
1 2
4 5
7 9
11 12
14 15
17 18 19 20

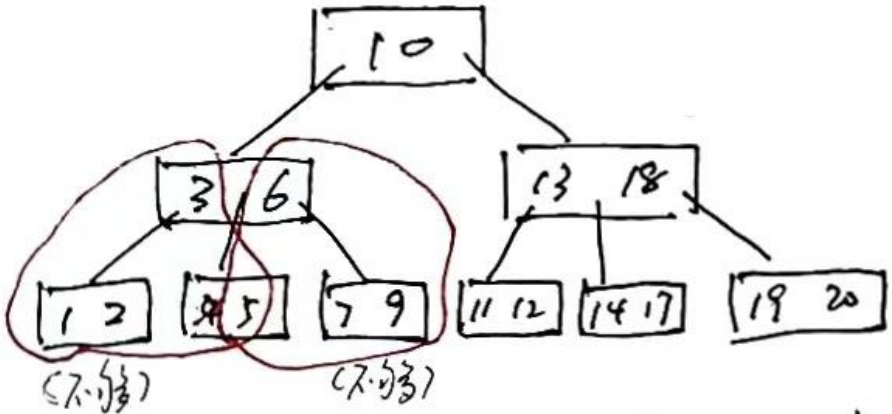

10
3 6
13 17
1 2
4 5
7 9
11 12
14 15
18 19 20


10
3 6
13 17
1 2
4 5
7 9
11 12
14 15
18 19 20
不够借
够借

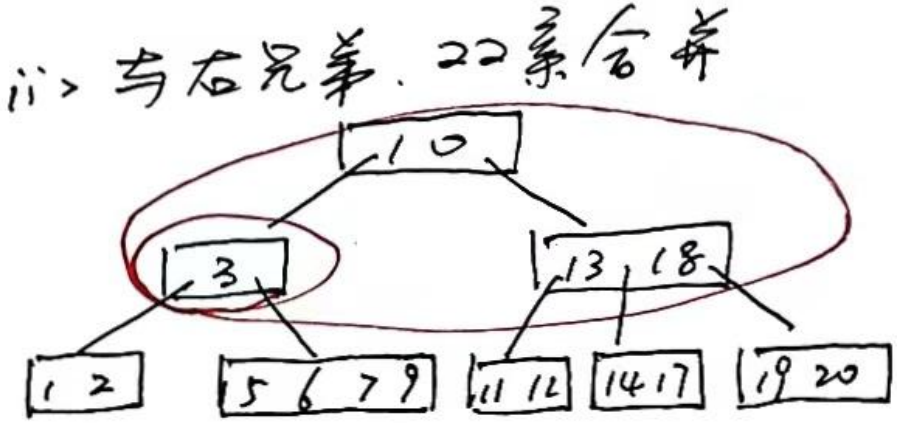

10
3 6
13 18
1 2
4 5
7 9
11 12
14 17
19 20

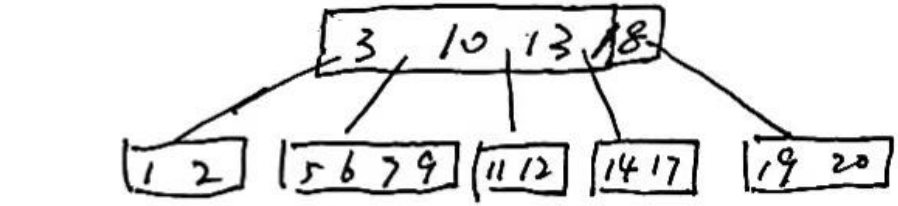
删除 4. ⇒ 删除叶子：左右兄弟都不够借

<1,2>. <7,9> 两兄弟均不够借：除了4与2个亲合并

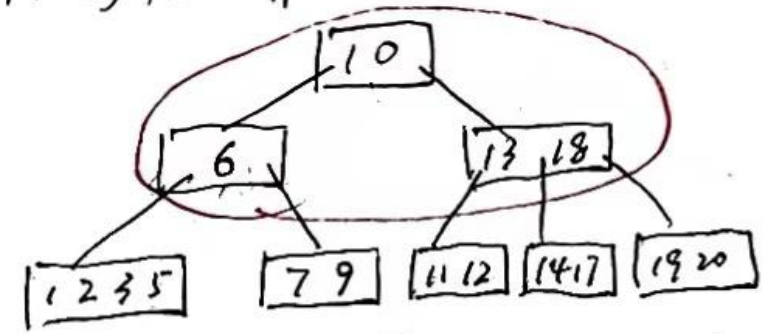


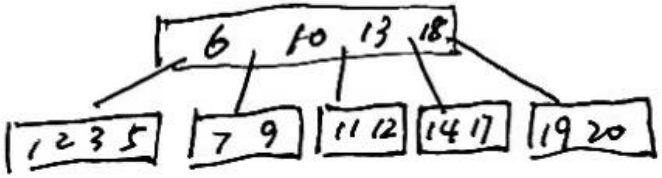
i> 与右兄弟，2个亲合并

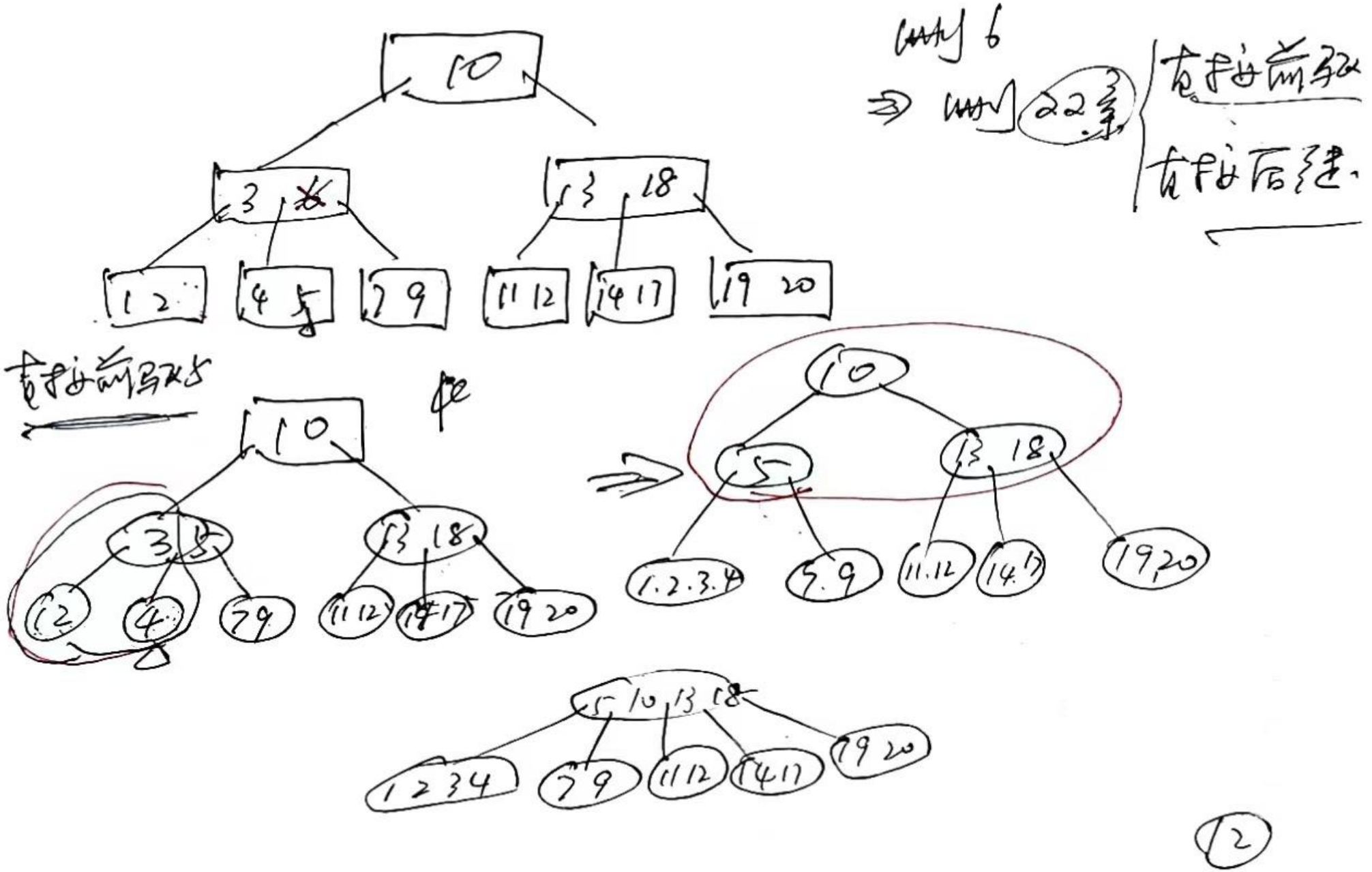


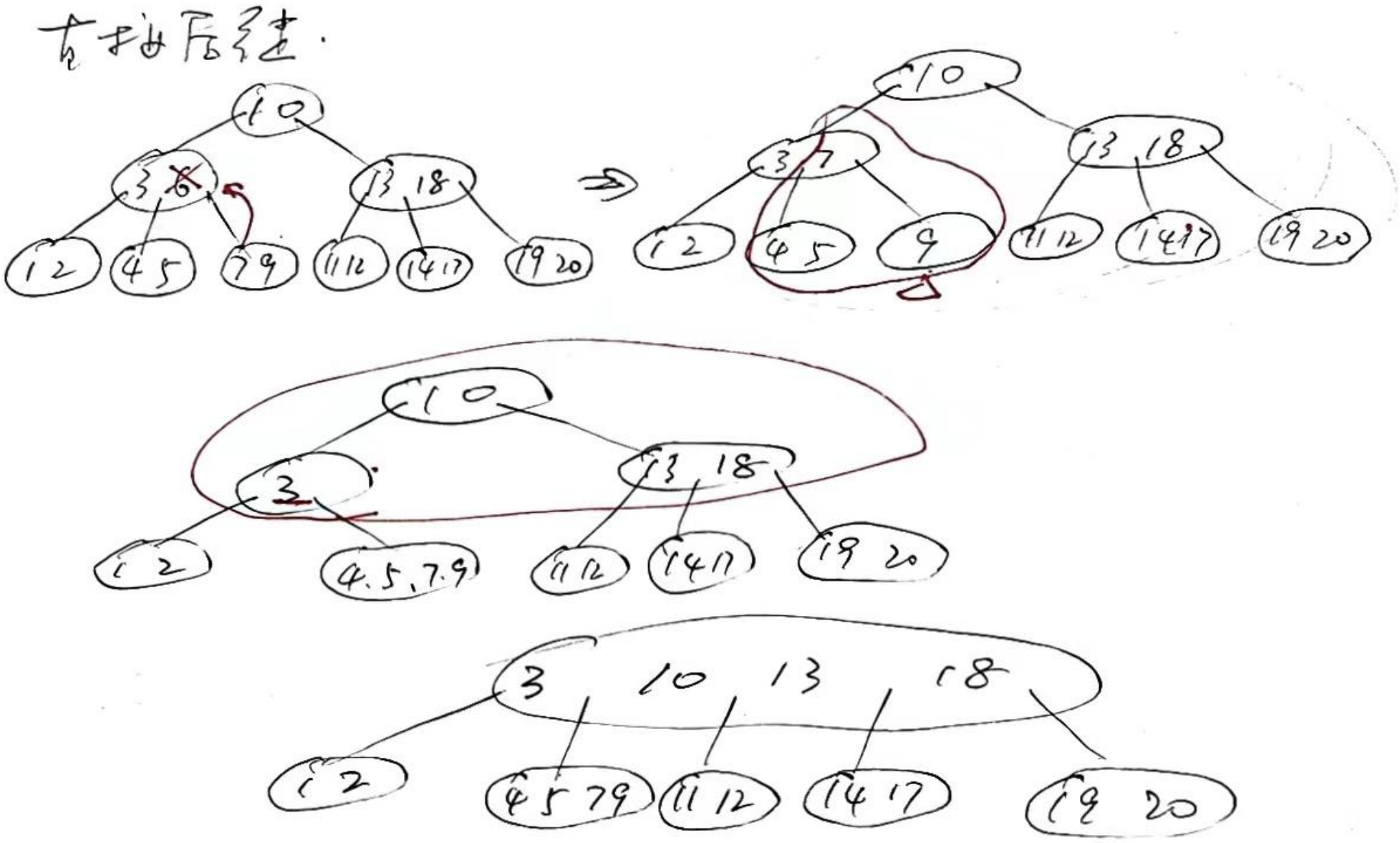


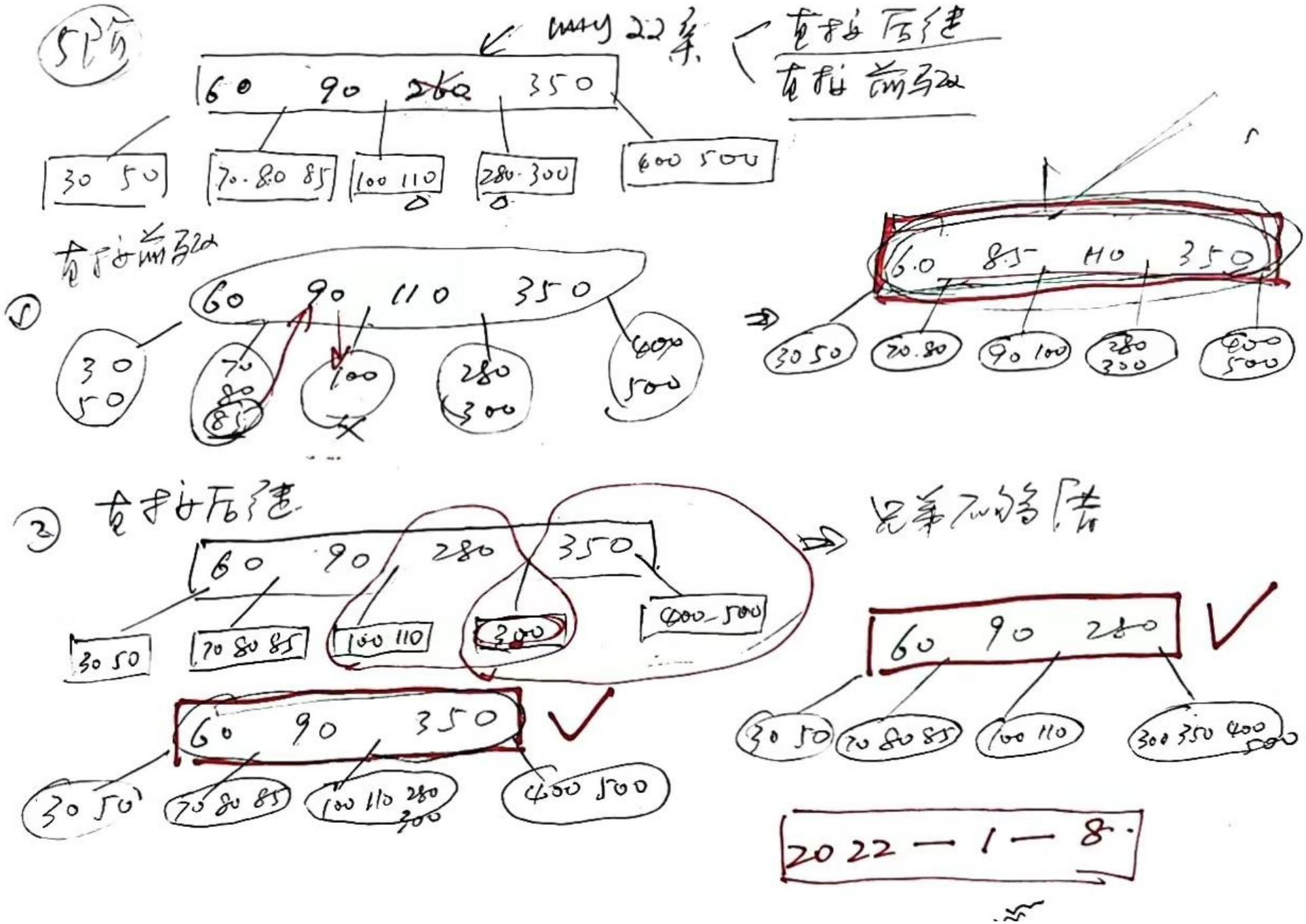
ii> 与右兄弟，2个亲合并






10
3 5
13 18
1 2
4 5
7 9
11 12
14 17
19 20
查找前驱
查找后继
查找前驱
10
3 5
13 18
1 2
4
7 9
11 12
14 17
19 20
10
5
13 18
1 2 3 4
7 9
11 12
14 17
19 20
5 10 13 18
1 2 3 4
7 9
11 12
14 17
19 20

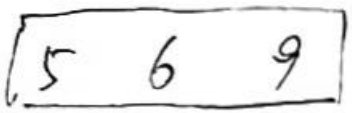

查找后结点.
10
3 6
13 18
1 2
4 5
7 9
11 12
14 17
19 20
10
3 7
13 18
1 2
4 5
9
11 12
14 17
19 20
10
3
13 18
1 2
4.5.7.9
11 12
14 17
19 20
3 10 13 18
1 2
4 5 7 9
11 12
14 17
19 20

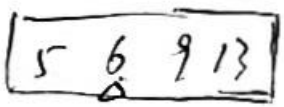

5阶
60 90 280 350
30 50
70 80 85
100 110
280 300
600 500
查找后继
查找前驱
查找前驱
60 90 110 350
30 50
70 80 85
100
280 300
600 500
60 85 110 350
30 50
70 80
90 100
280 300
600 500
查找后继
60 90 280 350
30 50
70 80 85
100 110
300
600 500
60 90 350
30 50
70 80 85
100 110 280 300
400 500
60 90 280
30 50
70 80 85
100 110
300 350 400 500
2022 — 1 — 8

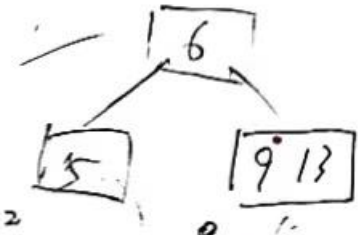
2020-1-10

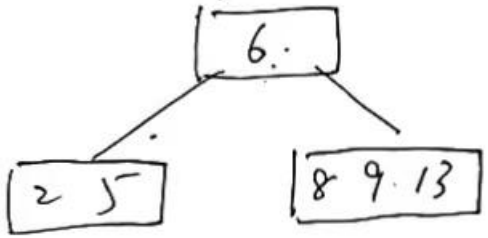
5 6 9 13 8 2 12 15

4阶 根：$1 \le k \le 3$ 内部：$\lceil \frac{m}{2} \rceil - 1 \le k$

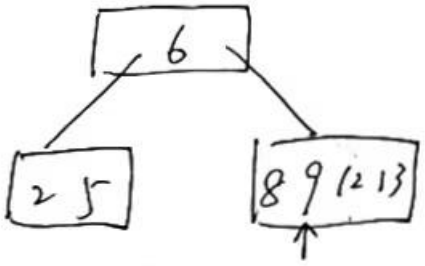




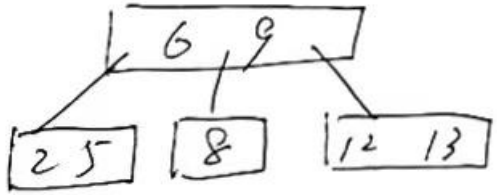


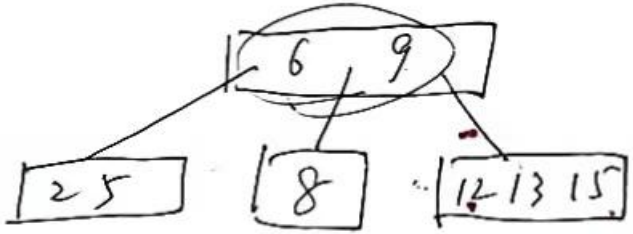


(插 8, 2)



(插 12)





(插 15)

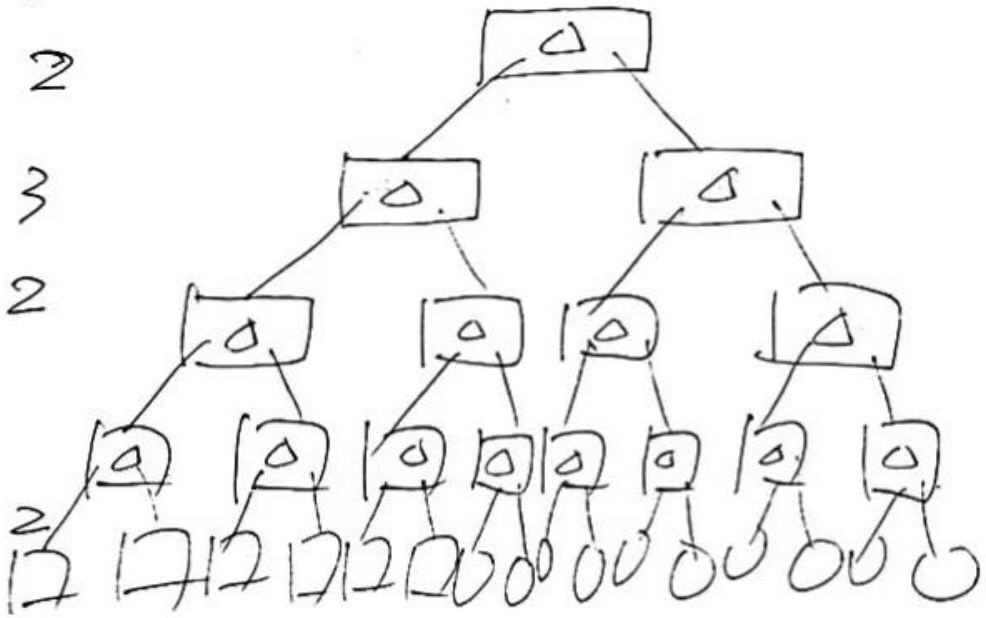
4阶：
- 根：$2 \le M \le 4$，$1 \le k \le 3$
- 内：$2 \le M \le 4$，$1 \le k \le 3$
- 叶：$M \ge 2$，$1 \le k \le 3$

31

3阶 B树 关键字 2,3

叶子结点在同一层

3阶
- 根: $2 \le M \le 3$, $1 \le k \le 2$
- 内部: $2 \le M \le 3$, $1 \le k \le 2$
- 叶子: $M = 0$, $1 \le k \le 2$



2018-1-8

B+ Tree ⇒ 从结构上看，是B-树 ⇓ 变形B树

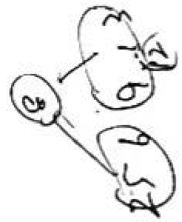
① 每个分支结点最多 m 棵子树

② 非叶根：至少 2 棵子树

分支：至少 $\lceil \frac{m}{2} \rceil$ 子树

根：

非根分支：特点



☆ ③ 结点子树个数与关键字个数相同

④ 叶结点：

i> 全部关键字及指向指针的记录

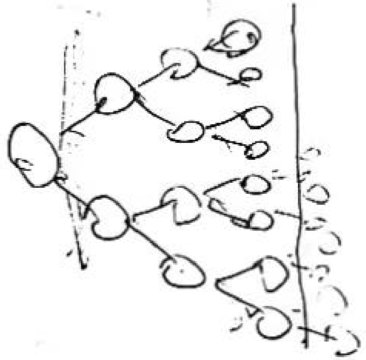
ii> 关键字按照顺序排列

iii> 相邻叶结点按顺序构成一根链表.

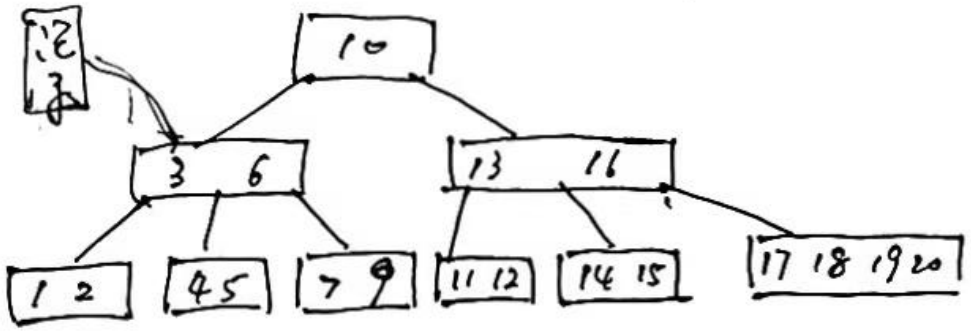
☆

⑤ 分支结点：只包含各子结点的关键字最大值及指向子结点的指针

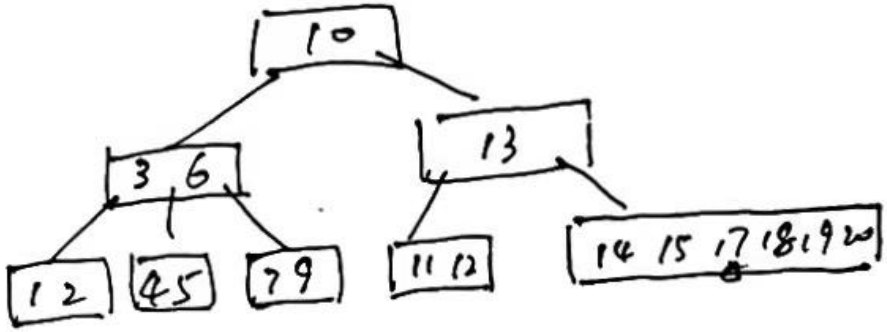
☆

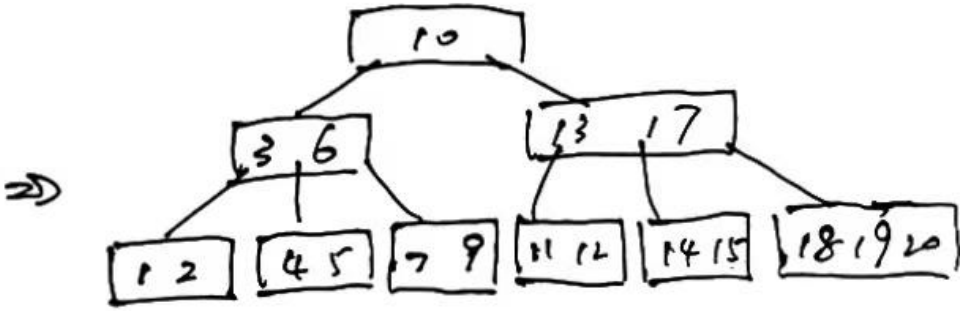


删除 8.

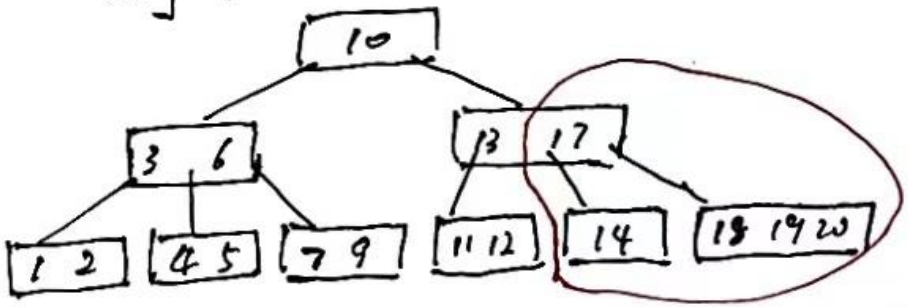


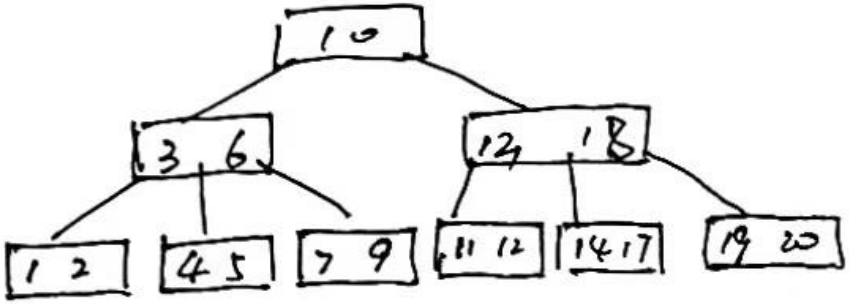
删除 16



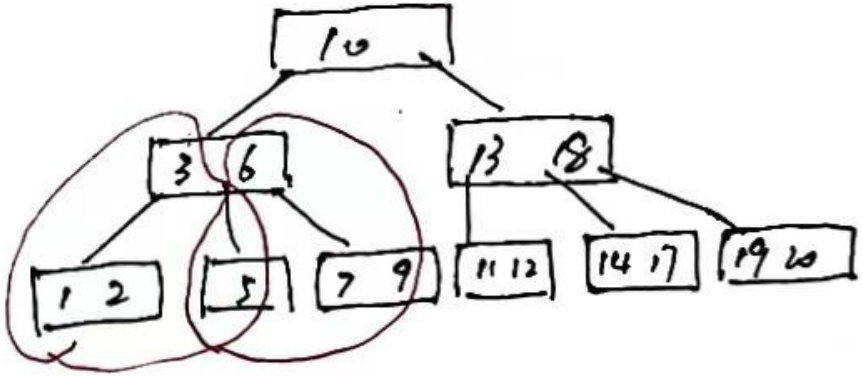


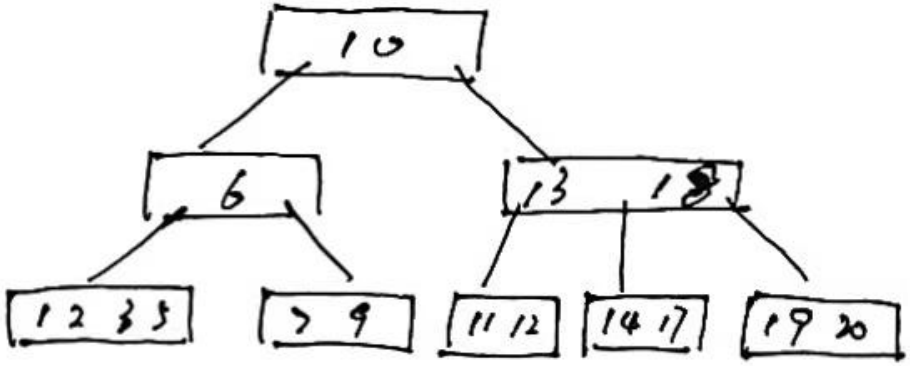
删除 15

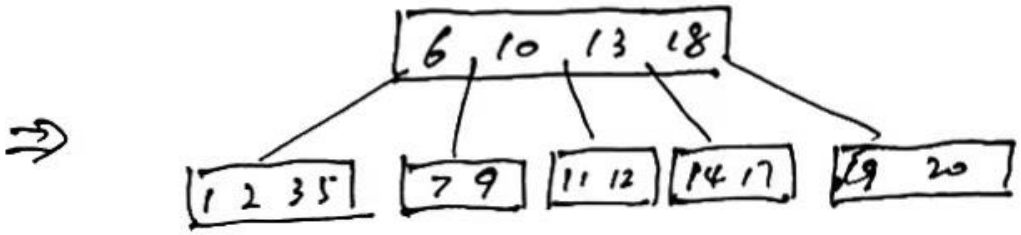


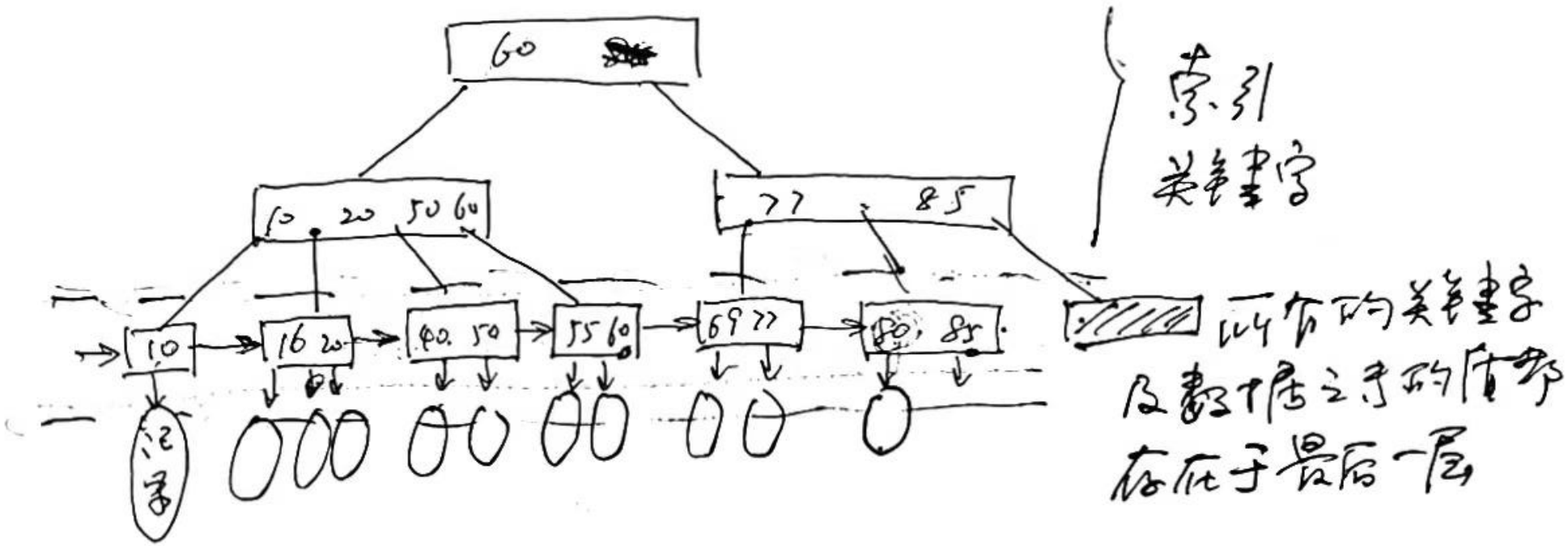


删除 4


10
3 6
13 18
1 2
5
7 9
11 12
14 17
19 20


10
6
13 18
1 2 3 5
7 9
11 12
14 17
19 20


6 10 13 18
1 2 3 5
7 9
11 12
14 17
19 20


60
10 20 50 60
77 85
索引
关键字
10
16 20
40 50
55 60
69 77
80 85
记录
所有的关键字
及数据记录的信息
存在于最后一层