情報構造 第十四回

辞書の実現

B木

今日の予定

- 木による辞書の実現
  - 2分探索木
  - AVL木
  - B-木

基本的な探索木
形がよくないと効率が悪い

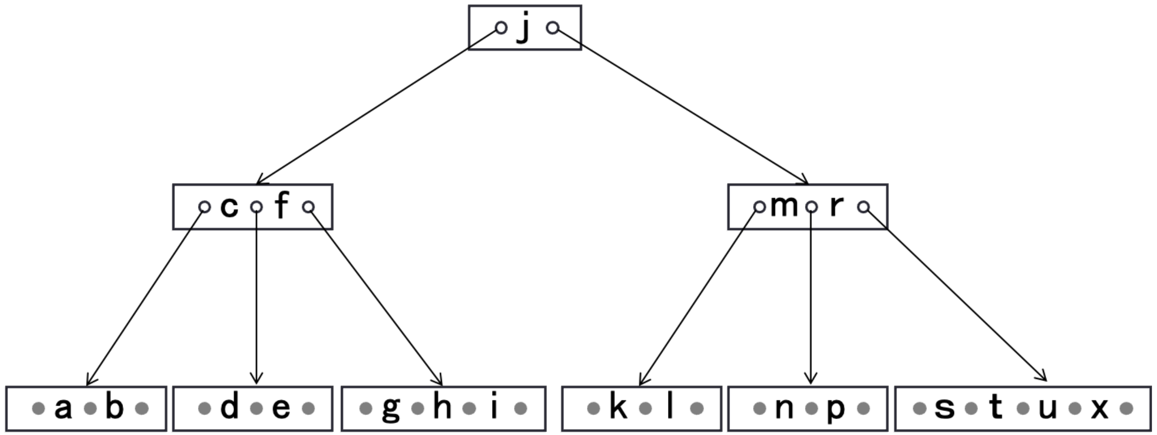
形をよくする方法を導入

多分木に一般化

- 今日はB木を行います

多分探索木（Multiway Search Tree）

- 2分探索木の一般化
- 各節点の子の個数が最大m（≧2）の木
- 節点中のキー（辞書要素）の個数は，子の個数より1小さい
- 節点n_1とn_2が同じ節点の子でn_1がn_2より左側に位置するとき，n_1の部分木中のキーの値は，n_2の部分木中のキーの値より小さい
- 従って，通りがけ順で多分探索木をたどると，キーのソート列が得られる



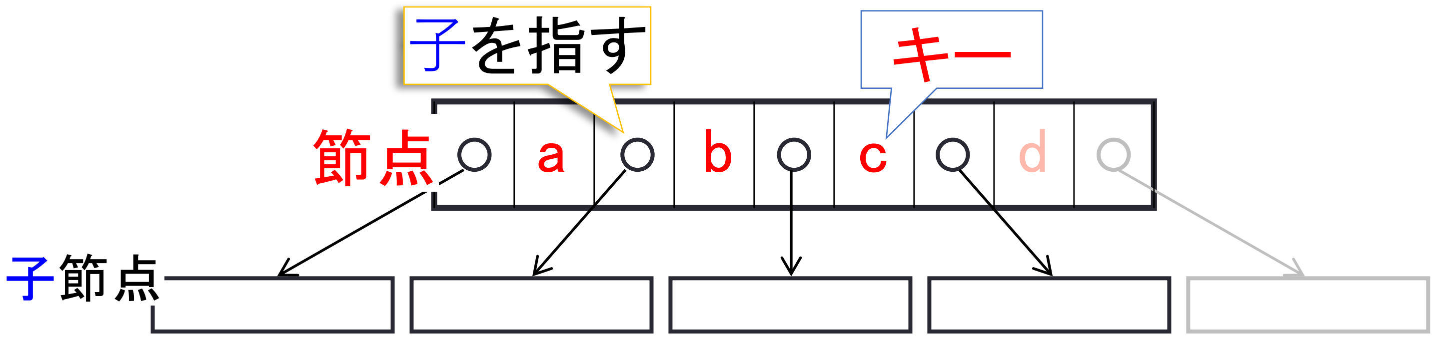
B木（R. Bayer & E. McCreight 1972）

- バランスした多分探索木，子の最大数mを次数（order）とよぶ
- 外部記憶向きのデータ構造で，節点はブロックで実現
- 外部記憶装置は，アクセス速度が遅いため，検索・更新でアクセス頻度の少ないデータ構造がよい

⇒B木は，葉にm-1のデータ量（mは通常100以上）を格納，木の高さが低いため，効率よく検索・更新が行える

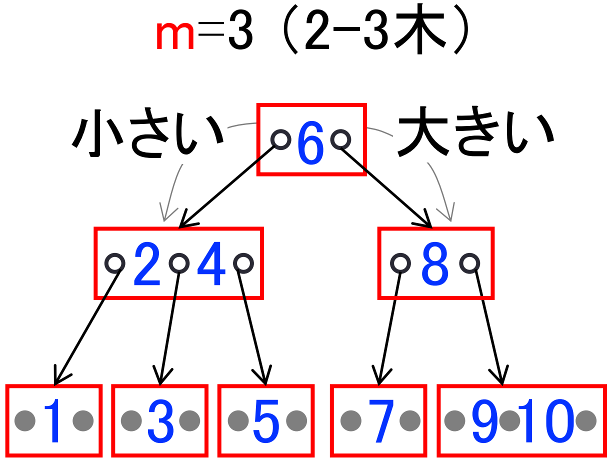
B木の節点

- 節点:「キー」と「子を指す」フィールドの構造体
  - 辞書要素をキーとよぶ
- 次数mのB木の節点は
  - キーフィールドがm-1個まで
  - 子を指すフィールドがm個まで
- キーの個数 k
  - 根以外の節点中：$\left\lceil \frac{m}{2} \right\rceil - 1$～$m-1$個,
  - 根節点中：1から$m-1$個
- キーは線形順序をもち，キーでB木を探索し，格納接点を見つけてアクセスする



次数mのB木：(m-1)-m木

- 節点の子の個数
  - 根のとき：2～m または，0（葉）
  - 根と葉以外のとき：$\left\lceil\frac{m}{2}\right\rceil$～$m$個
- 節点のキーの個数
  - 葉以外のとき：子の個数-1
  - 葉かつ根のとき：1～m-1
  - 葉かつ根ではないとき：$\left\lceil\frac{m}{2}\right\rceil - 1$～m-1
- すべての葉は，同じ深さにある（バランスしている）
- 節点中のキーはソートされている
- 各キーは2分探索木のように，そのキーの隣の子の部分木中のキーを線形順序で分割する



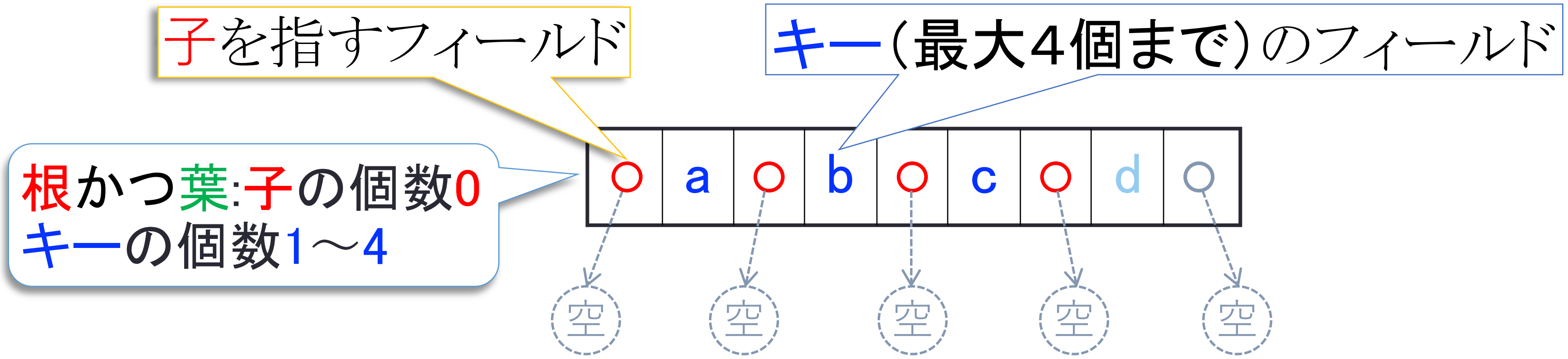
B木の具体例

- 4-5木
- 2-3木

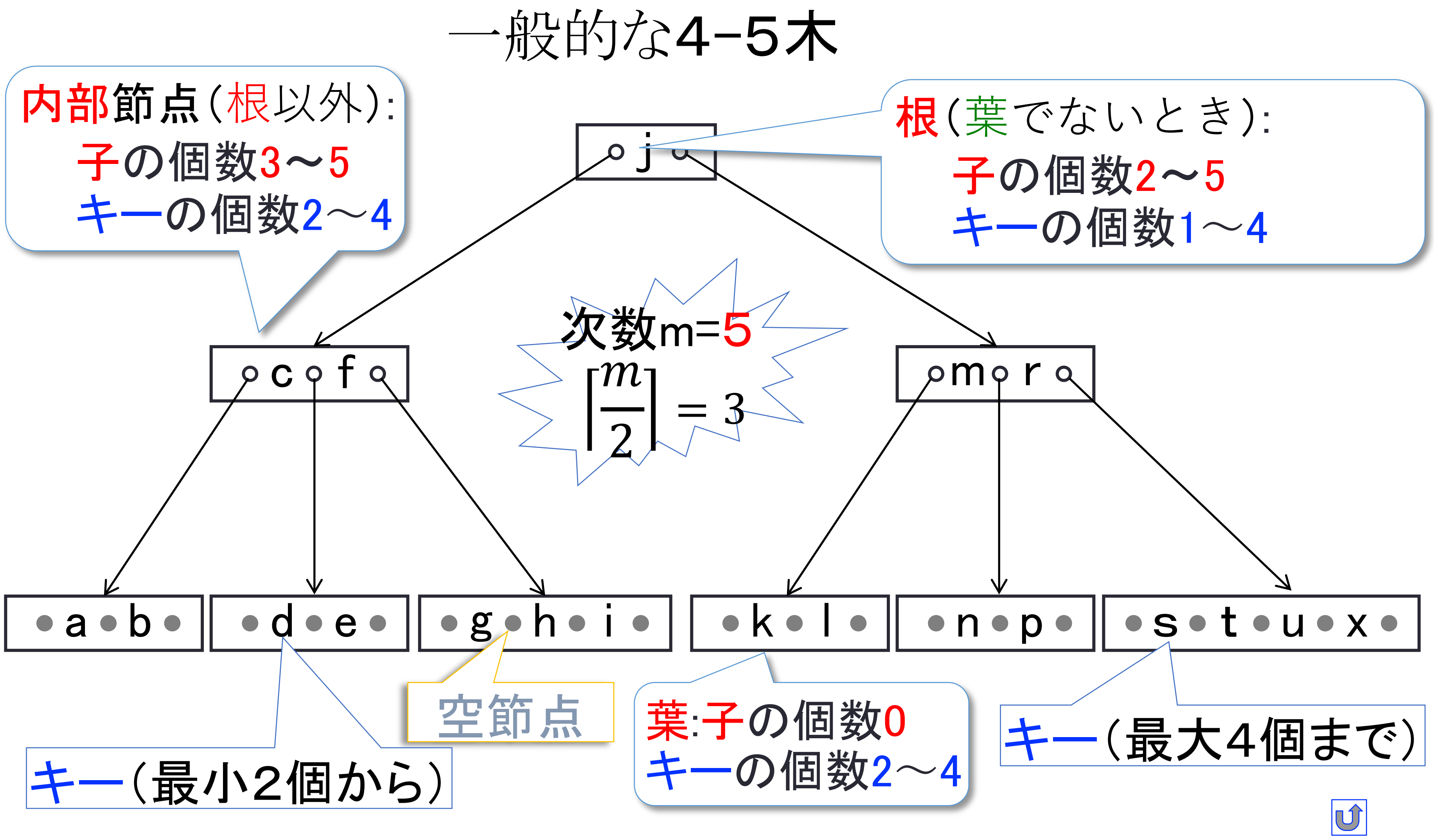
B木：4-5木（次数 m=5）

【例】4-5木（次数m = 5）

- ただひとつの節点からなる4-5木
- 根が葉である木

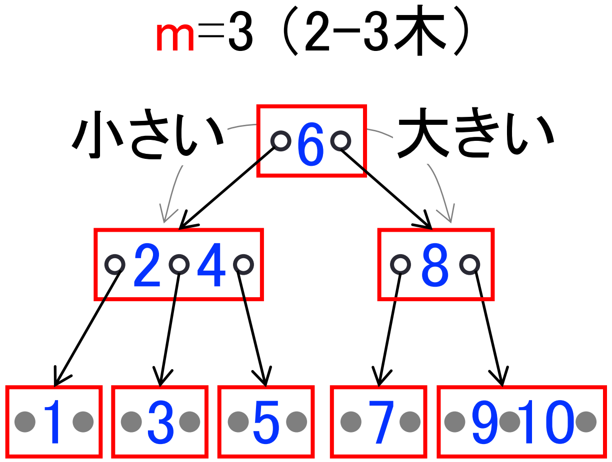


【例】4-5木（次数m = 5）


一般的な4-5木
内部節点(根以外):
子の個数3～5
キーの個数2～4
根(葉でないとき):
子の個数2～5
キーの個数1～4
j
c f
m r
次数m=5
$\left\lceil \frac{m}{2} \right\rceil = 3$
a b
d e
g h i
k l
n p
s t u x
空節点
キー(最小2個から)
葉:子の個数0
キーの個数2～4
キー(最大4個まで)

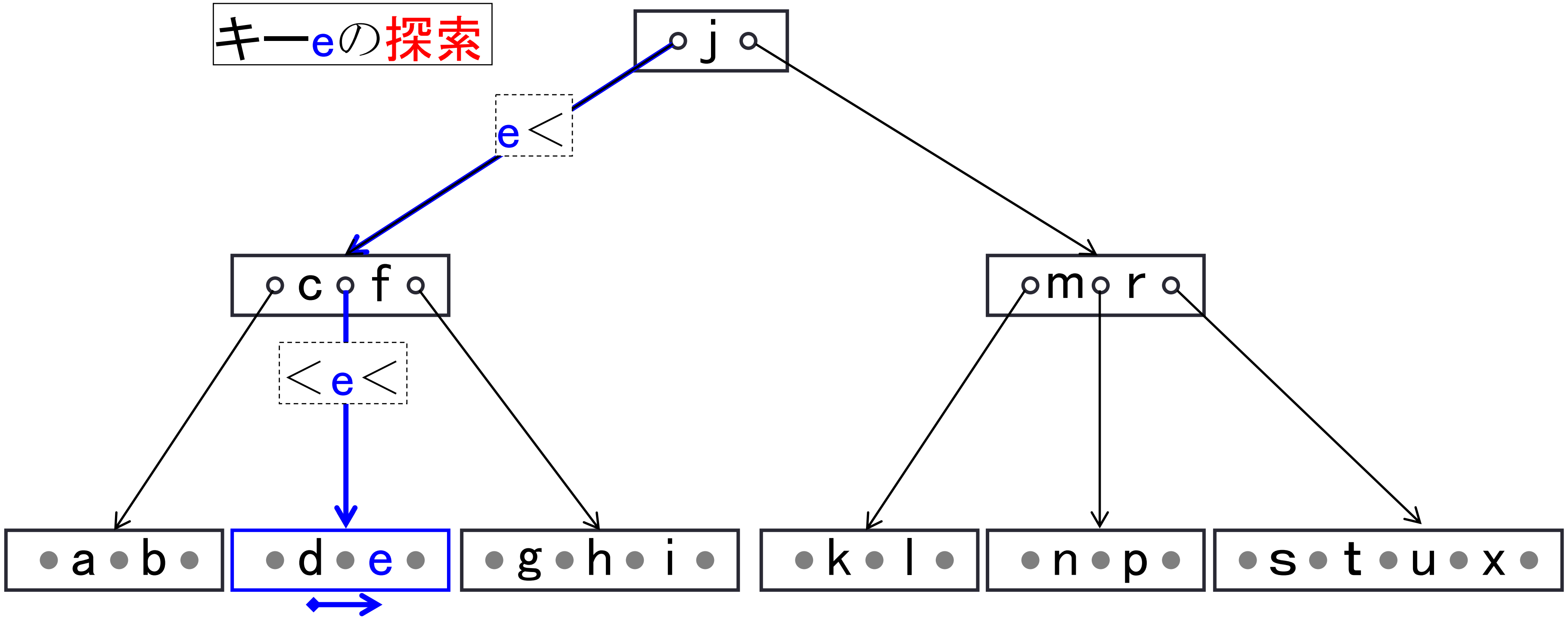
【再掲】次数mのB木：(m-1)-m木

- 節点の子の個数
  - 根のとき：2～m または，0（葉）
  - 根と葉以外のとき：$\left\lceil \frac{m}{2} \right\rceil$～$m$個
- 節点のキーの個数
  - 葉以外のとき：子の個数-1
  - 葉かつ根のとき：$1 \sim m-1$
  - 葉かつ根ではないとき：$\left\lceil \frac{m}{2} \right\rceil - 1 \sim m-1$
- すべての葉は，同じ深さにある（バランスしている）
- 節点中のキーはソートされている
- 各キーは2分探索木のように，そのキーの隣の子の部分木中のキーを線形順序で分割する



【例】4-5木のキー探索

- キー探索は，2分探索木と同じ
  - 最悪，木の高さのステップを要する



【例】4-5木のキー挿入

- 以下の順にキーを挿入
  - g a f b k d h m j e s i r x c l n t u p

1. g a f bを挿入

● a ● b ● f ● g ●

根を生成、
キーを
辞書引き順
a b f gで挿入

2. kを挿入

● a ● b ● f ● g ● k ●

キーは4 個まで
⇒節点を分割

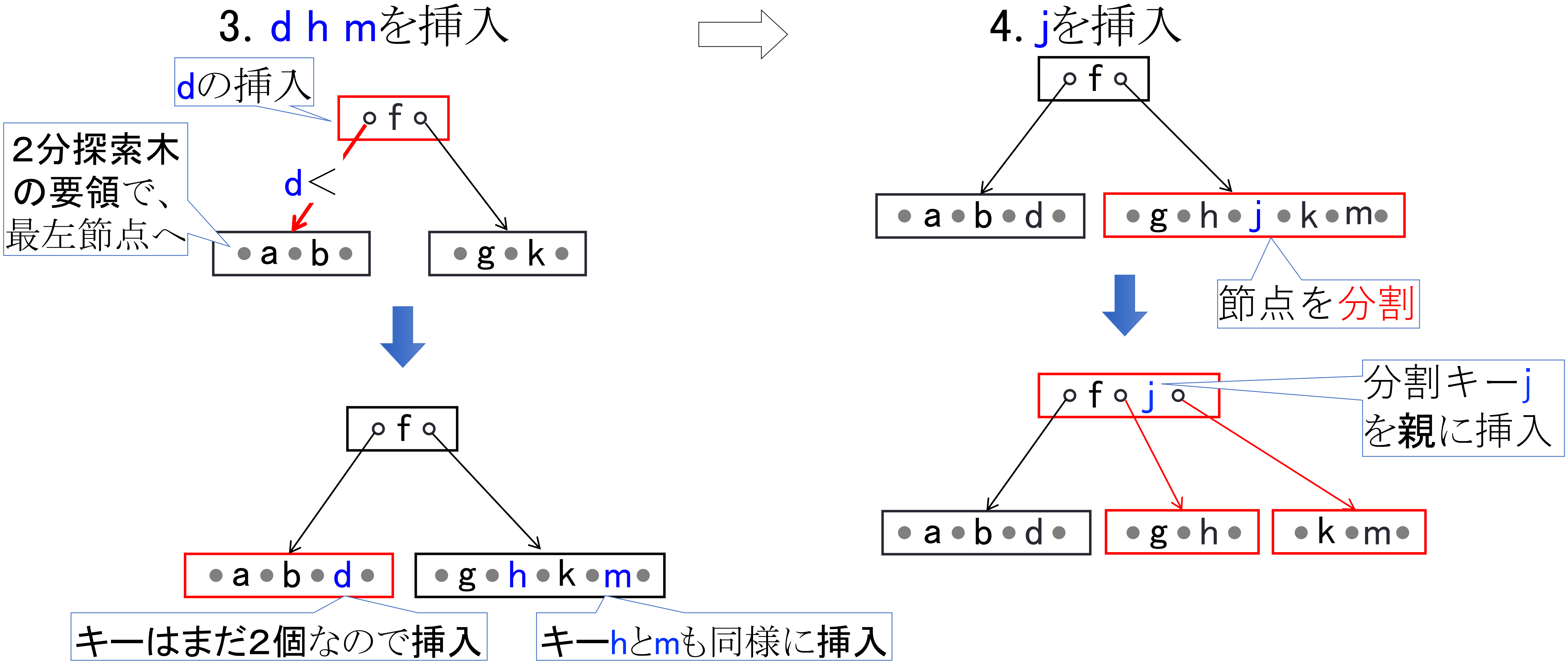
○ f ○

● a ● b ●　　● g ● k ●

親を生成し
分割キーfを
挿入

【例】4-5木のキー挿入（つづき）

- g a f b k d h m j e s i r x c l n t u p

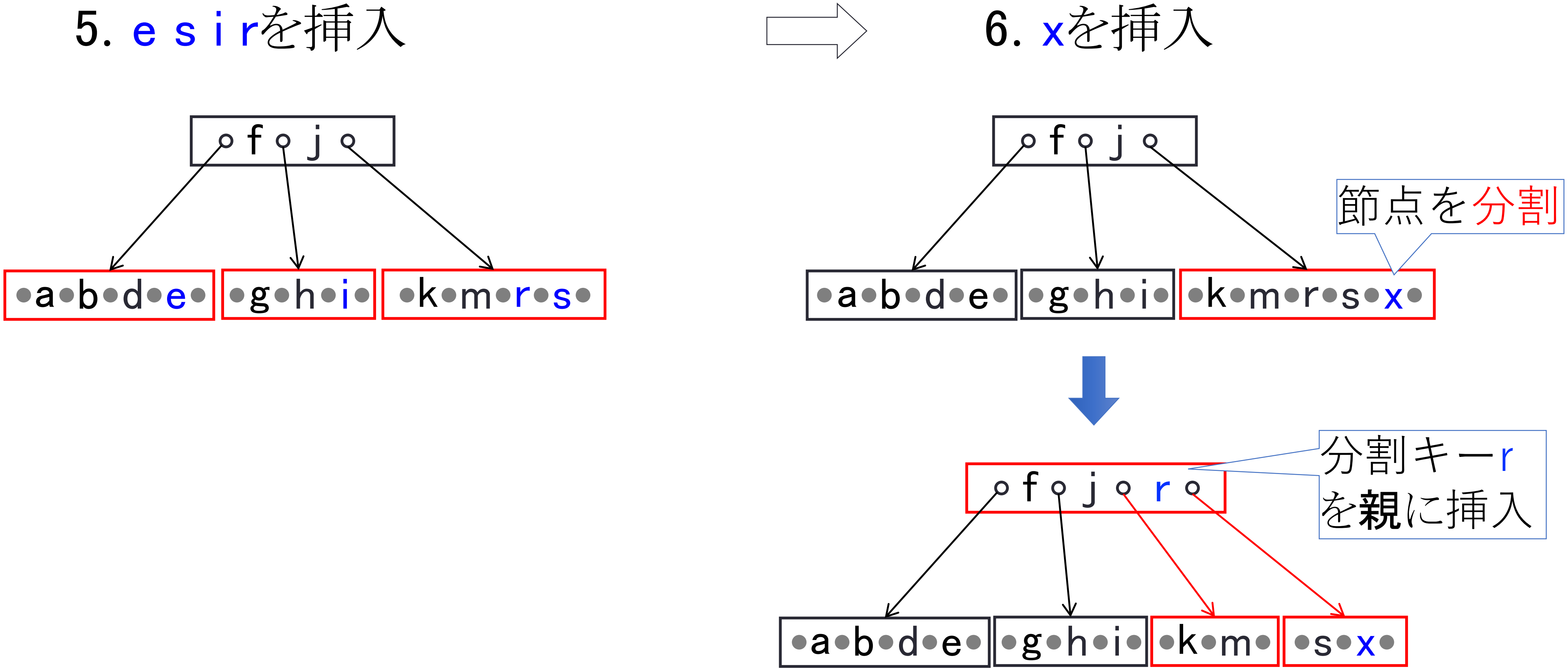


【例】4-5木のキー挿入（つづき）

- g a f b k d h m j e s i r x c l n t u p

5. e s i rを挿入

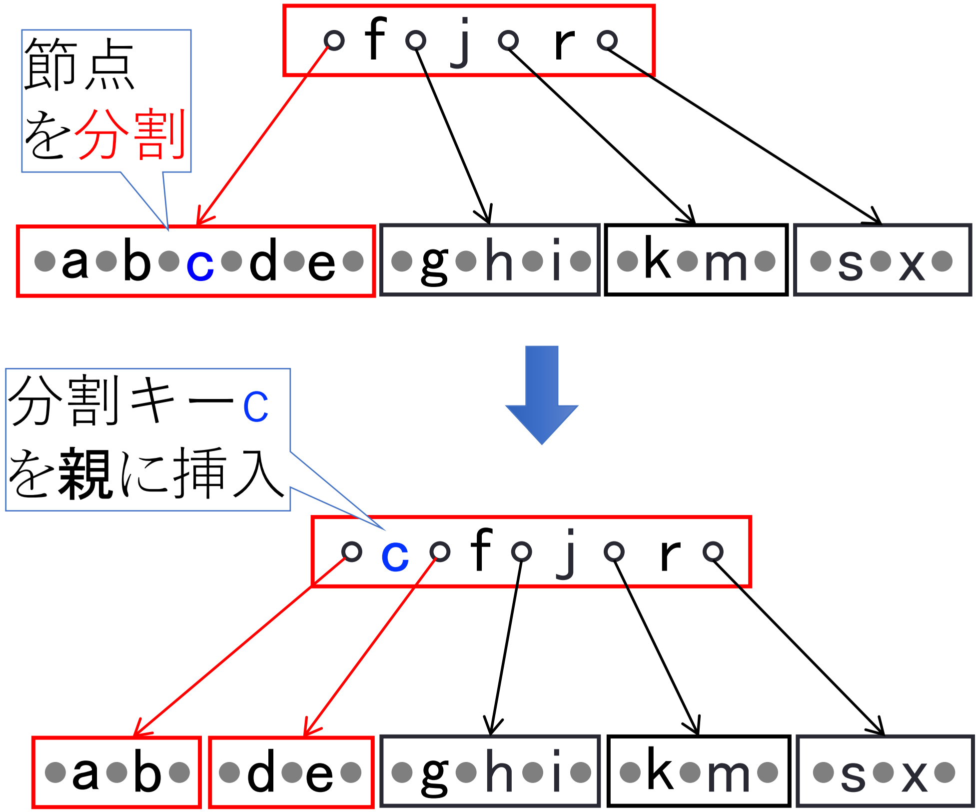
6. xを挿入



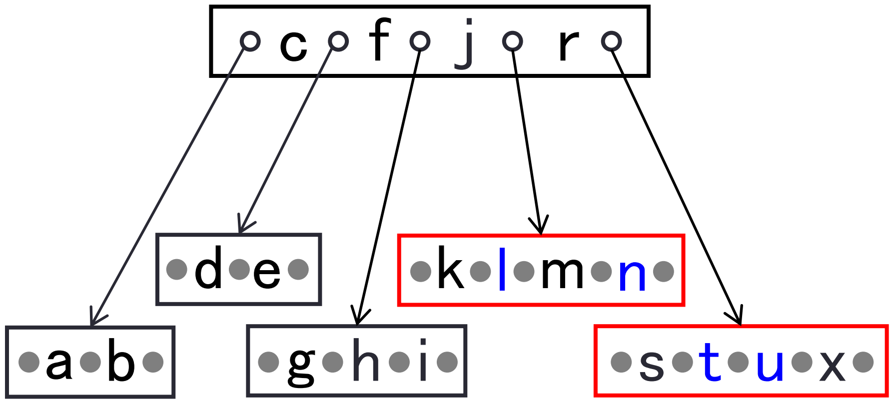
【例】4-5木のキー挿入（つづき）

- g a f b k d h m j e s i r x c l n t u p

7. cを挿入



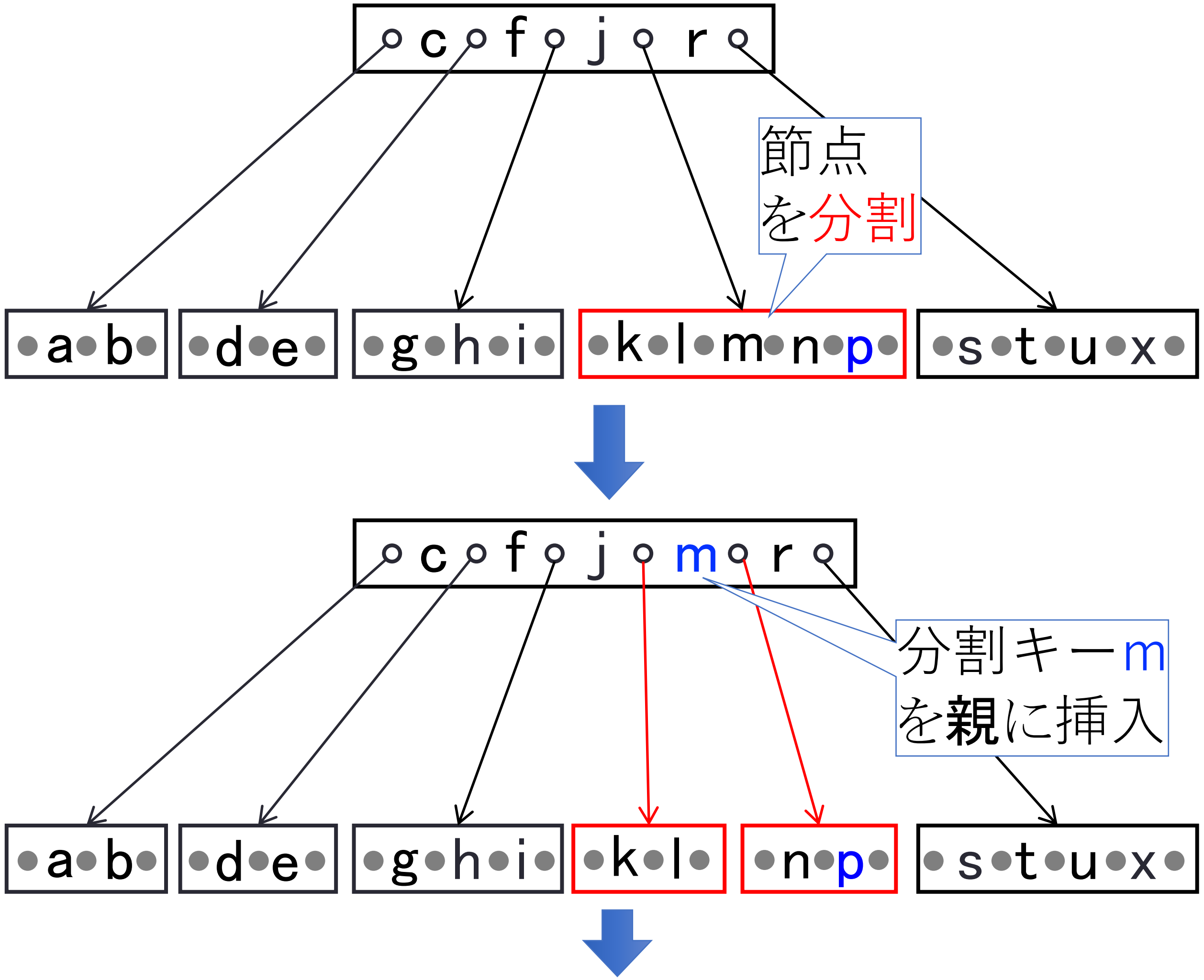
8. l n t uを挿入

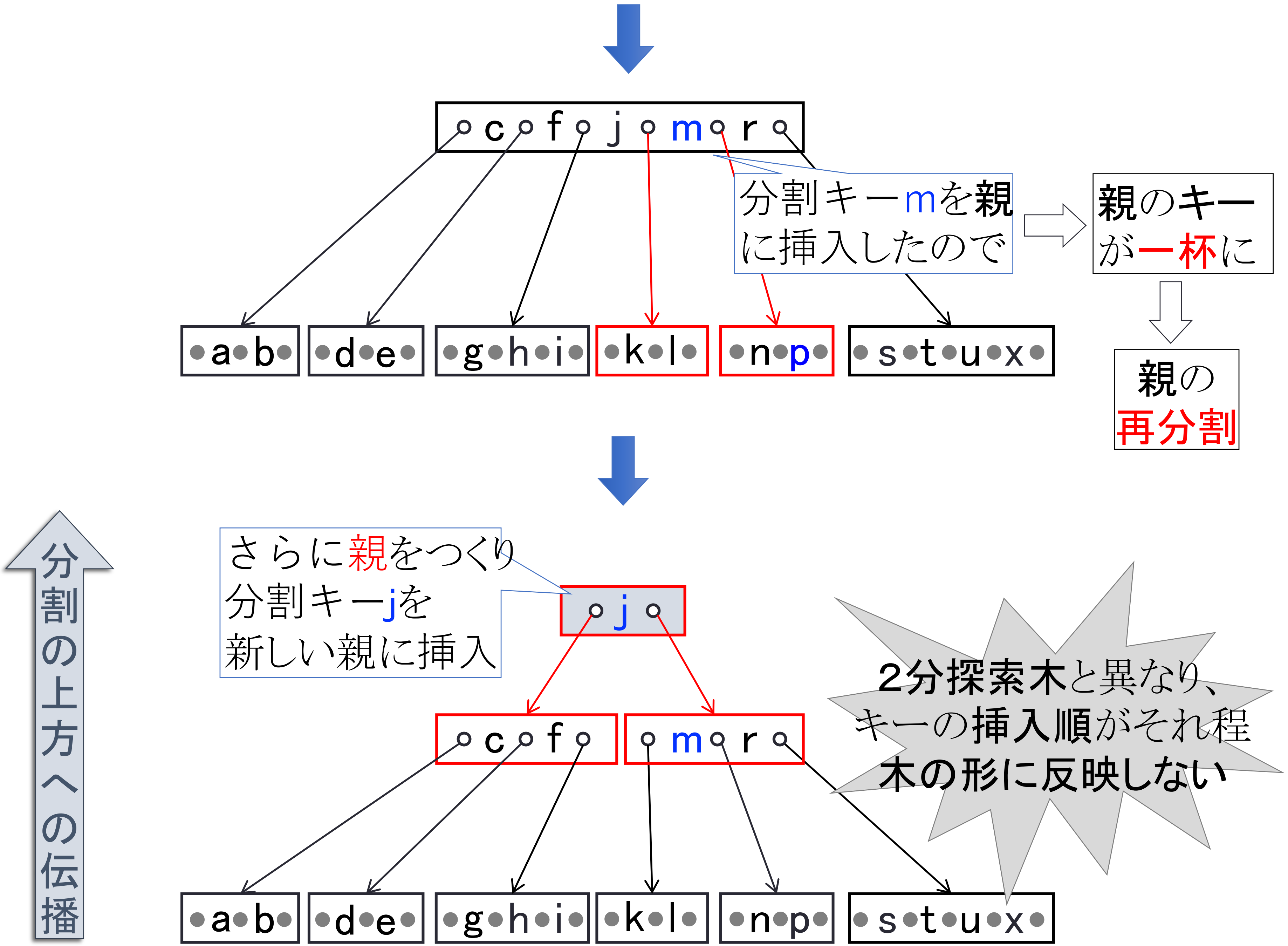


【例】4-5木のキー挿入（つづき）

- g a f b k d h m j e s i r x c l n t u p

9. pを挿入





c f j m r
a b
d e
g h i
k l
n p
s t u x
分割キーmを親に挿入したので
親のキーが一杯に
親の再分割
さらに親をつくり分割キーjを新しい親に挿入
j
c f
m r
a b
d e
g h i
k l
n p
s t u x
分割の上方への伝播
2分探索木と異なり、キーの挿入順がそれ程木の形に反映しない

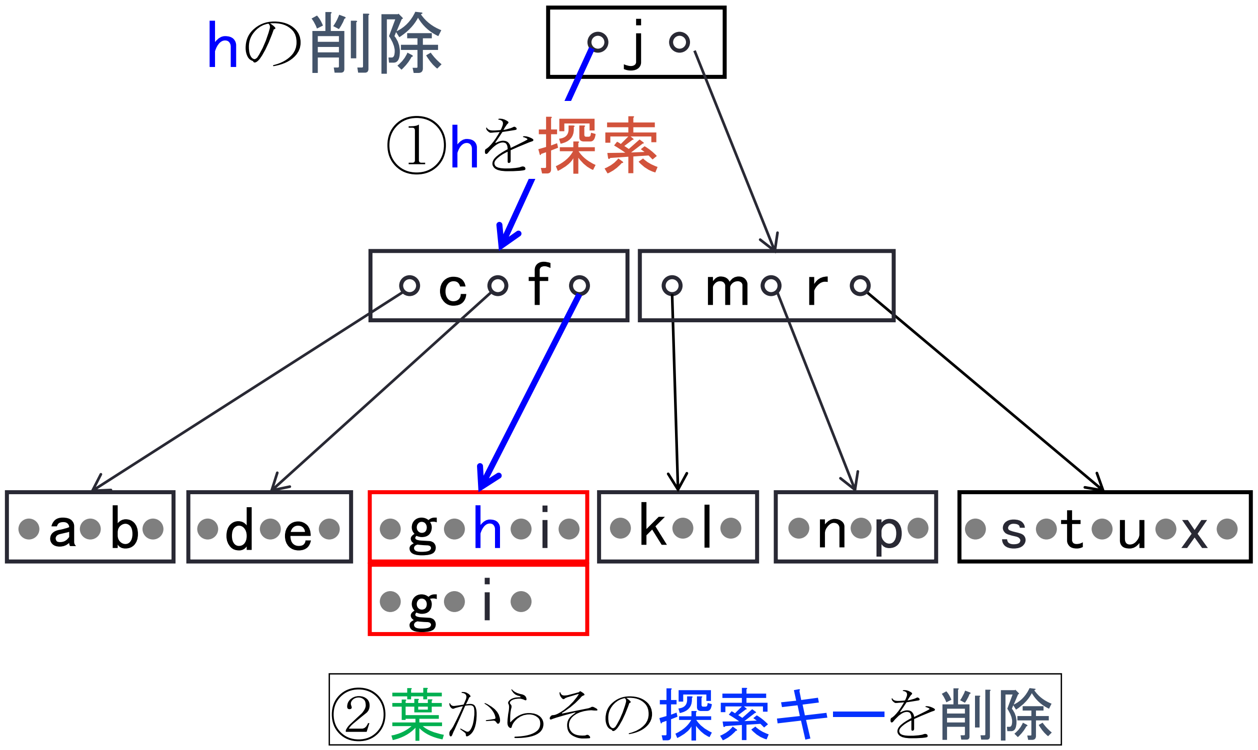
【例】4-5木のキー削除

- 削除キーを探索
  - ないときは終了
  - あるときは場合分け

- 葉節点にある
  - 葉節点のキーが3個以上
  - 葉節点のキーが2個
- 内部接点にある
- 根節点にある

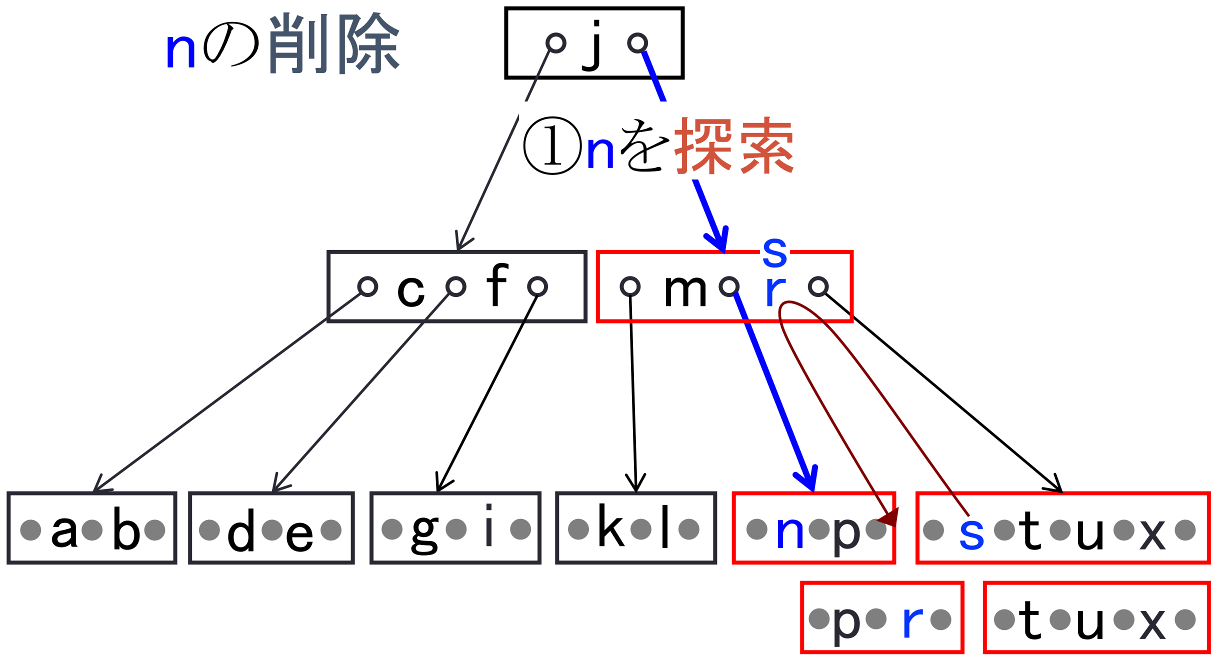
【例】4-5木のキー削除（葉節点から）

- 葉のキーが3個以上のとき



【例】4-5木のキー削除（葉節点から）

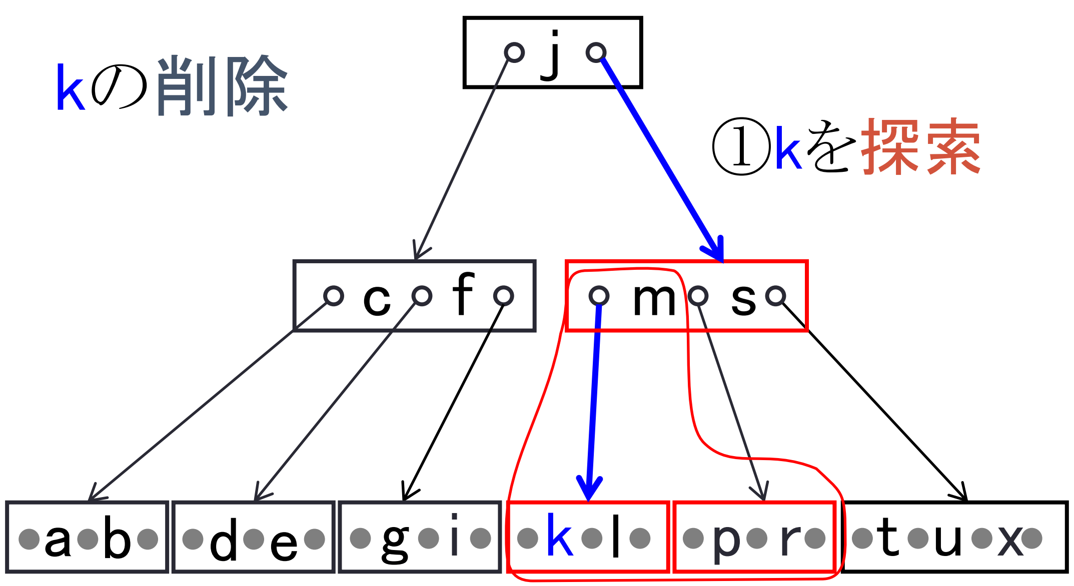
- 葉のキーが2個のとき
  - 隣の節点のキーが3個以上の時

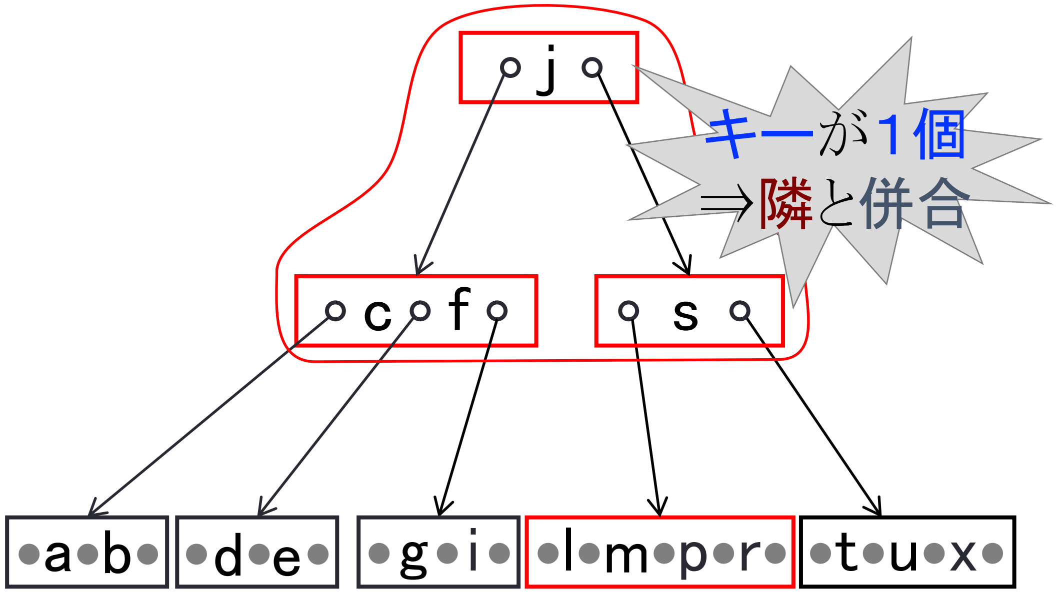


②探索キーの削除し、隣の葉のキーが3個以上なら、隣からキーを借りてくる。
このとき、分割キーも考慮。

【例】4-5木のキー削除（葉節点から）

- 葉のキーが2個のとき
  - 隣の節点のキーが2個のとき

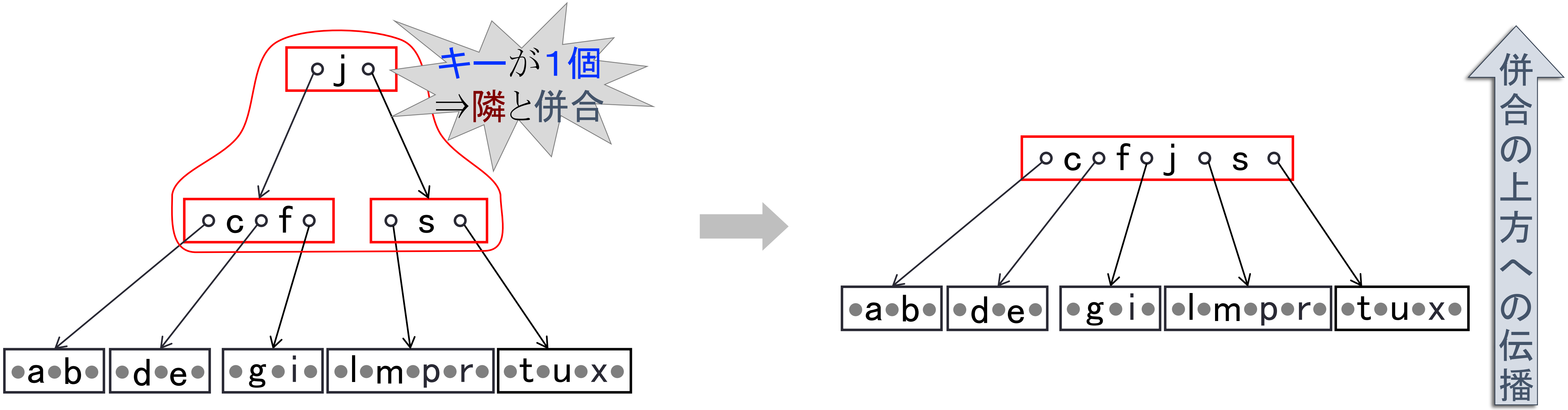




②キーを削除し、隣の葉と併合する。
このとき、分割キーも組込む。

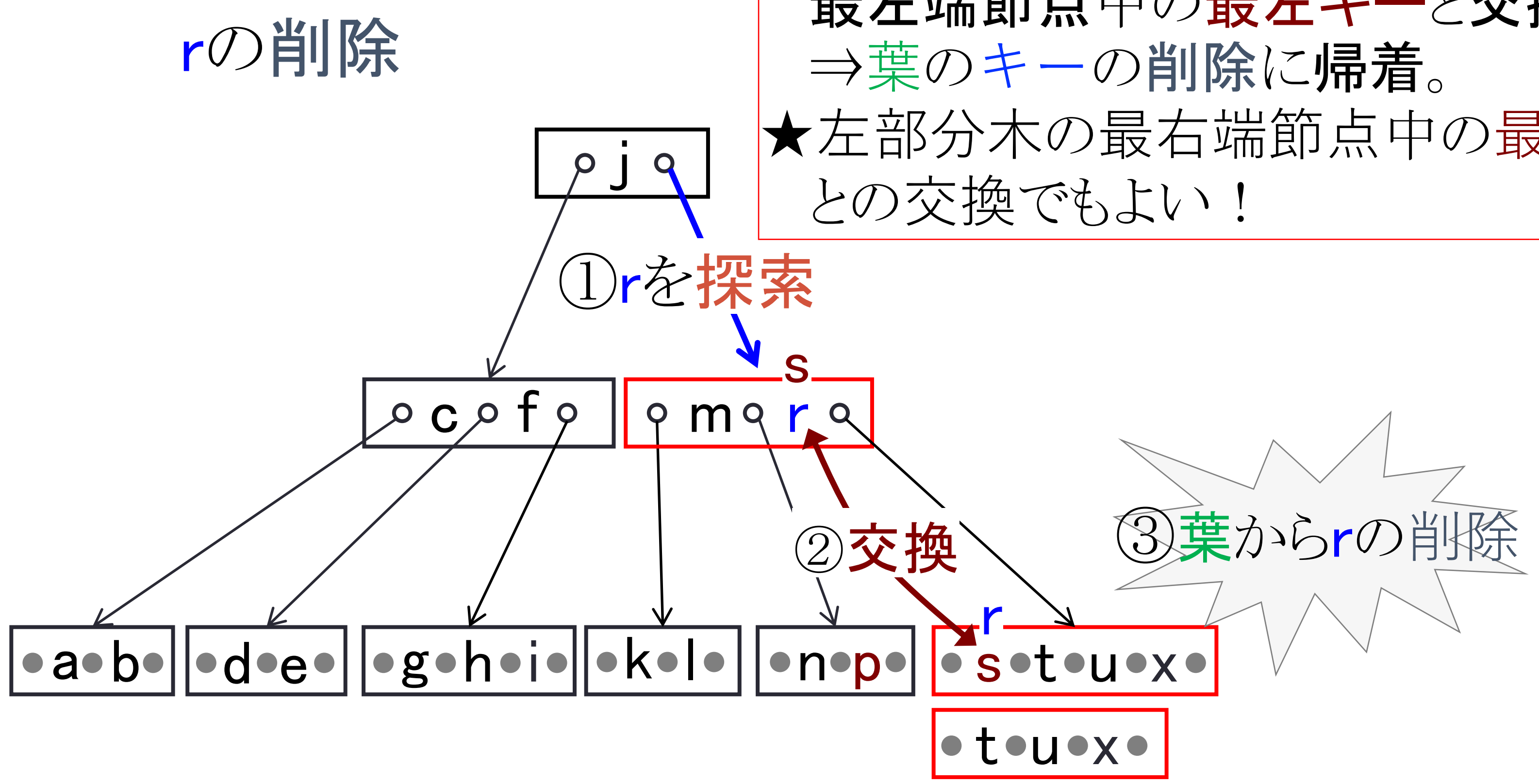
③もし節点のキーが1個に減るとき、
隣のキーが全て2個なら、隣と節点を併合する。
分割キーも組込むため、親節点が無くなる

kの削除


j
c f
s
a b
d e
g i
l m p r
t u x
キーが1個
⇒隣と併合
c f j s
a b
d e
g i
l m p r
t u x
併合の上方への伝播

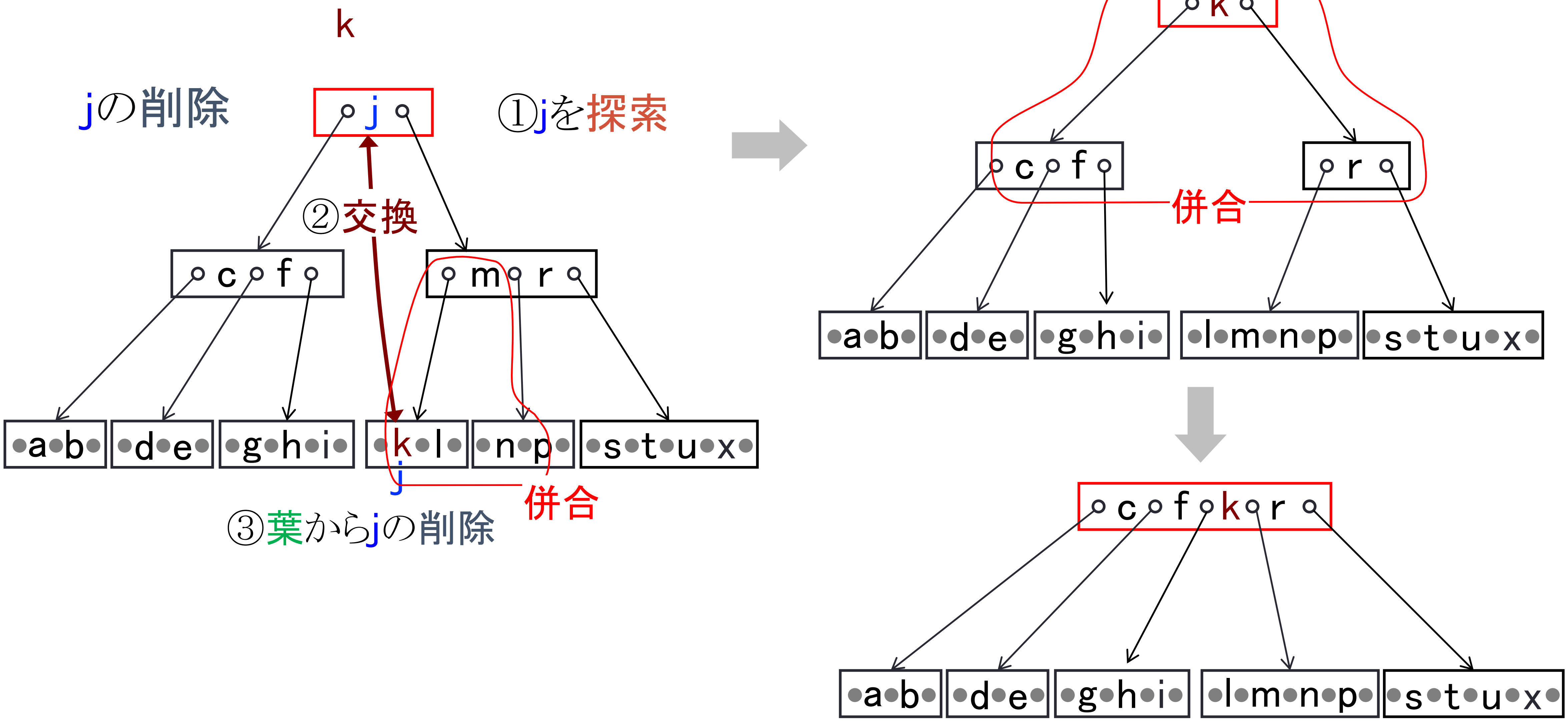
【例】4-5木のキー削除（内部節点から）

②内部節点中の削除キーの**右部分木**の**最左端節点**中の**最左キー**と**交換**する。
⇒葉のキーの削除に帰着。
★左部分木の最右端節点中の最右キーとの交換でもよい！



【例】4-5木のキー削除（根節点から）

- 内部接点のときと同じ



B木：2-3木（次数 m=3）

【例】2-3木のキー挿入

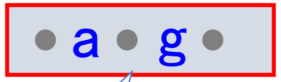
次数 $\mathrm{m} = 3, \left\lceil \frac{m}{2} \right\rceil = 2$

節点の子の数2～3個

キーの数1～2個

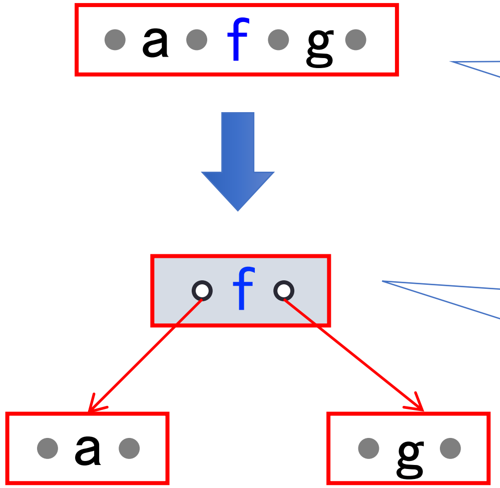
- 以下の順にキーを挿入
  - g a f b k d h m j e s i r x c l n t u p

1. g a を挿入



根を生成、
キーを
辞書引き順
a gで挿入

2. f を挿入



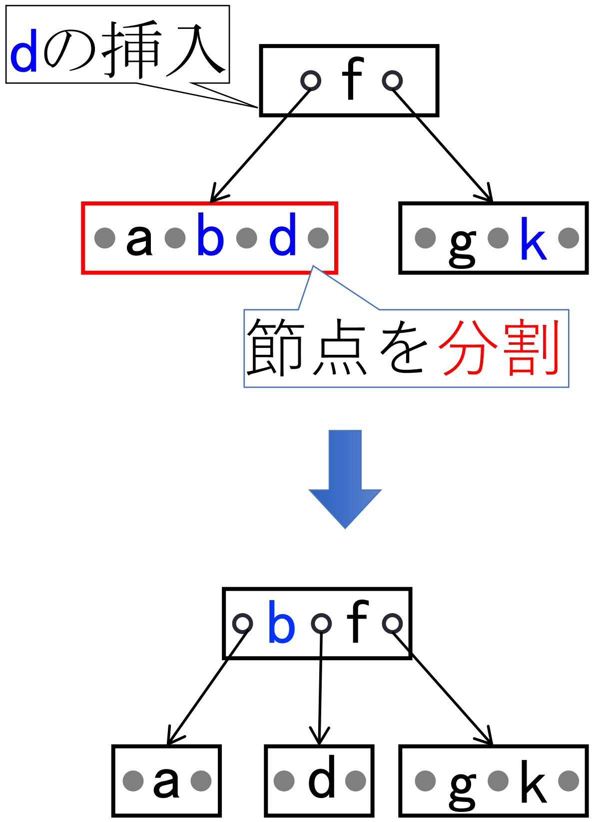
キーは２個迄
⇒節点を分割

親を生成し
分割キーfを
挿入

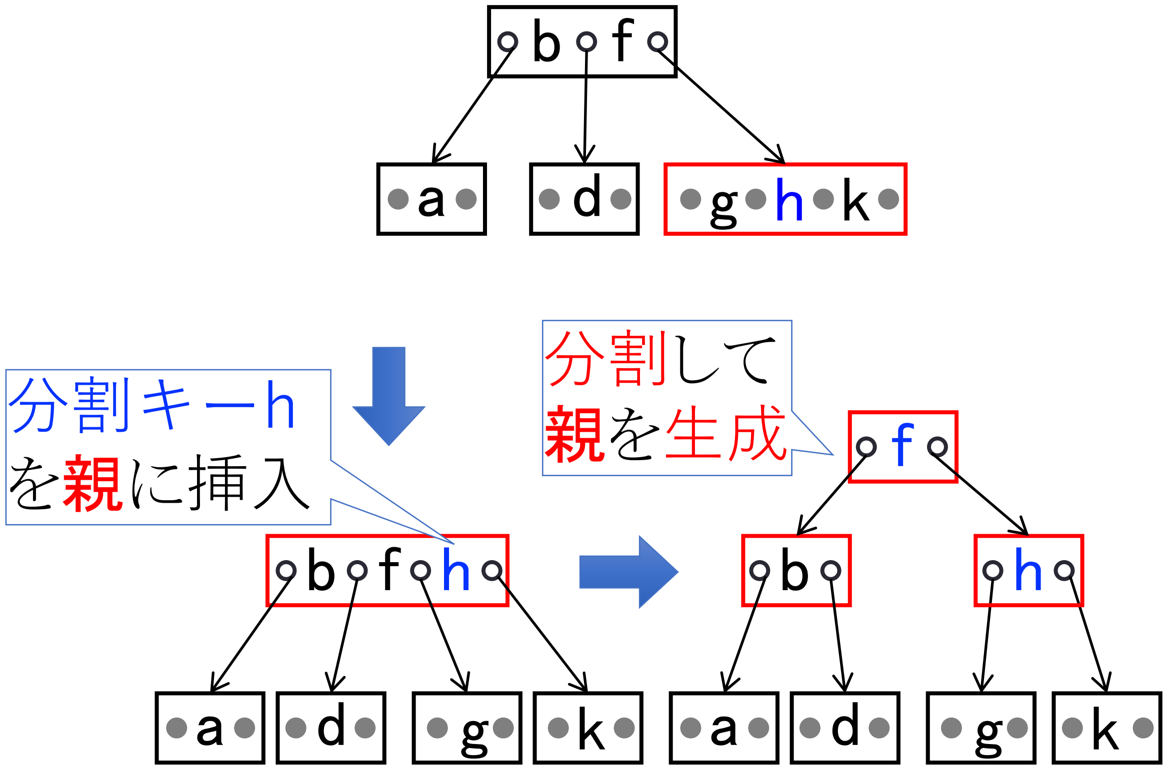
【例】2-3木のキー挿入（つづき）

- g a f b k d h m j e s i r x c l n t u p

3. b k d を挿入



4. h を挿入

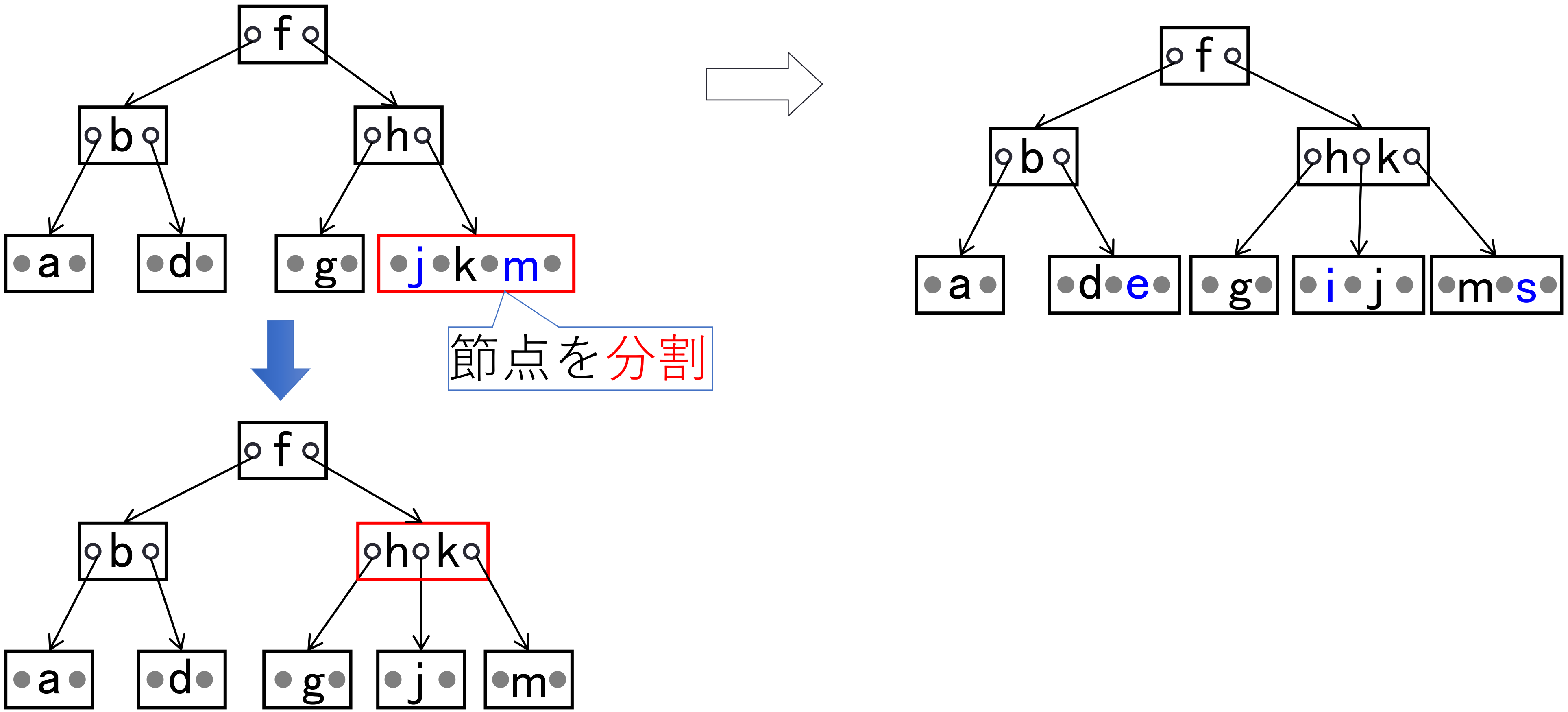


【例】2-3木のキー挿入（つづき）

- g a f b k d h m j e s i r x c l n t u p

5. m j を挿入

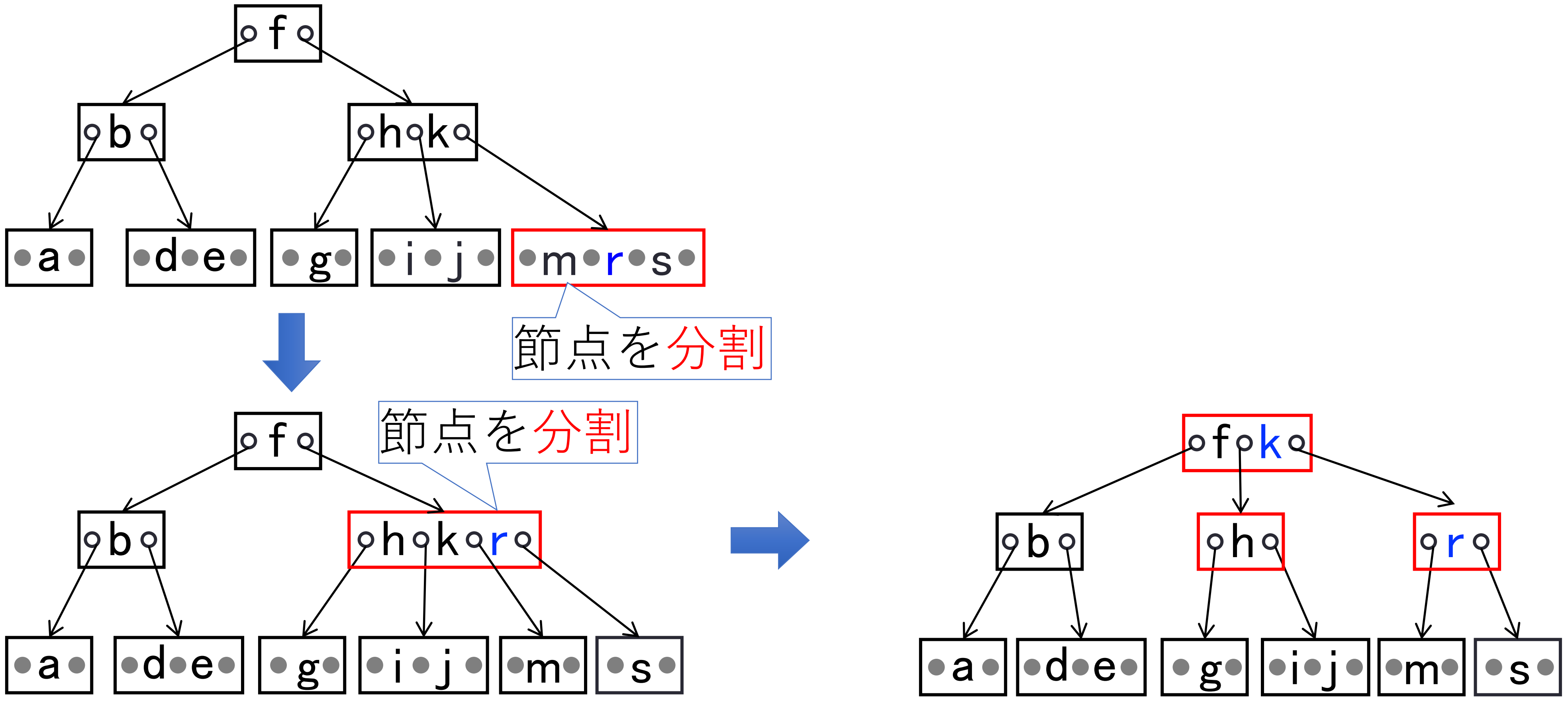
6. e s i を挿入



【例】2-3木のキー挿入（つづき）

- g a f b k d h m j e s i r x c l n t u p

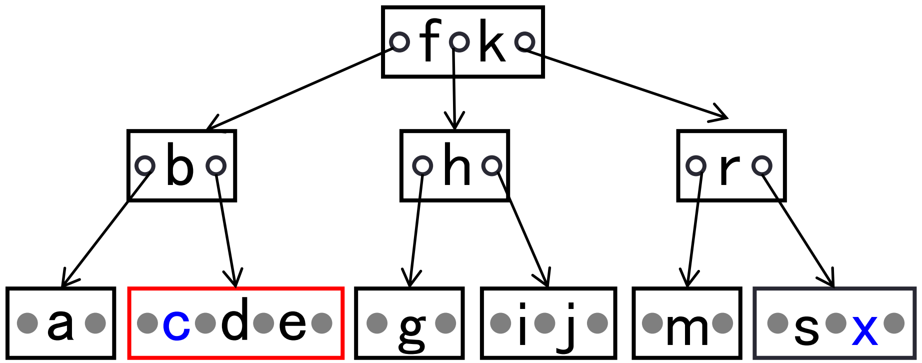
7. r を挿入

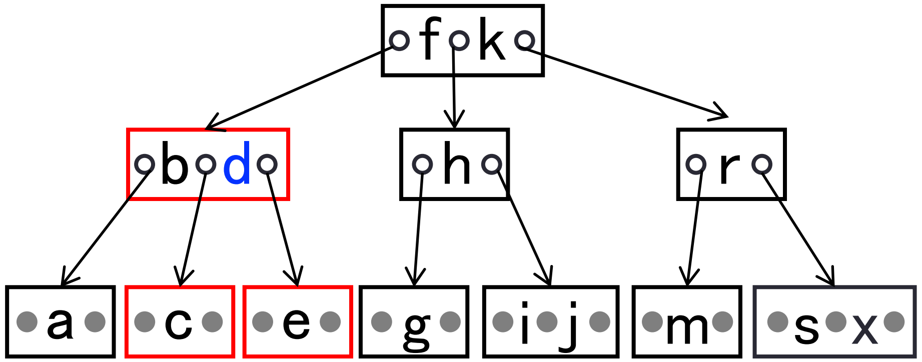


【例】2-3木のキー挿入（つづき）

- g a f b k d h m j e s i r x c l n t u p

8. x c を挿入

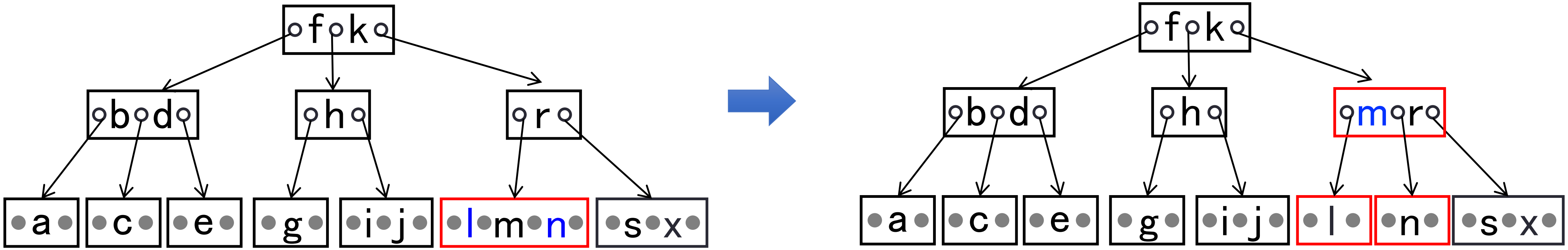




【例】2-3木のキー挿入（つづき）

- g a f b k d h m j e s i r x c l n t u p

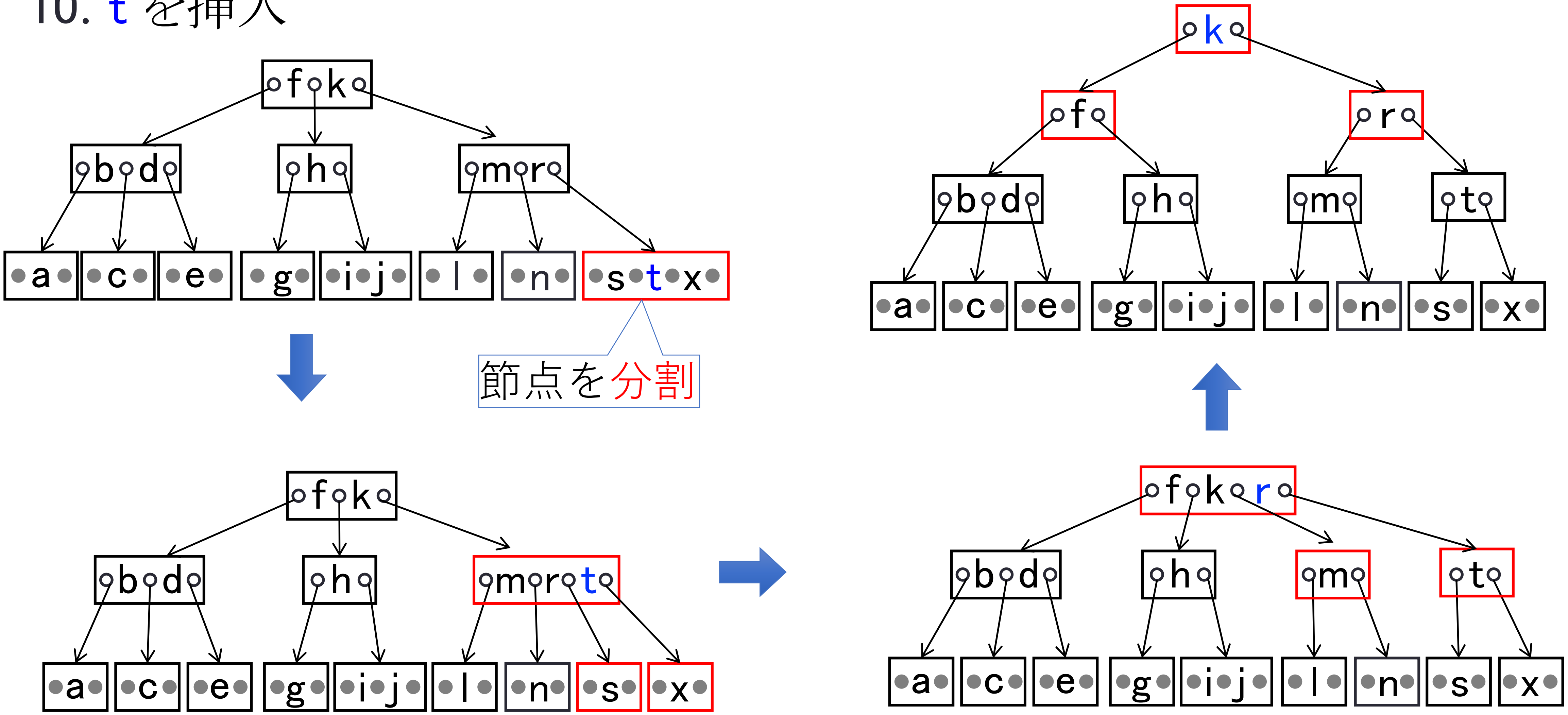
9. l n を挿入



【例】2-3木のキー挿入（つづき）

- g a f b k d h m j e s i r x c l n t u p

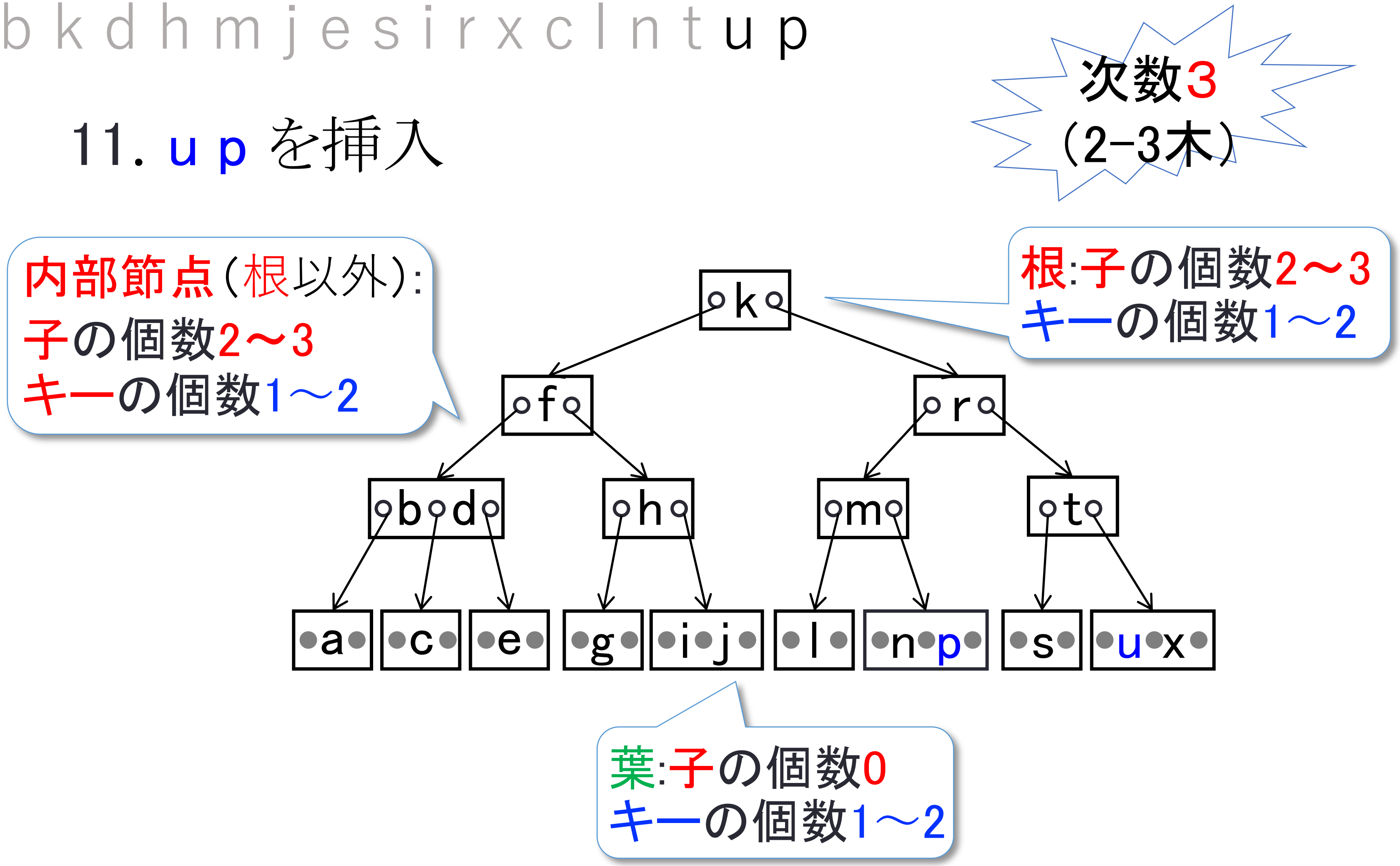
10. t を挿入



【例】2-3木のキー挿入（つづき）

- g a f b k d h m j e s i r x c l n t u p

11. u p を挿入



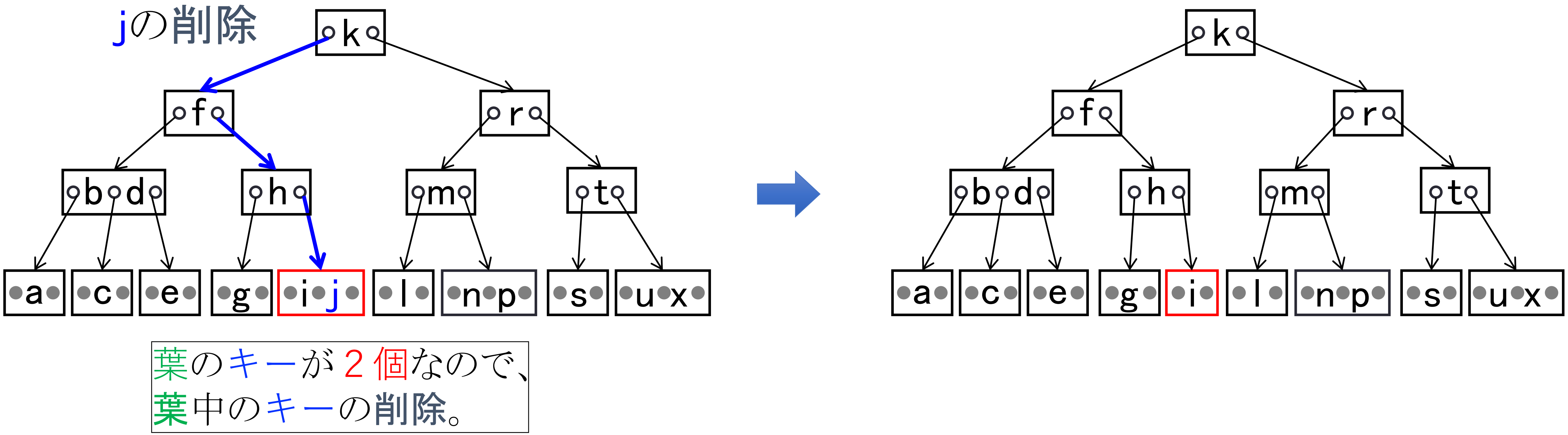
【例】2-3木のキー削除

- 削除キーの探索
  - ないときは終了
  - あるときは場合分け

- 葉節点にある
  - 葉のキーが2個
  - 葉のキーが1個
- 内部節点にある
- 根節点にある

【例】2-3木のキー削除（葉節点から）

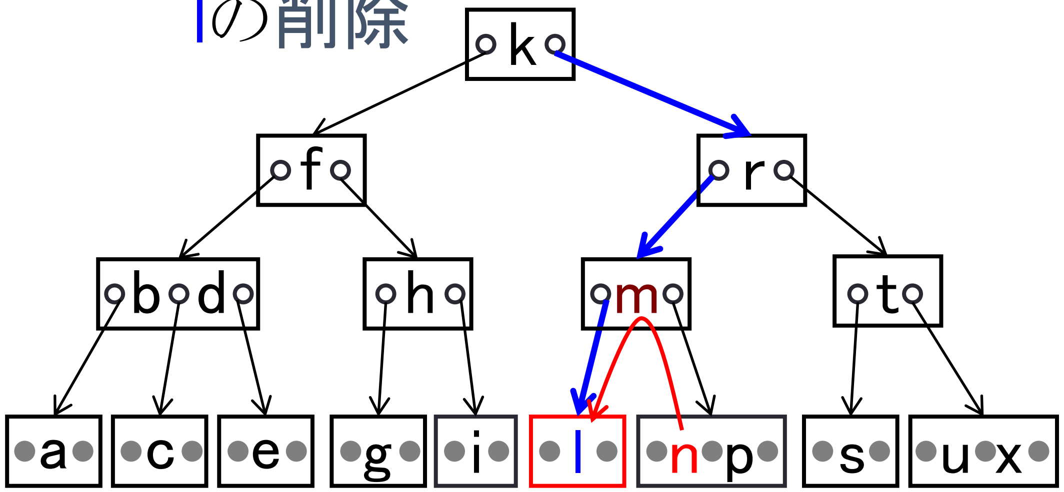
- 葉のキーが2個のとき

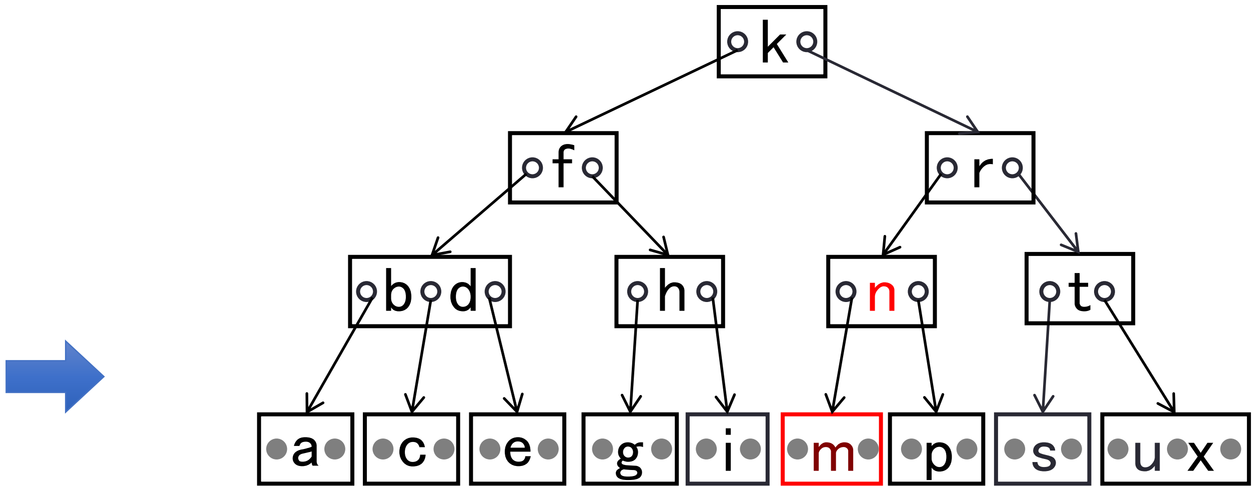


【例】2-3木のキー削除（葉節点から）

- 葉のキーが1個のとき
- 隣の葉のキーが2個のとき

lの削除



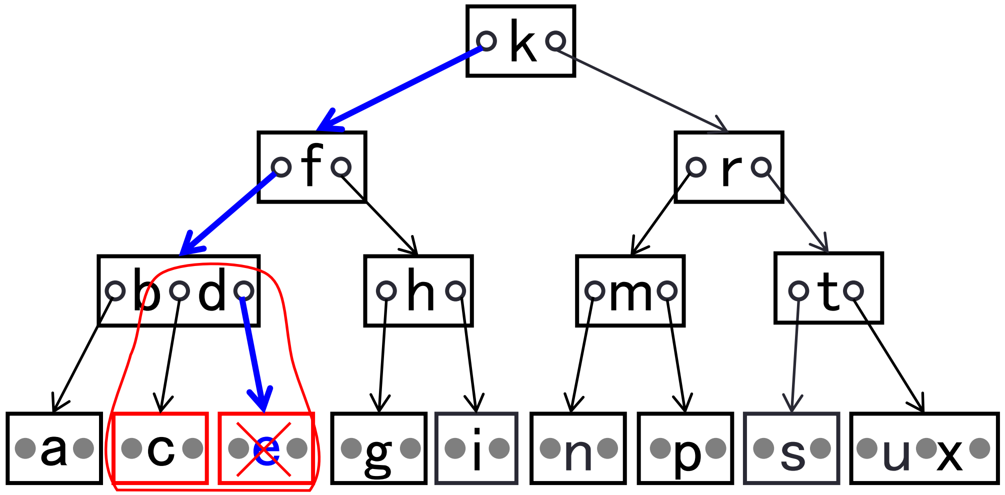


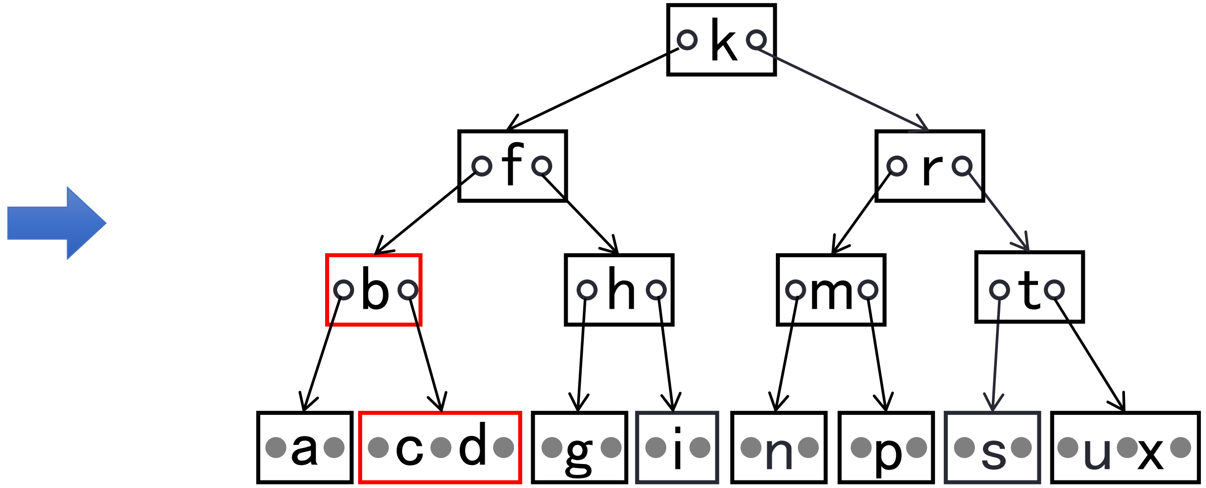
葉のキーが１個、隣のキーが２個より、
キーの削除とともに、隣からキーを借りる。
(実際には分割キー)

【例】2-3木のキー削除（葉節点から）

- 葉のキーが1個のとき
- 隣の葉のキーが1個のとき

eの削除

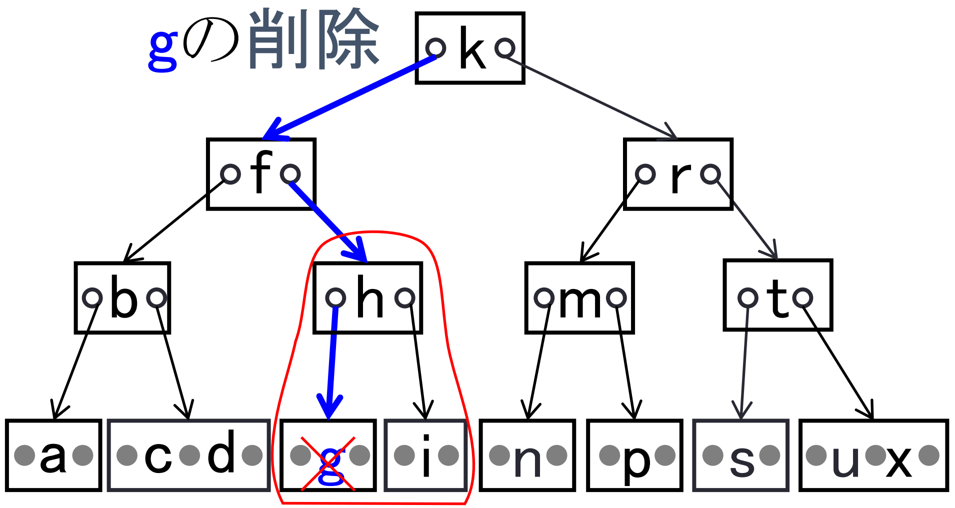


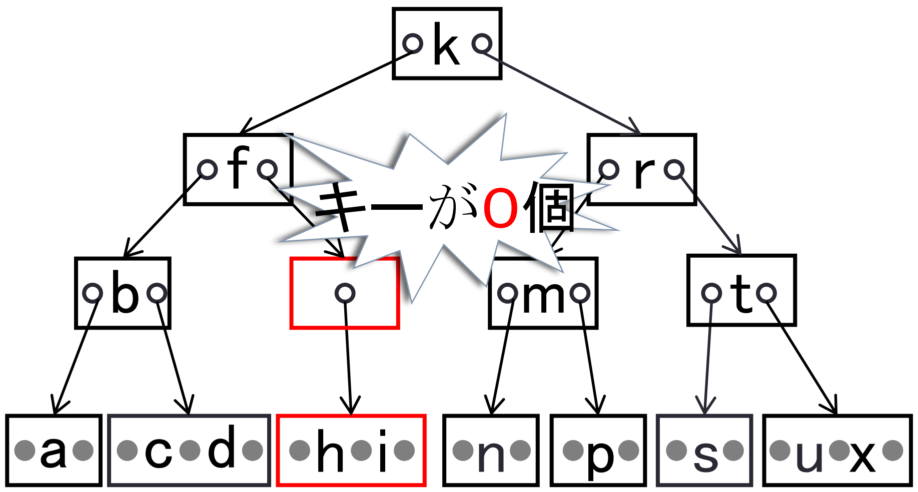


葉のキーが１個、隣のキーも１個より、
キーの削除とともに、分割キーも含め隣の葉と併合。

【例】2-3木のキー削除（葉節点から）

- 葉のキーが1個のとき
- 隣の葉のキーが1個のとき
- 隣の葉の併合=>親節点のキーがなくなるとき

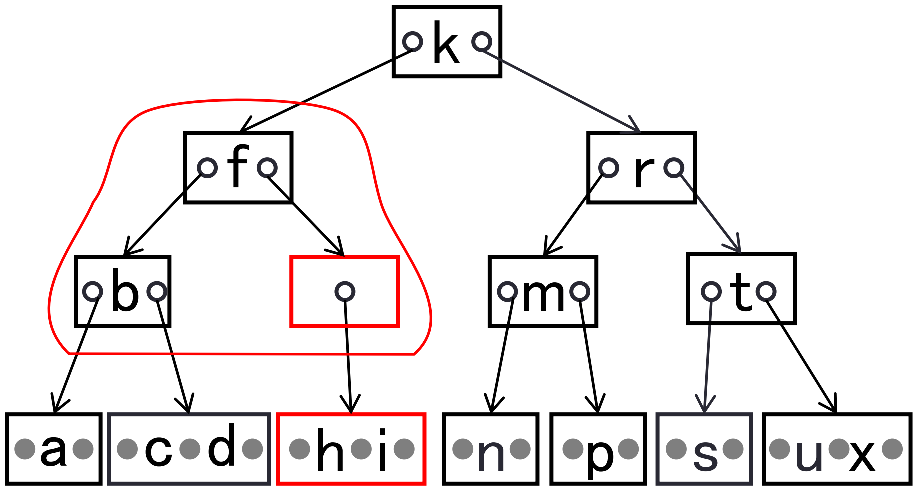


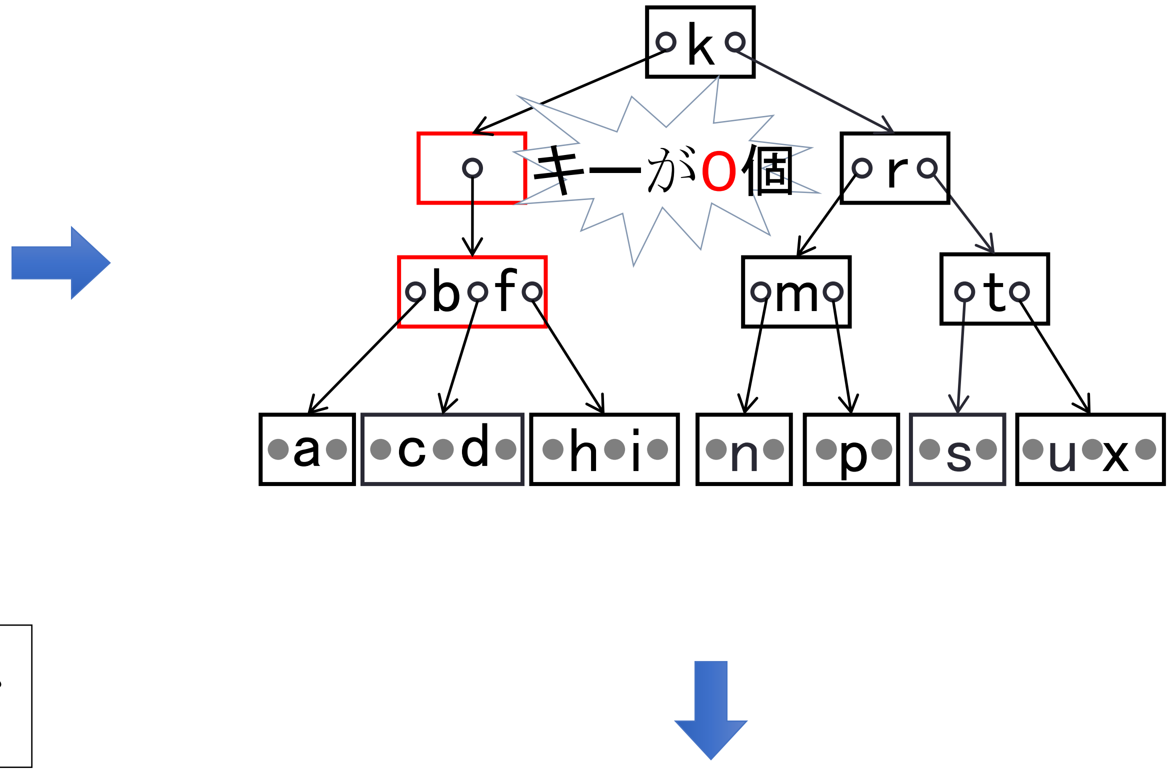


葉のキーが１個、隣のキーも１個より、
キーの削除とともに、分割キーも含め隣の葉と併合。

つづき

- 内部節どうしの併合

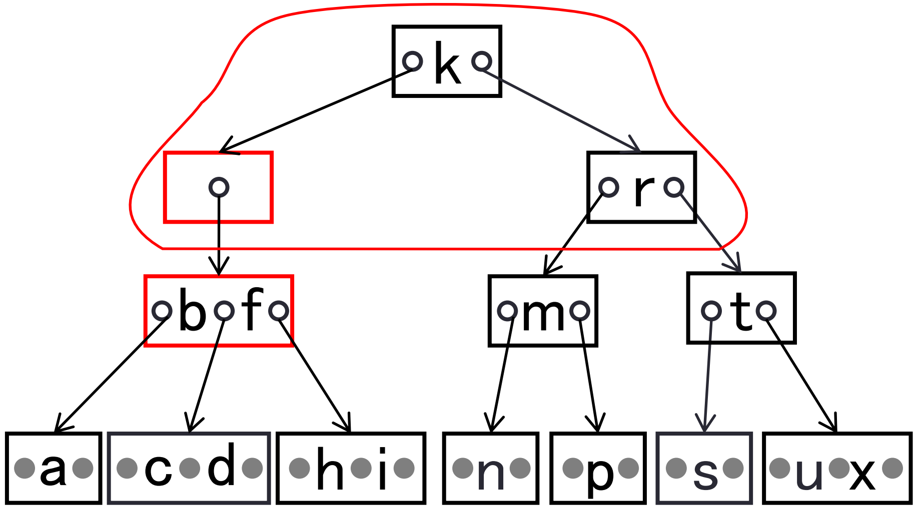


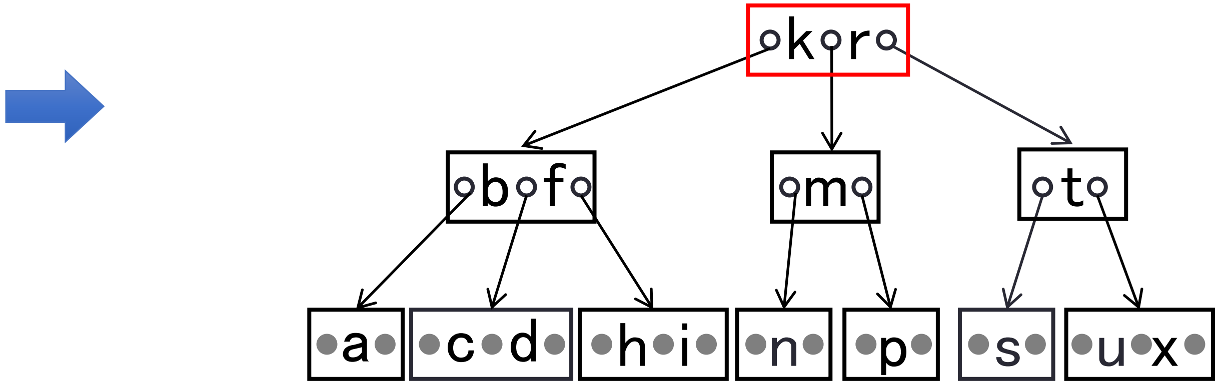


節点のキーが 0 個、隣のキーも 1 個より、
分割キーも含め隣の節点と併合。

つづき

- 内部節点どうしの併合 => 根になる

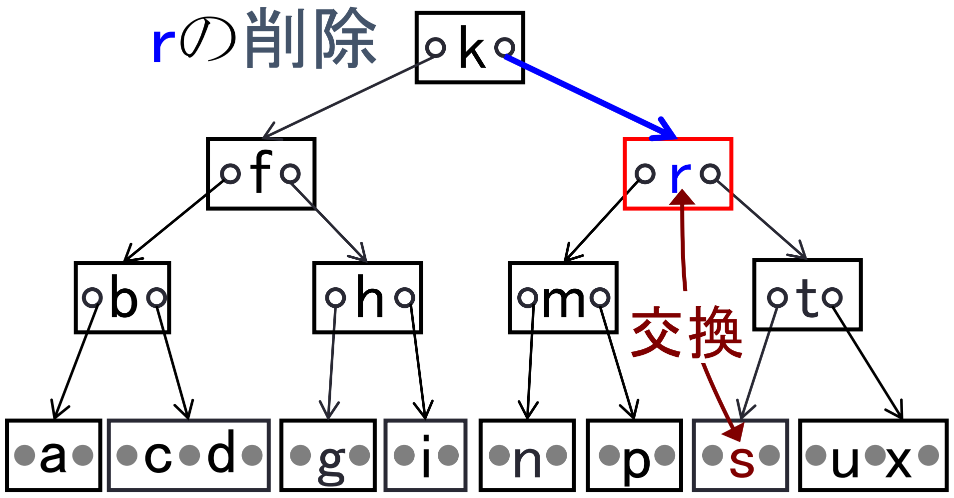


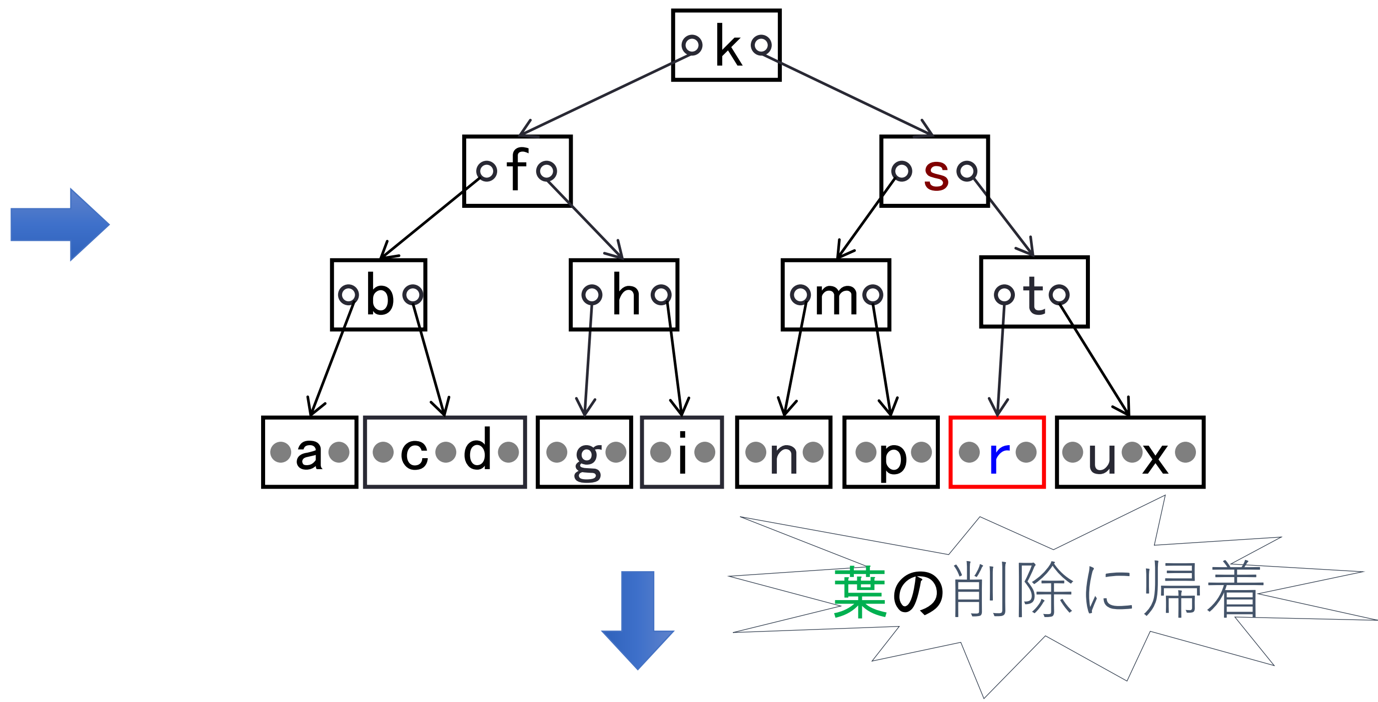


節点のキーが０個、隣のキーも１個より、
分割キーも含め隣の節点と併合。

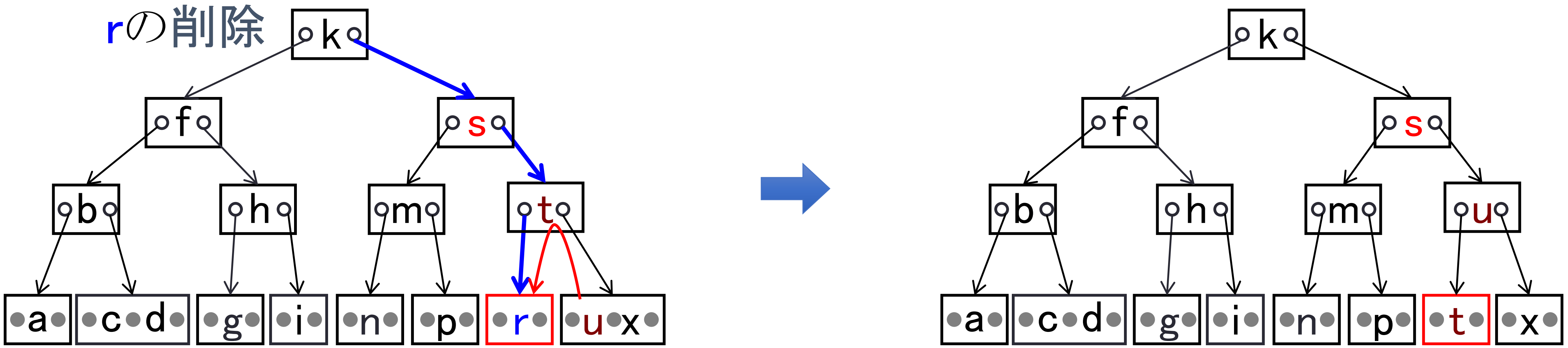
【例】2-3木のキー削除（内部節点から）

- 右部分木の最左キーと交換 => 葉のキーの削除





つづき



葉のキーが１個、隣のキーが２個より、
キーの削除とともに、隣からキーを借りる。

操作の時間計算量

- キーの総数をN，木の高さをhとすると（次数はm）

$$\log_m(N+1) - 1 \leq h \leq \log_{\lceil \frac{m}{2} \rceil} \frac{N+1}{2}$$

- 探索・挿入・削除の時間計算量は最悪でも

$$O(\log N)$$

- 例えば,
  - キーの総数 $N = 2^{24} - 1 = 16{,}777{,}215$
  - 次数 $m = 2^9 = 512$

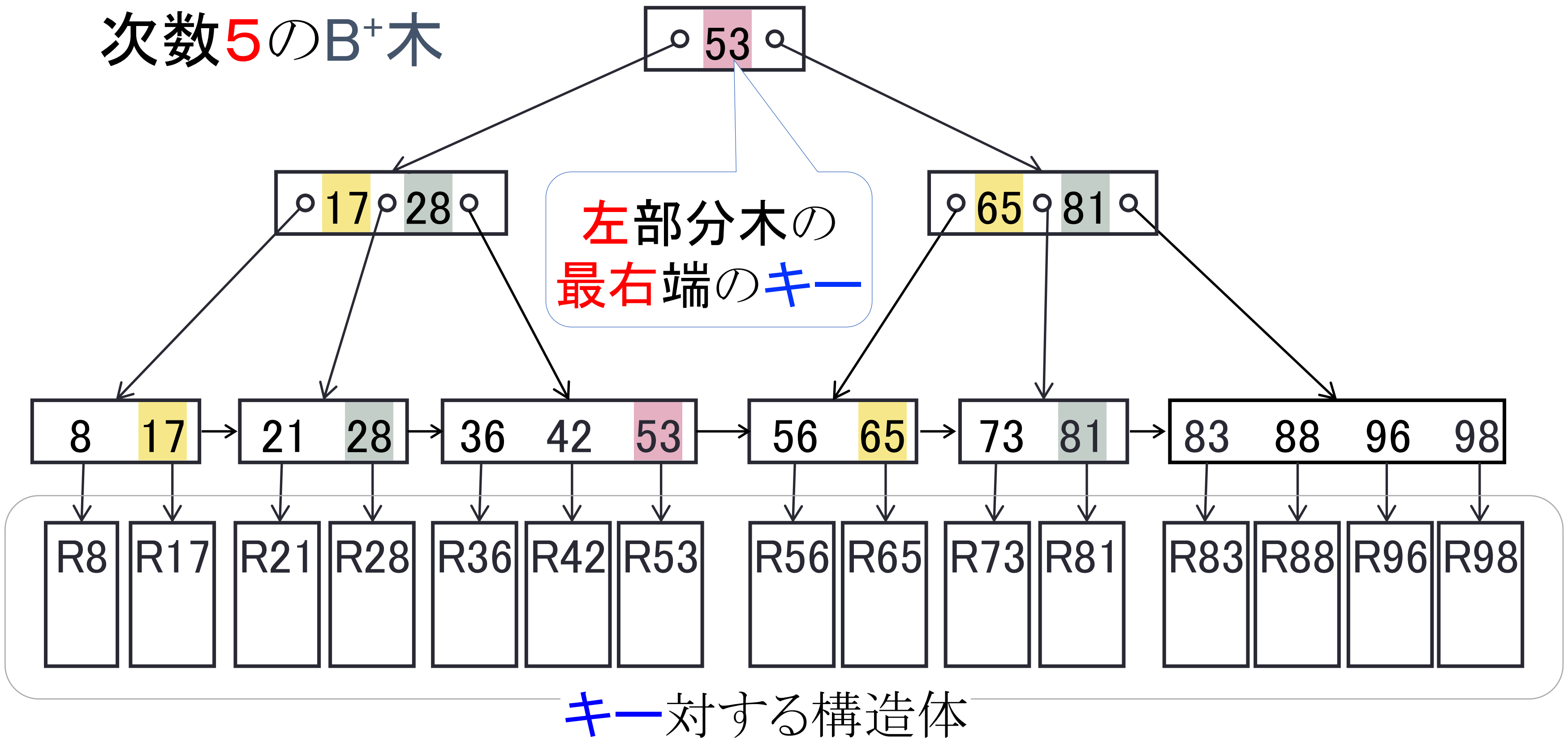
$$\log_{2^9} 2^{24} - 1 \leq h \leq \log_{2^8} 2^{23}$$

$$\frac{5}{3} \leq h \leq \frac{23}{8}$$

⇒探索・挿入・削除は，最悪$h+1$回なので，3回のアクセスですむ

B⁺木

- 葉にすべてのキーを持つ
- 内部節点には，分割点を示すキーが入る
- 葉どうしは，線形順序で連結し，キーの順次アクセスに適する



B*木は2/3
B木は1/3

B*木

- 内部節点における子の数の下限が$\left\lceil\frac{2m}{3}\right\rceil$の多分探索木である
  - これにより，節点のキー格納効率がよい
- キーの総数をN，木の高さをhとすると

$$\log_m(N+1)-1 \leq h \leq \log_{\left\lceil\frac{2m}{3}\right\rceil}\frac{N+1}{2}$$

- 木の高さの上限が，B木より小さい
- 分割は，隣り合う節点のキーが一杯になるまで待たされ，3つの節点に分割される
  - 節点のキーは一杯になっても，隣に空きがあれば，分割せずにキーを移す
- 探索・挿入・削除における探索経路は短くなる
- 節点中のキー数の下限が$\left\lceil\frac{2m}{3}\right\rceil-1$のため，分割・併合が起きやすい

まとめ

- B木による，辞書の実現を行った
  - 多分探索木にバランス
  - 子の最大数m（次数）の時 (m-1)-m木と言う

演習問題 1

- 4-5木の性質について，次の文の(a)から(e)まで埋めてください

4-5木とは次数 (a) のB木である．

根以外の節点が，内部節点のときの子の個数は (b) から (c)まで，
葉の時の子の個数は (d) である．

内部節点のキーの個数は子の個数より (e) 少ない

演習問題 2

- 4-5木に，次の整数をキーとして加えていく

7, 1, 6, 2, 11, 4, 8, 13, 10, 5, 19, 9, 18, 24

1. 最初に，**7, 1, 6, 2**を加えた直後の4-5木を描きなさい
2. さらに，**11**を加えた直後の4-5木を描きなさい
3. さらに，**4, 8, 13**を加えた直後の4-5木を描きなさい
4. さらに，**10**を加えた直後の4-5木を描きなさい
5. さらに，**5, 19, 9, 18**を加えた直後の4-5木を描きなさい
6. 最後に，**24**を加えた直後の4-5木を描きなさい

提出方法

- 提出無し