関数型言語（OCaml 演習）(12)

付録：赤黒木

浅井 健一

お茶の水女子大学

赤黒木

２分探索木の各節に、赤か黒の色がついており、
さらに以下の２条件を満たすもの。

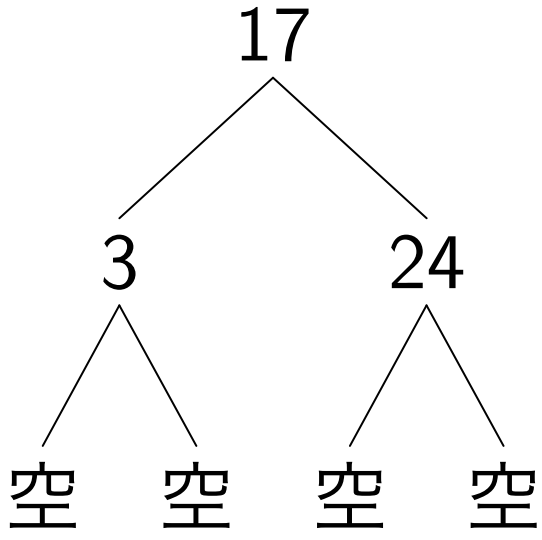
1. 根から空の木に至る全てのパスで黒の数は同じ。
2. 赤の節の子供は、どちらも黒。（赤は連続しない。）

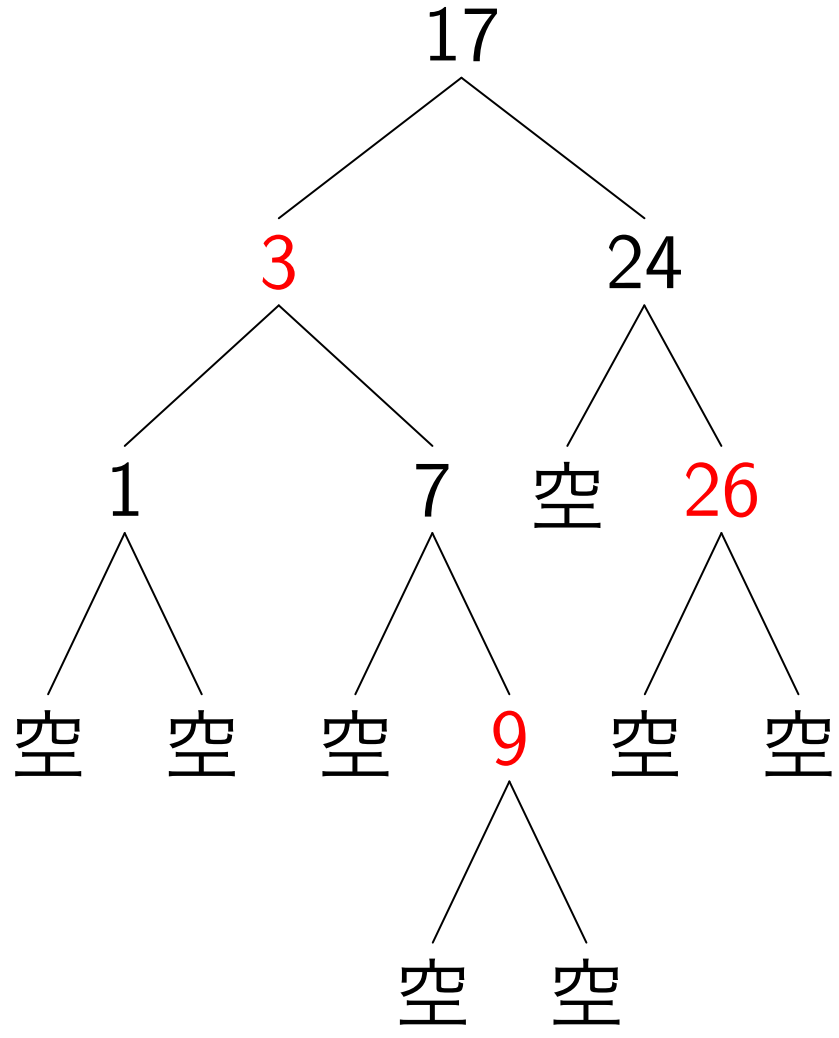
（空の木は、便宜的に黒と考える。）

赤黒木は「適度に」バランスした木になっている。

- 赤の節がなければ、完全にバランスした木になっている。
- 赤の節が入るとバランスは崩れるが、赤は連続して現れないので、数が制限される。

赤黒木の例





いずれも、根から
空の木に至るパスの
上に黒の節はふたつ。

赤は連続していない。

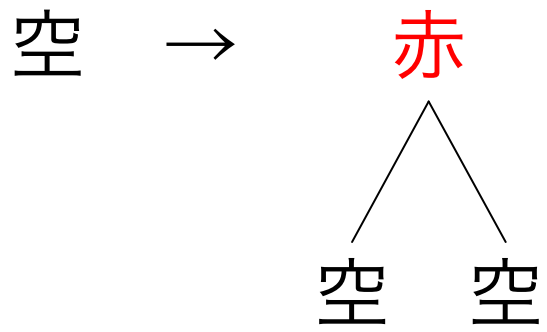
赤黒木の実装

```
(* 赤か黒かを示す型 *)
type color_t = Red | Black

(* キーが 'a 型、値が 'b 型の赤黒木の型 *)
type ('a, 'b) t = Empty
  | Node of ('a, 'b) t * 'a * 'b * color_t * ('a, 'b) t

(* empty : ('a, 'b) t, 空の赤黒木 *)
let empty = Empty

(* 目的：木の中からキーに対応する値を探して返す *)
(* なければ例外 Not_found を起こす *)
(* search : ('a, 'b) t -> 'a -> 'b *)
let rec search tree k = ... (* ２分探索木とほぼ同じ *)
```

赤黒木への挿入

```
(* 目的：赤黒木にキーと値を挿入した赤黒木を返す *)
(* insert : ('a, 'b) t -> 'a -> 'b -> ('a, 'b) t *)
let rec insert tree k v = match tree with
    Empty -> ...
  | Node (left, key, value, color, right) ->
      ... (* insert left k v *)
          (* insert right k v *)
```

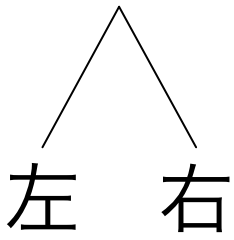
木が空の場合：新しく赤い節を作る。



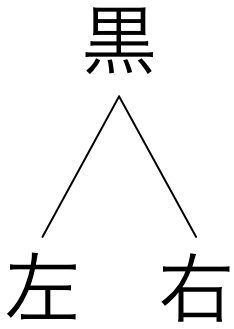
木が空ではない場合

普通の２分探索木のときと同じように、キーの値によって左、または右の木に挿入する。その上で：

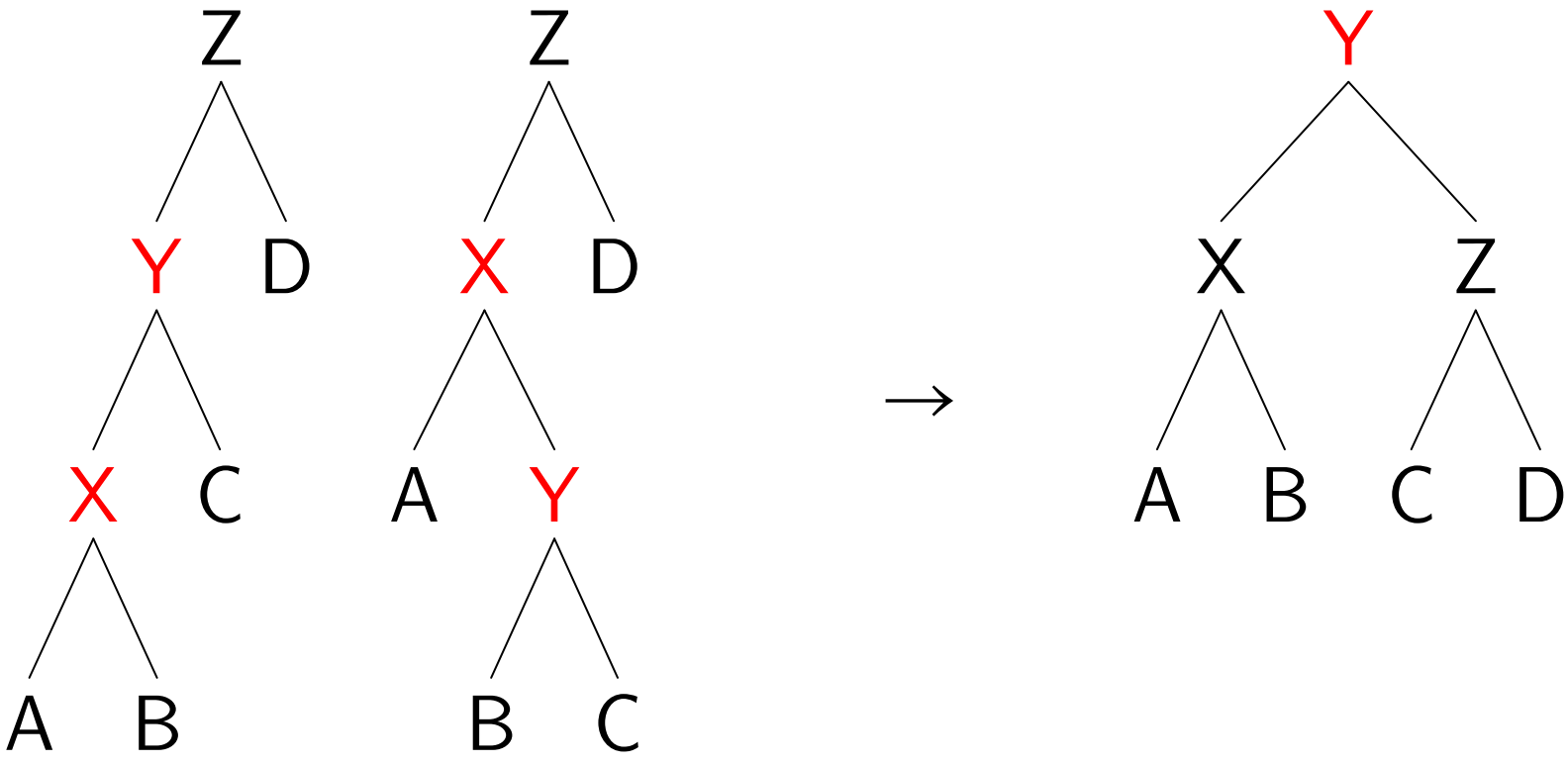
現在の節が赤の場合　挿入した結果、左または右の子が赤だと困る。が、ここでは放置して、親のところで対処する。



現在の節が黒の場合　子供と孫を見て、赤の連続が見つかったら、木を再構成する。

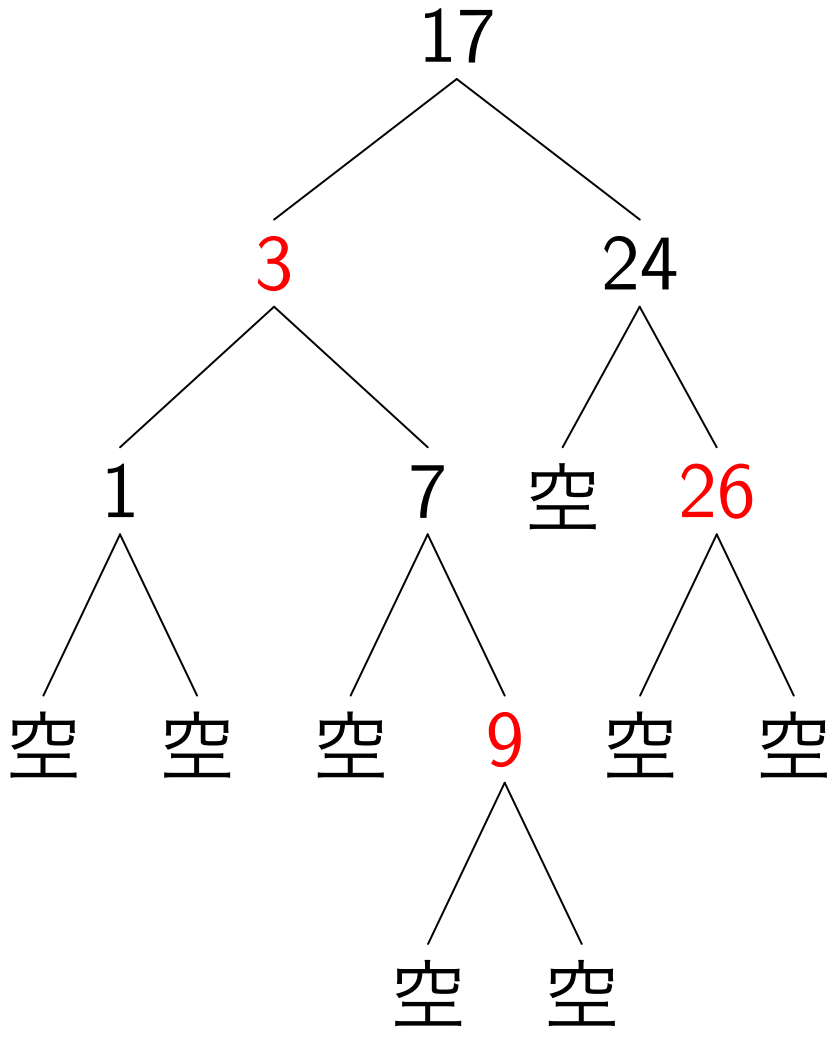


木の再構成

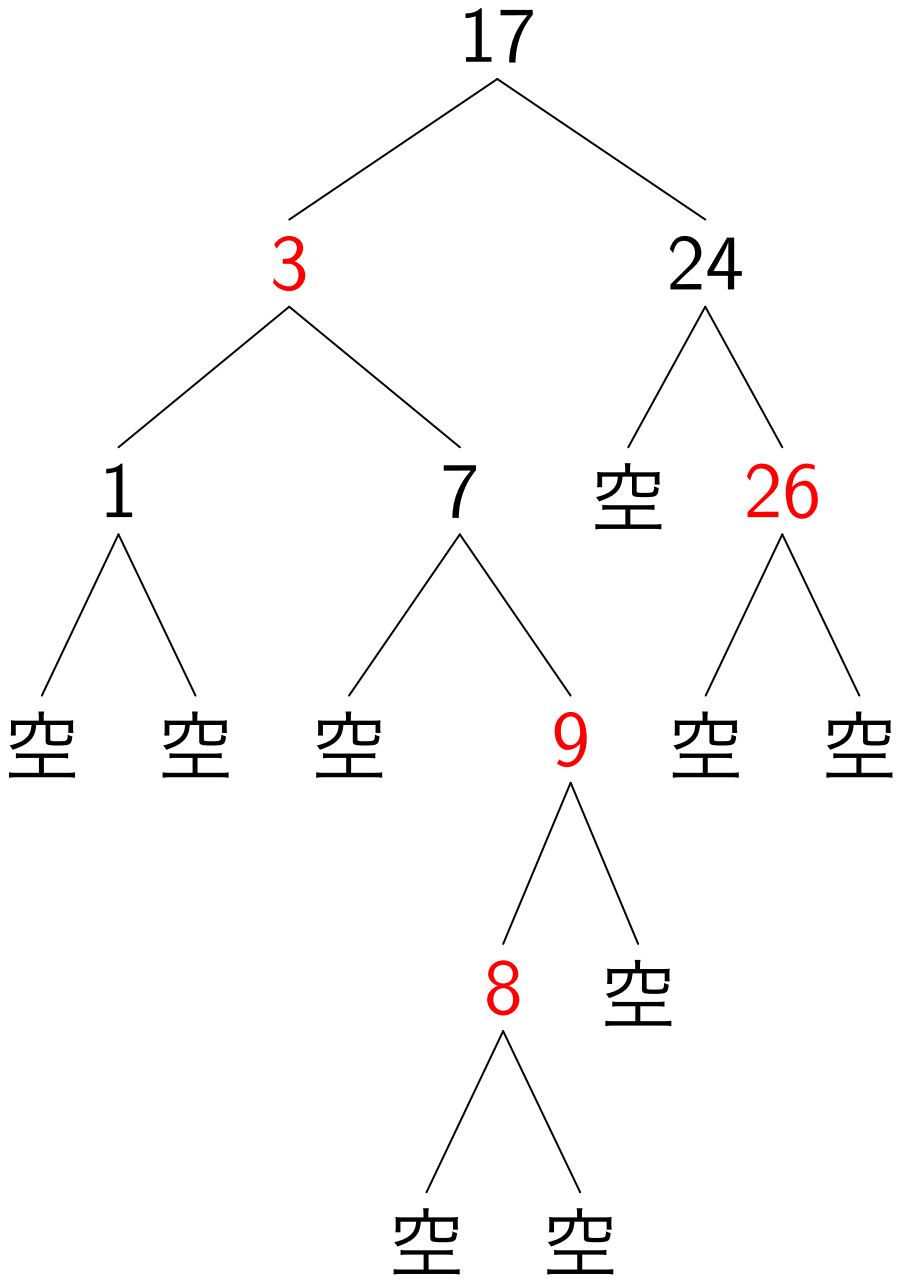


- 大小関係を崩すことなく、赤の連続がなくなった。
- パス上の黒の数は不変。
- 根が赤になっている。→ さらに親で再構成。

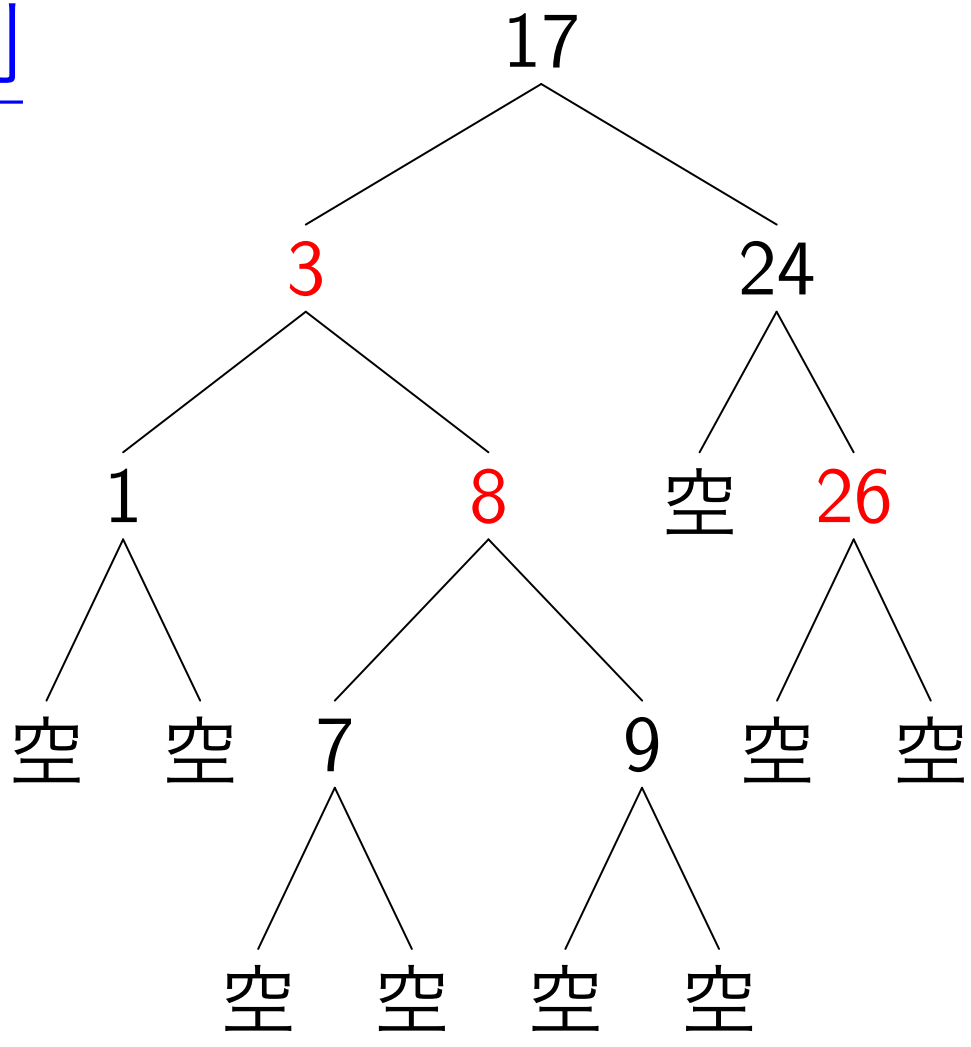
再構成の例



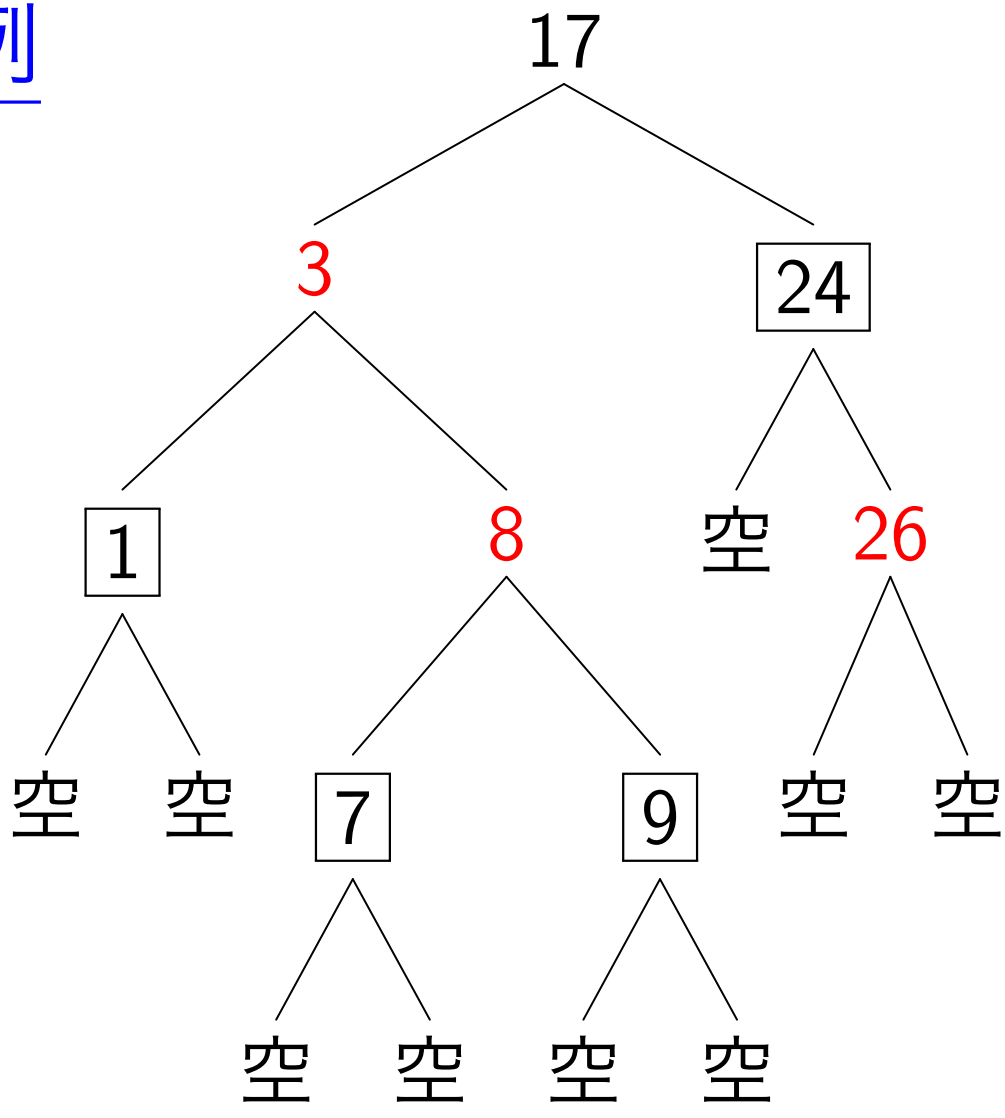
再構成の例



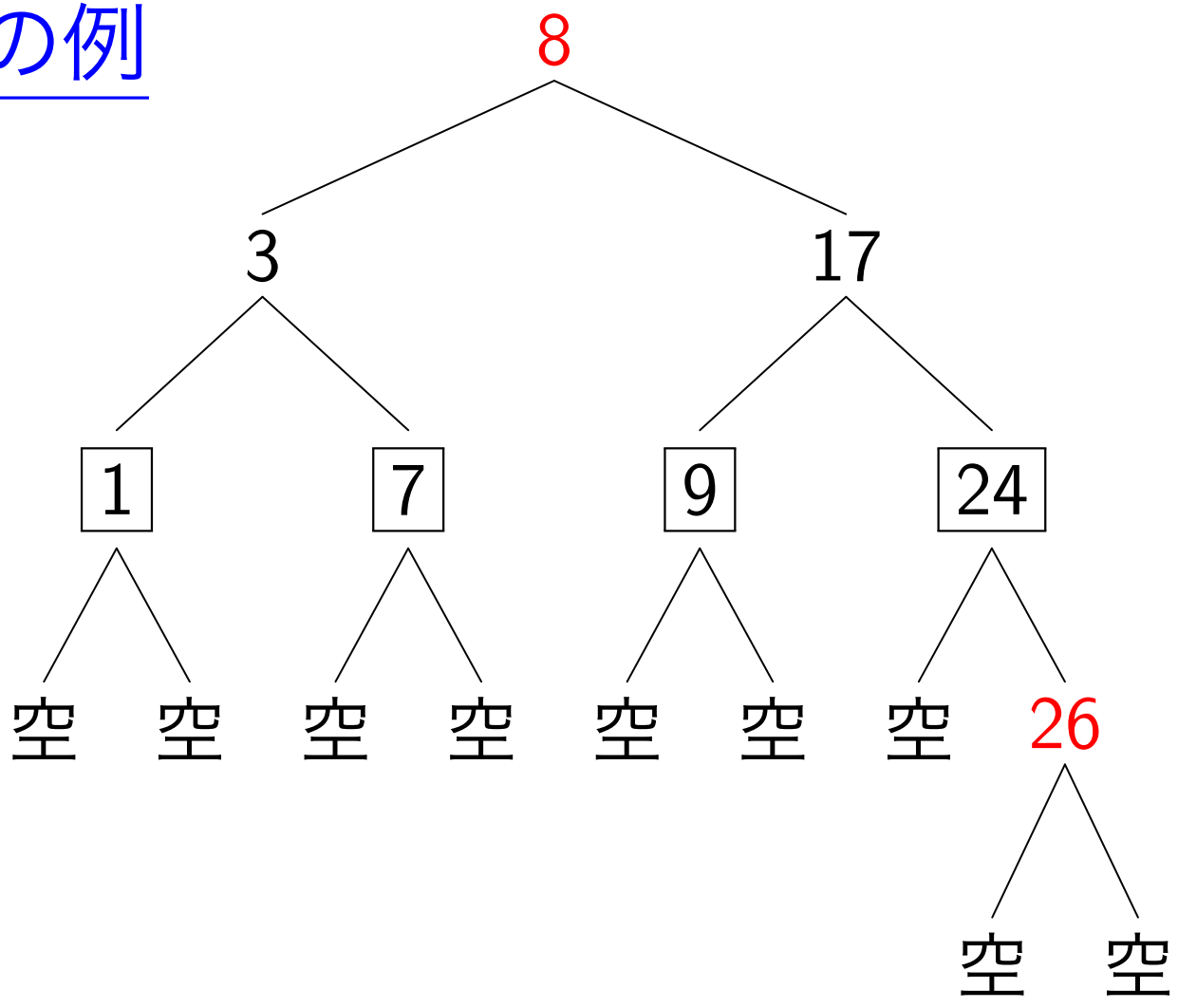
再構成の例



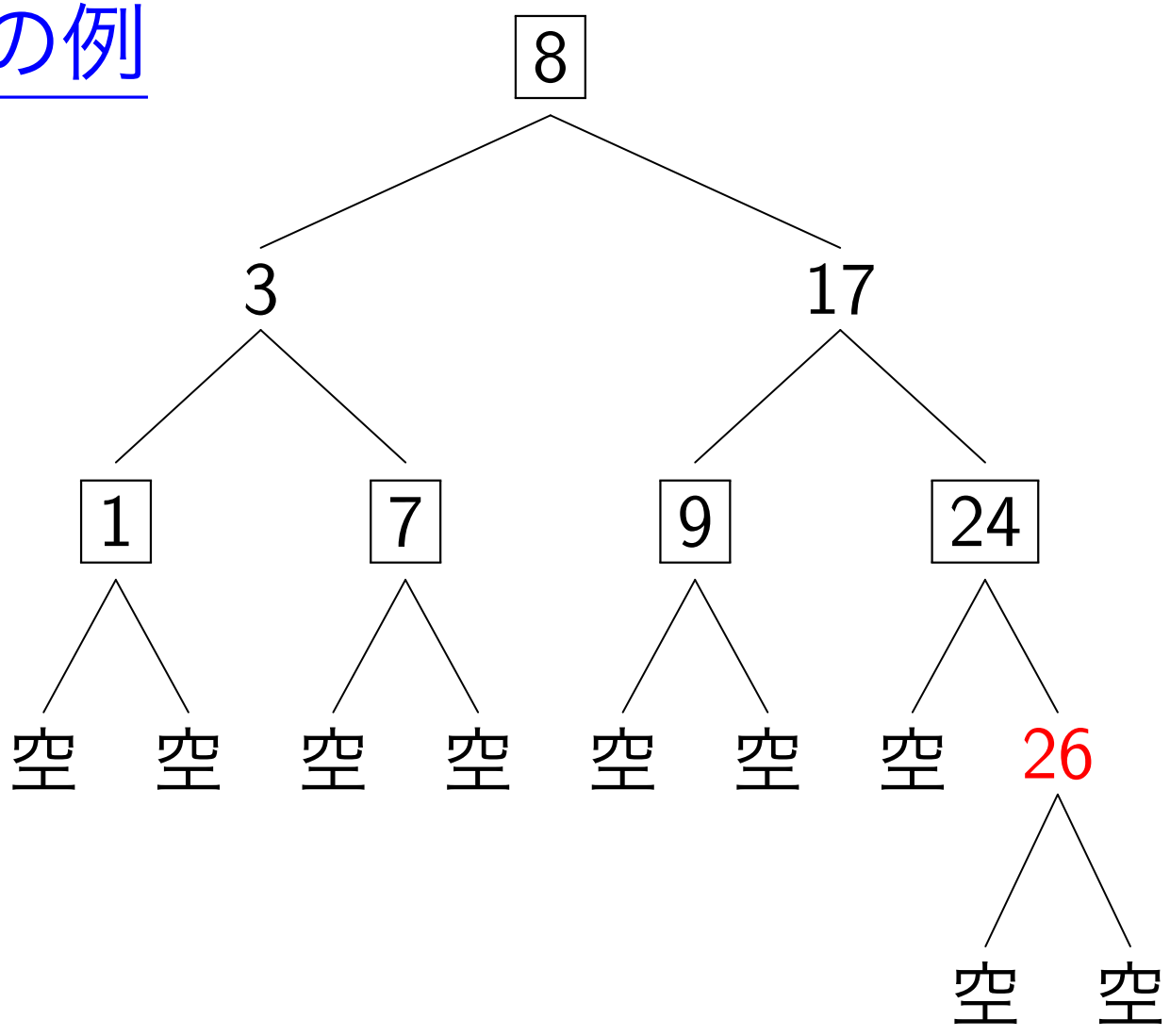
再構成の例



再構成の例



再構成の例



根が赤になったら、最後にそれを黒にする。

モジュールの入れ替え

２分探索木のモジュール

```
type ('a, 'b) t = Empty
  | Node of ('a, 'b) t * 'a * 'b * ('a, 'b) t
...
```

赤黒木のモジュール

```
type ('a, 'b) t = Empty
  | Node of ('a, 'b) t * 'a * 'b * color_t * ('a, 'b) t
...
```

シグネチャ

```
type ('a, 'b) t    (* 型の中身（構成子）は非公開 *)
...
```

実装は異なるが、シグネチャは同一。
→ どちらを使ってもプログラムは同じように動く。

まとめ

赤黒木：節に赤か黒の色がついた２分探索木

- 根から空の木に至る全てのパスで黒の数は同じ。
- 赤の節の子供は、どちらも黒。（赤は連続しない。）

適度にバランスした木を表現できる。

赤が並んだら、その親と共に木を再構成する。

シグネチャを２分探索木と同一にしておくと、

- ２分探索木の代わりに使うことができ、
- 木がバランスすることが保証される。