データ構造とプログラミング技法
（第９回）

ーデータの探索（B木／トライ／パトリシア木）ー

B木

- 目的：大規模、且つ、動的に変化するデータを効率良く探索するためのデータ構造。磁気ディスク上のデータを効率よく探索するために用いられる.
- 構造：

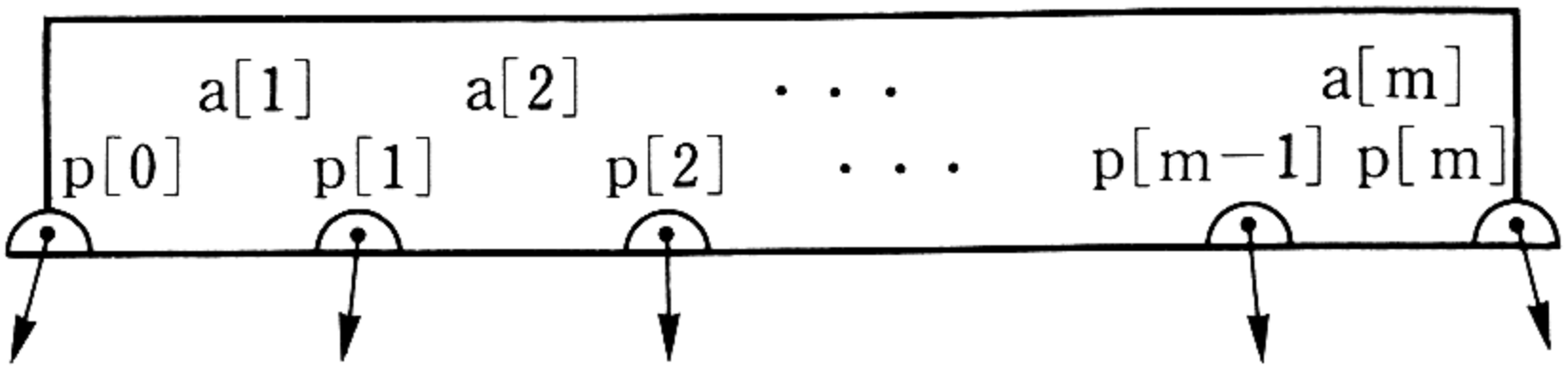


図 8.12　B 木の節の構成

B木の定義

〔1〕 k 次 の B 木

(1) 根以外の任意の節には，k 個以上，$2k$ 個以下のキーが格納される．それらのキー（m 個とする）を a[1]，…，a[m] とする．

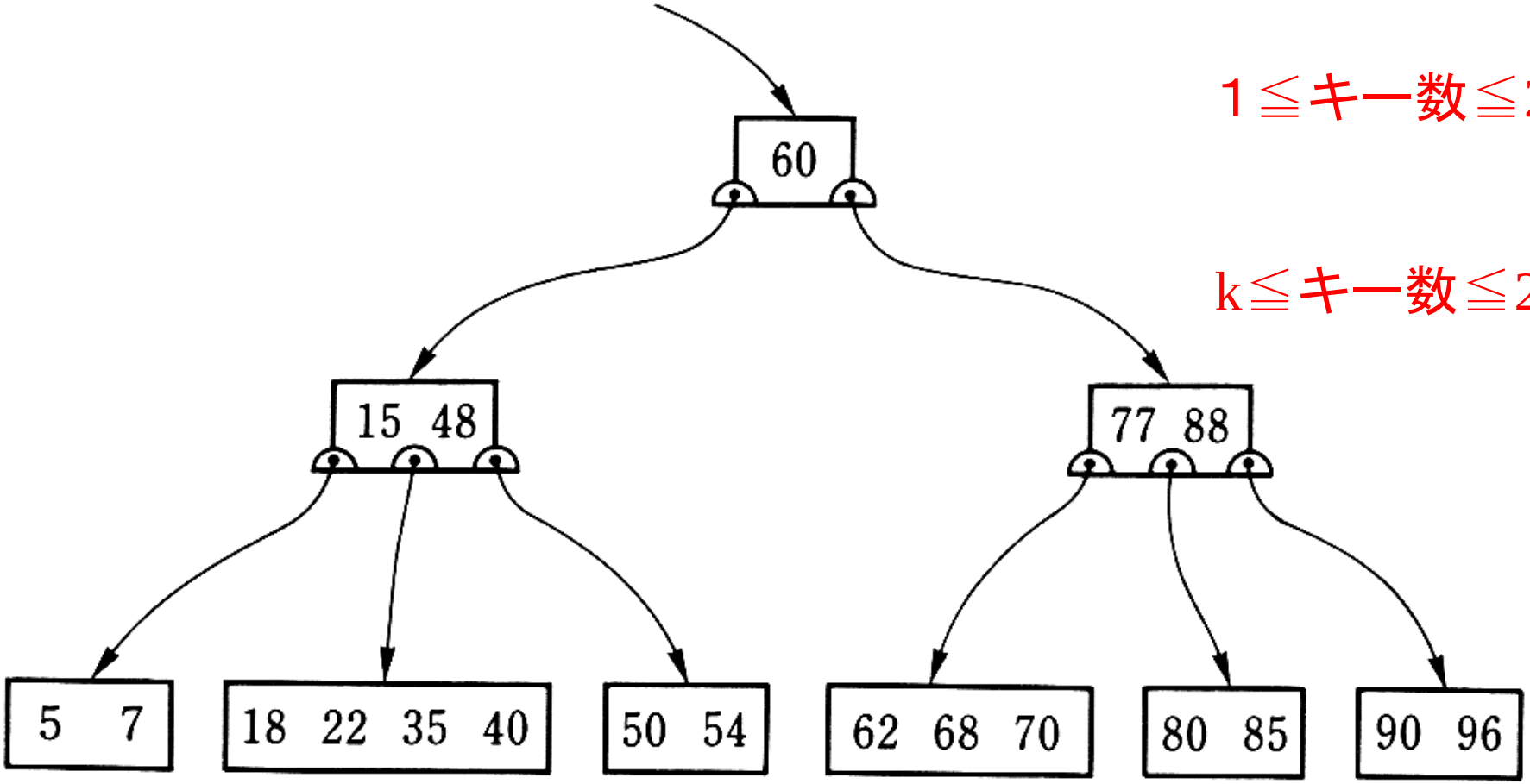
(2) 根には 1 個以上，$2k$ 個以下のキーが格納される．

(3) 葉以外の任意の節には，部分木へのポインタが $m+1$ 個ある．ただし，m はその節に格納されているキーの個数である．これらのポインタを p[0]，p[1]，…，p[m] とする．

(4) すべての葉は同一レベルにある．

(5) 任意の節において，キー列 a[1]，…，a[m] は整列している．また，キー a[i]（$1 \leqq i \leqq m$）は，p[i−1] の指す部分木内のどのキーよりも大きく，逆に p[i] の指す部分木内のどのキーよりも小さい．

B木の構造

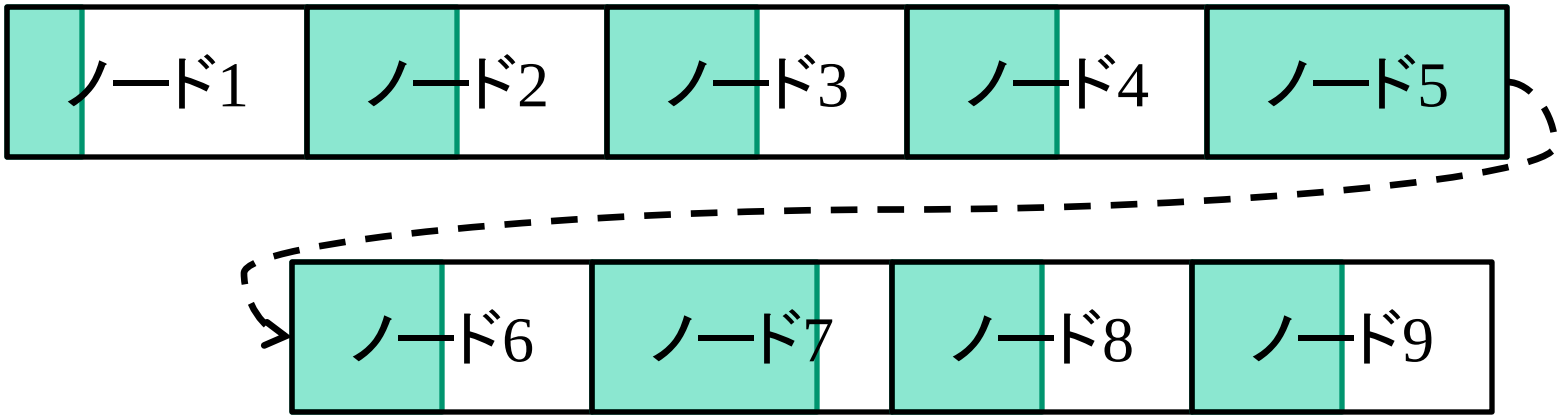


1≦キー数≦2k

k≦キー数≦2k

図 8.13　2 次の B 木の例

次数の必要性



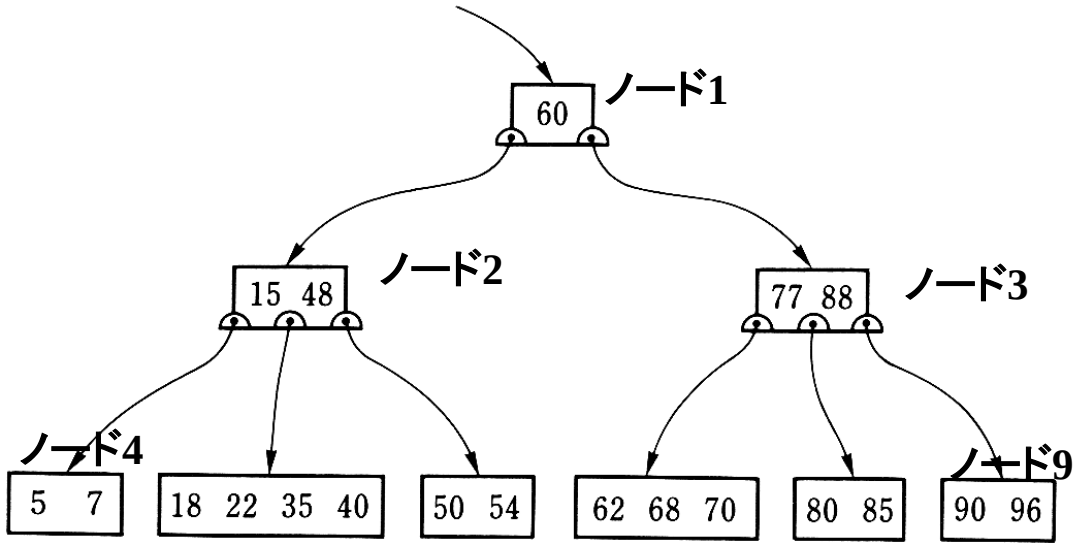


図 8.13　2 次の B 木の例

- 各ノードは磁気ディスク装置の固定サイズレコードに対応
- 次数を決めないと各ノードのレコードが有効に使えない

各節内では二分探索

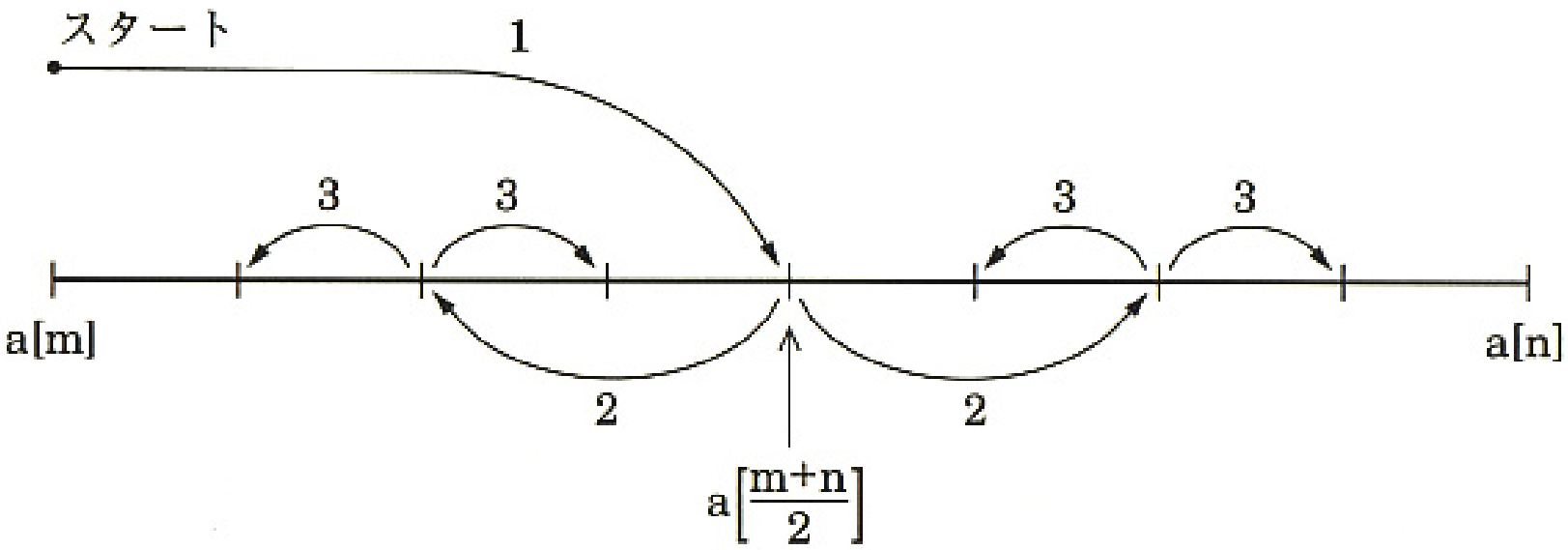
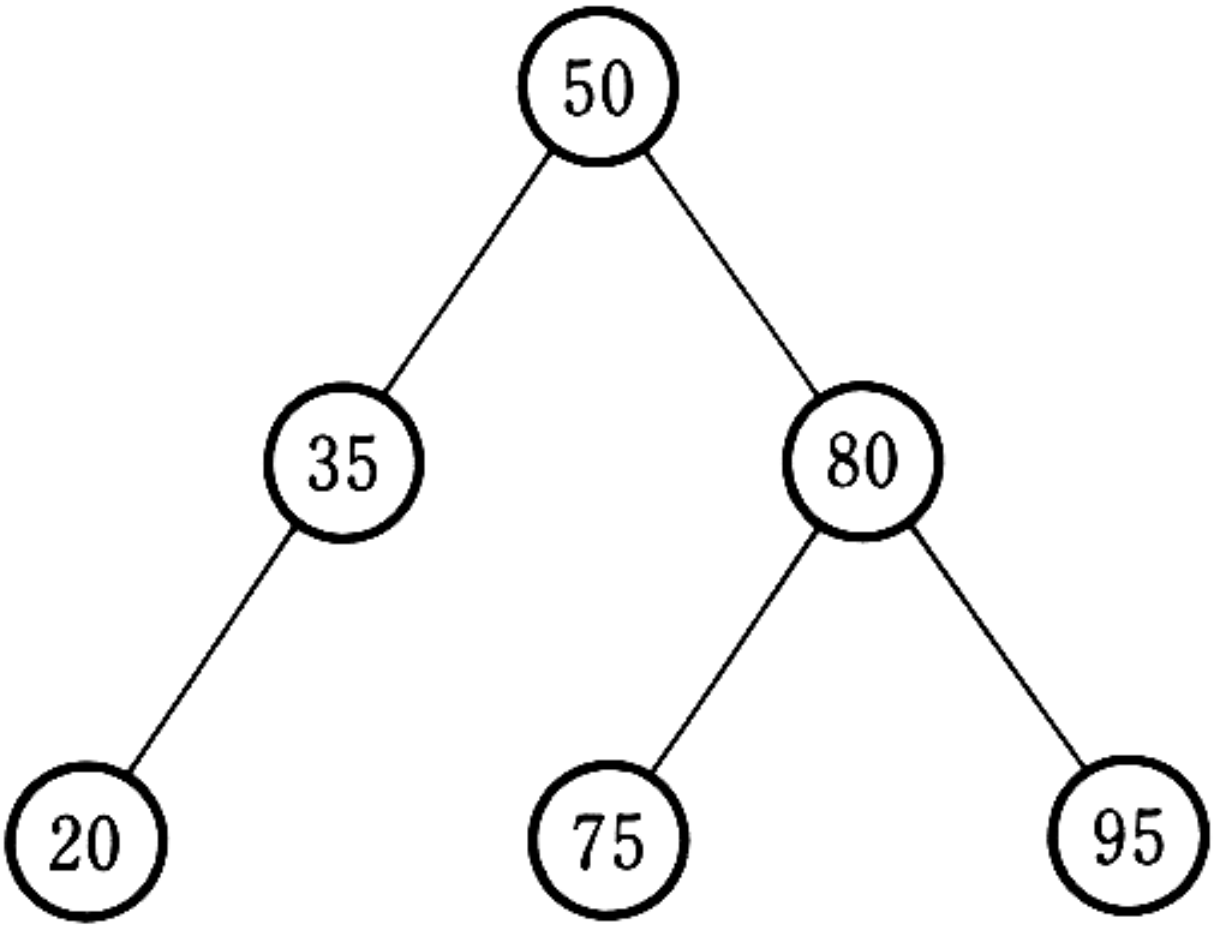


図 8·2　二分探索の探索過程

節の間では木探索



B木の構造変化

- データの追加，削除を行った際に
  - 根以外のノードはk以上2k以下のデータを格納
  - 根には1以上2k以下のデータを格納
  - 全ての葉は同じレベルにある

  という条件が崩れる場合に，大きな構造変化が起きる.

キーの追加によるB木の変化：単純追加

10を追加

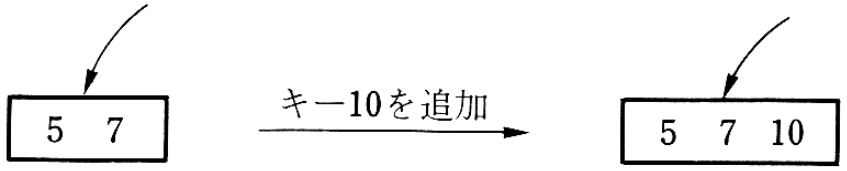


図 8.14　キー 10 の追加（単純追加）

図 8.13　2 次の B 木の例

キーの追加によるB木の変化：分割

一旦追加した後、
中央のキーを上に
引き上げる。
この場合は、たまた
ま30が上に引き上
げられた。

30を追加

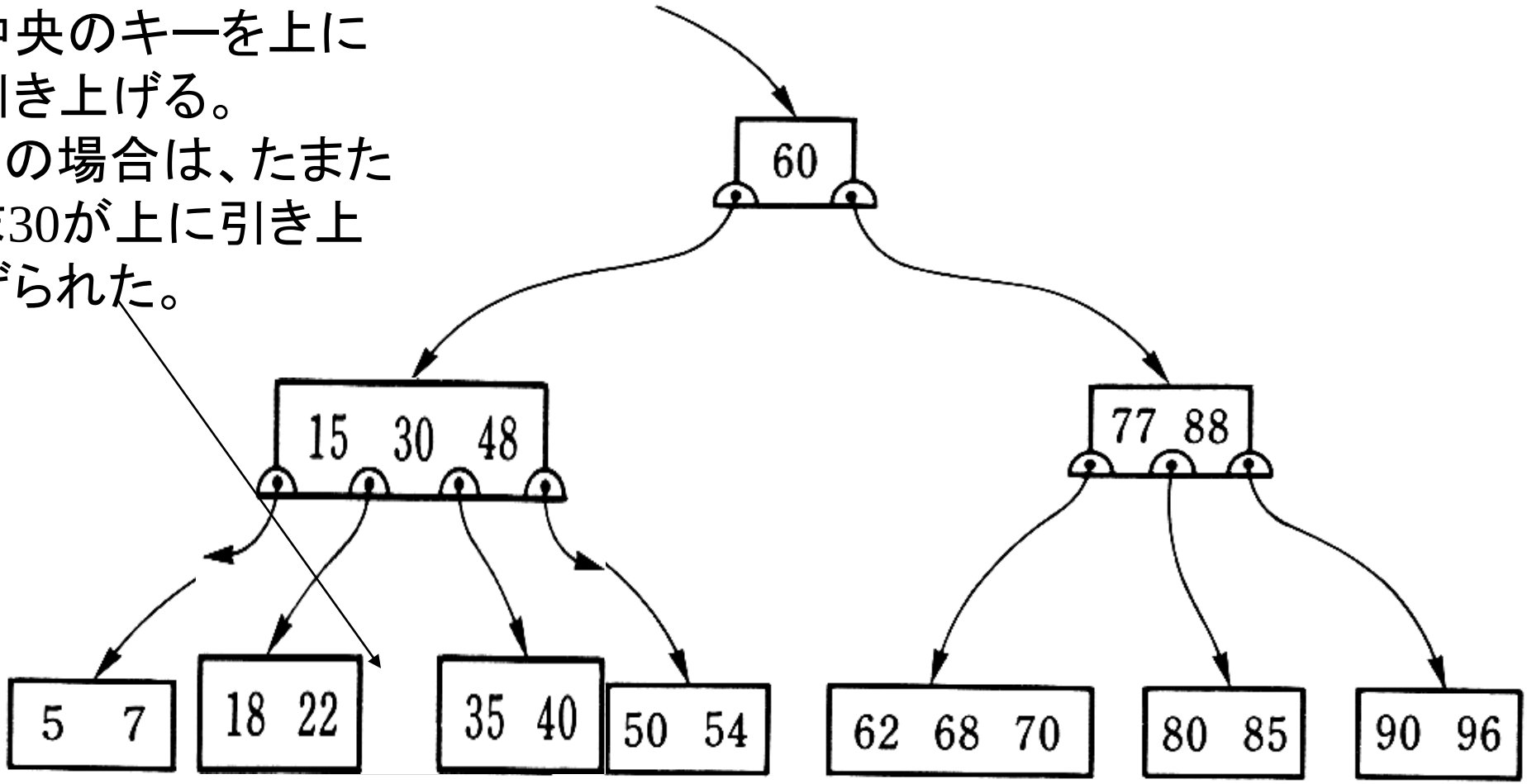


図 8.13　2 次の B 木の例

キーの削除によるB木の変化：単純削除

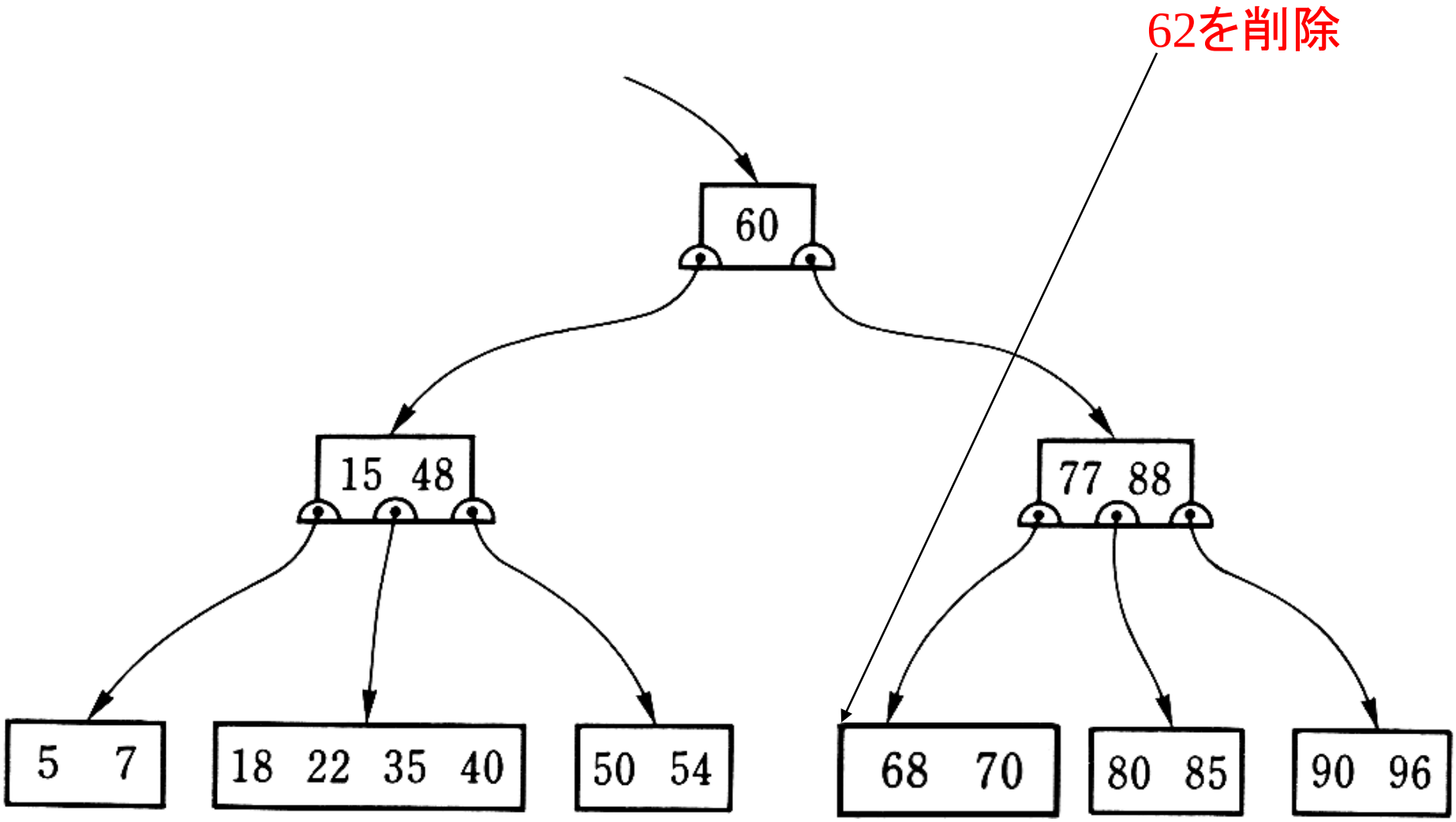


図 8.13　2 次の B 木の例

キーの削除によるB木の変化：アンダーフロー

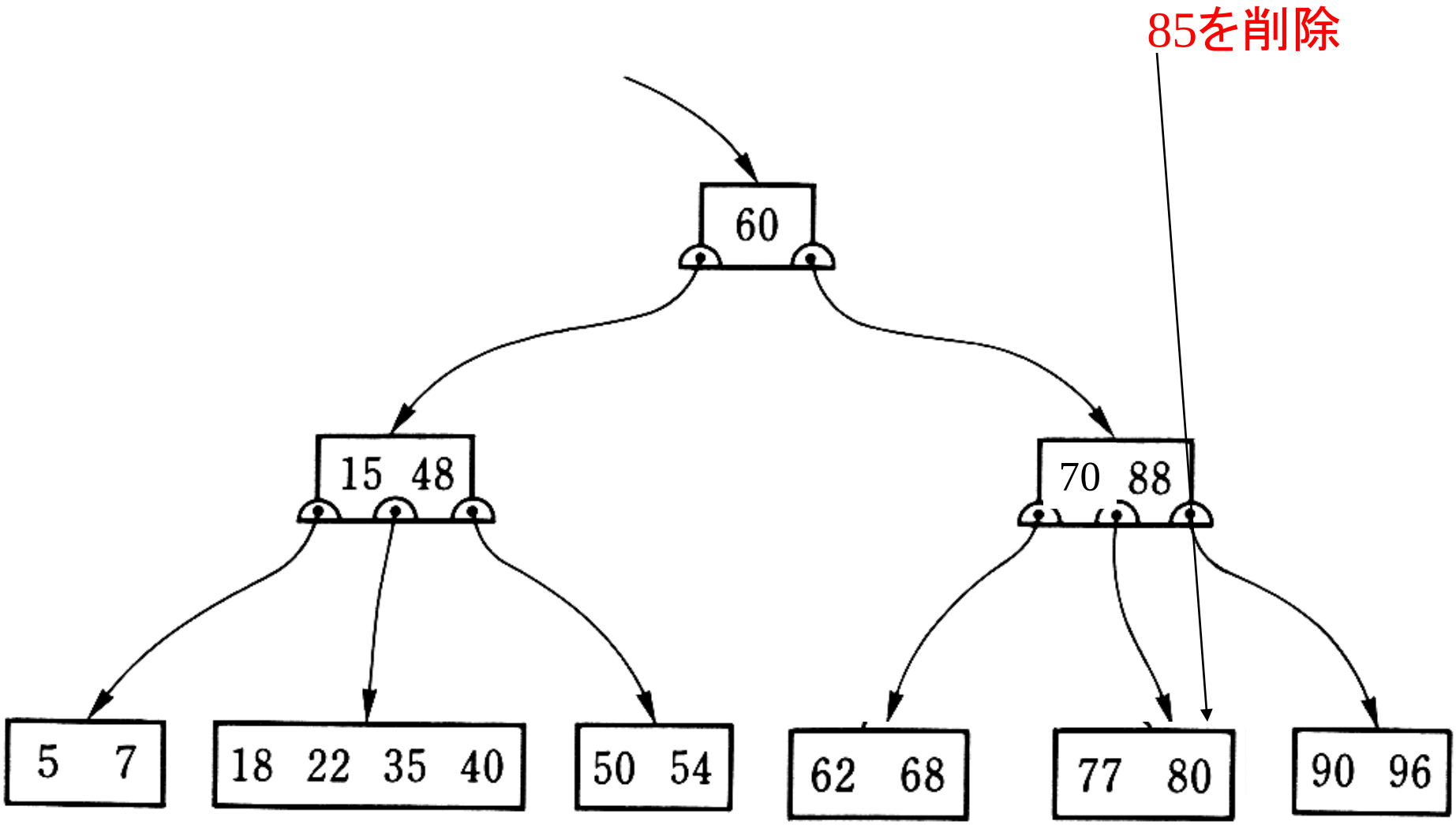


図 8.13　2 次の B 木の例

キーの削除によるB木の変化：連結

90を削除

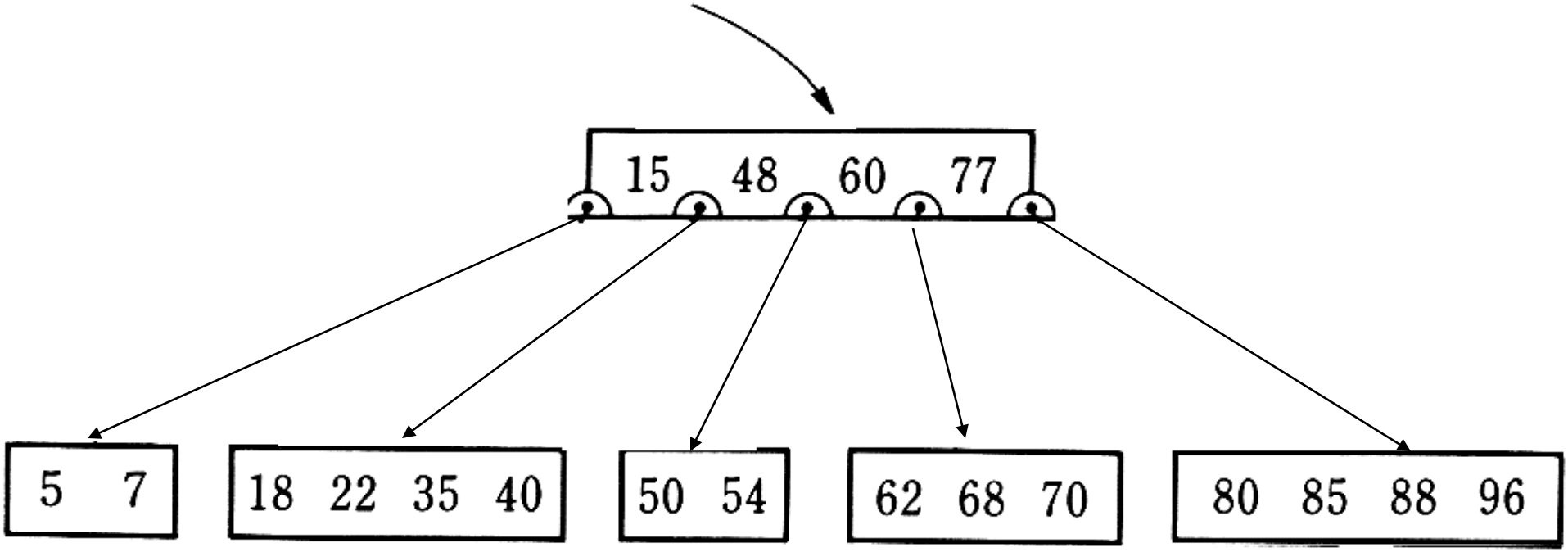


図 8.13　2 次の B 木の例

他の木探索用データ構造

- ユーザが項目を登録可能な辞書等のアプリケーションを想定
  - 適切なハッシュ関数を求めることが困難.
  - B木や2分探索木などは，登録時の計算コストが高い.
- 桁探索木

桁探索木:トライ

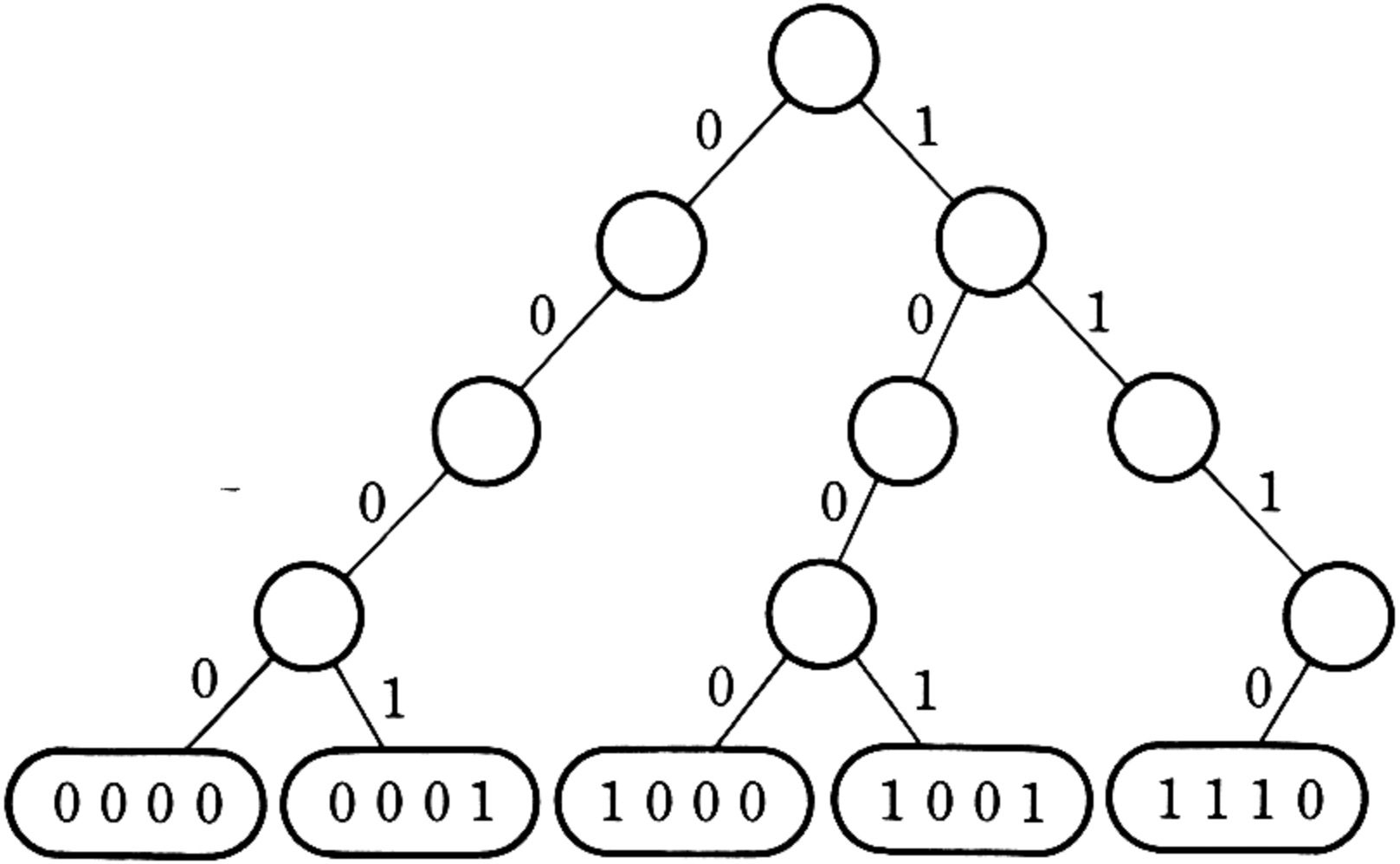


図 8.19　トライの例

桁探索木:パトリシア木

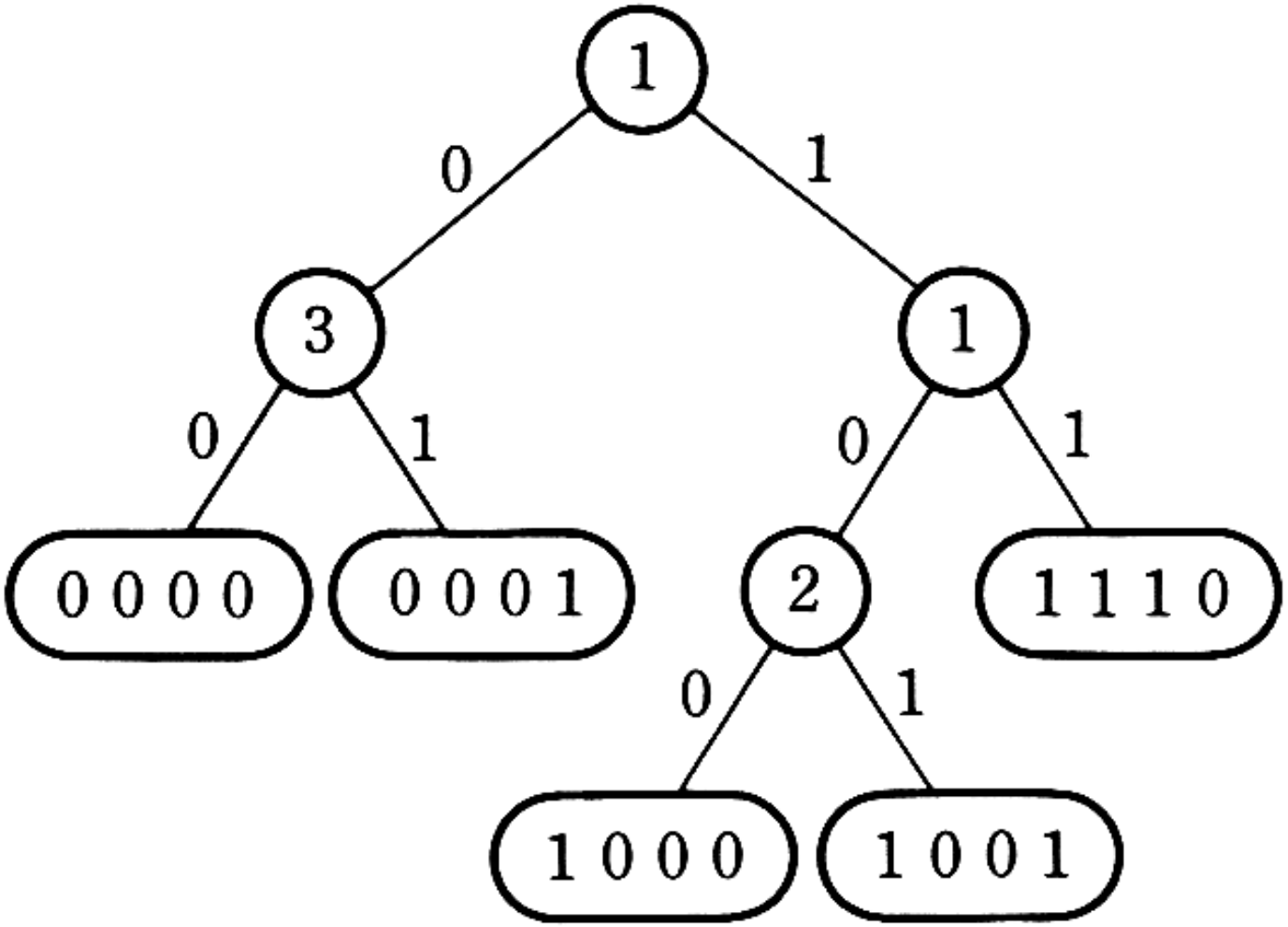


図 8.20　図 8.19 と等価なパトリシア木