基本情報技術Ⅰ

9. 2分検索木，ハッシング

菊池浩明

講義目標

- 教科書
  - 2-1 データ構造
    - » 4. 2分木
    - » 5. ハッシュ法

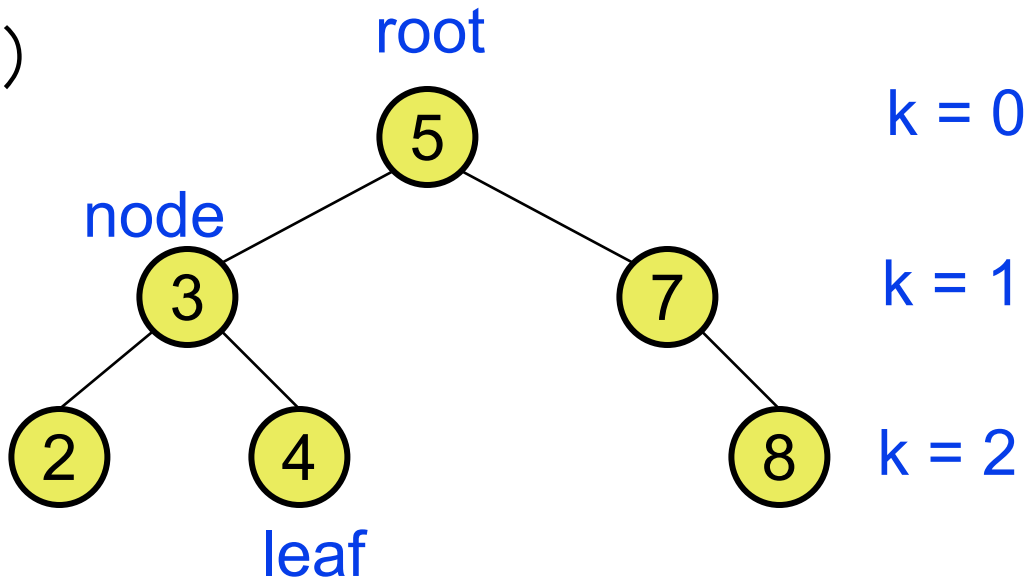
検索問題

- 問題
  - 次の表の金額から駅名を調べるプログラムを作れ.

料金	データ
150	新宿
160	四ツ谷
210	東京
290	新橋
380	立川

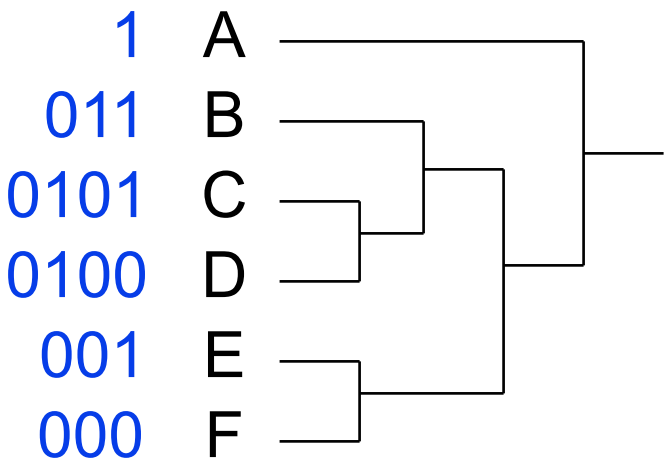
1. 二分探索木★★頻出

- 木構造
  - 節点（ノード，節）
  - 根（ルート）
  - 葉（リーフ）
  - 親，子
  - 部分木
  - 高さ（深さ）k



木構造の応用分野

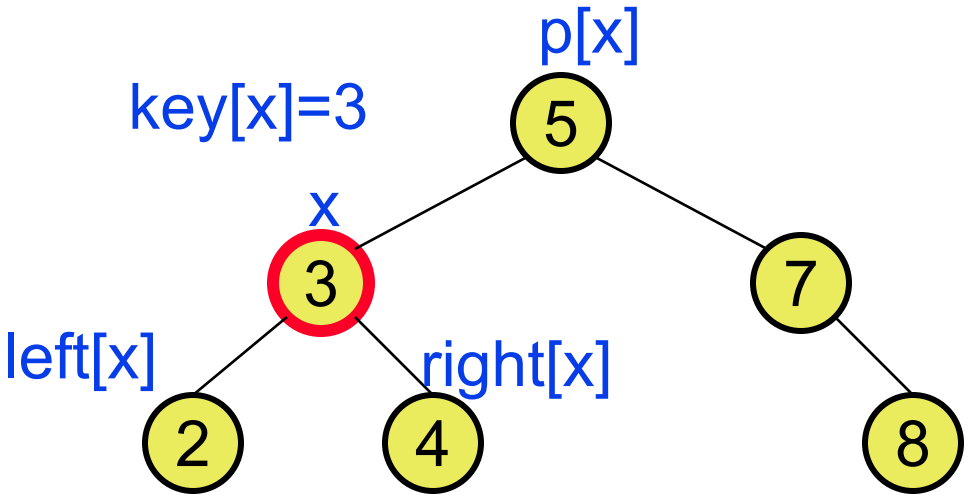
- ハフマン符号



- ドメインネームシステム(DNS)
  - www.meiji.ac.jp

2分探索木 (binary search tree)

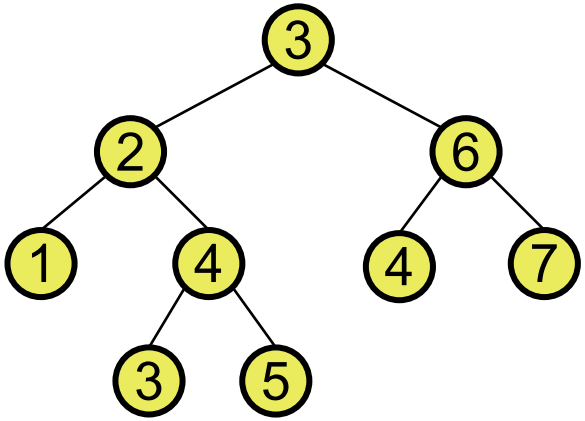
- 2分木
  - p (parent) 親
  - key キー
  - left 左部分木
  - right 右部分木

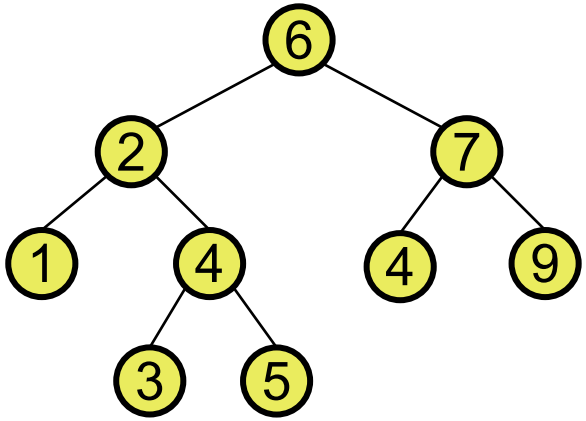


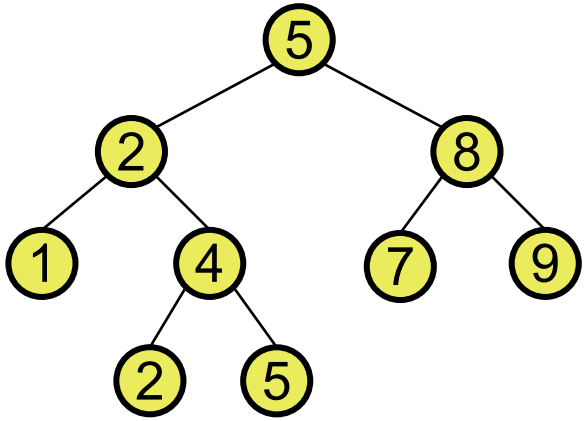
- 条件
  - 1. key[left[x]] ≤ key[x]
    （左部分木の要素は自分(親)より小さい）
  - 2. key[x] ≤ key[right[x]]

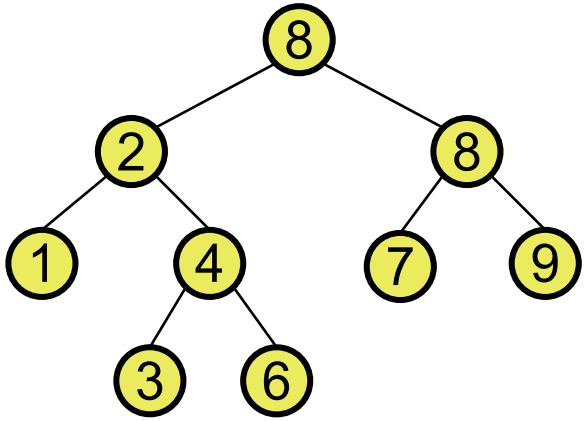
例1

- 2分条件を満たしているのはどれか？









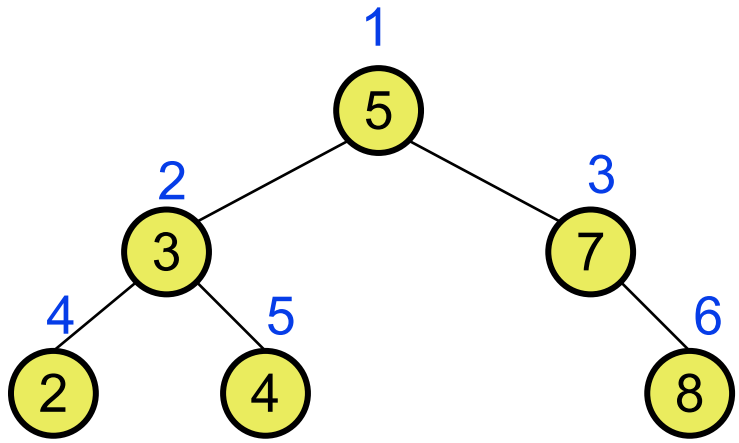
検索

- Search (x, k)
  1. If x = NIL or k = key[x]
  2. then return x
  3. If k < key[x]
  4. then return Search(left[x], k)
  5. else return Search(right[x], k)

  - NIL: 子（や親）が無いことを表す値
  - 再帰呼出し: 関数の中で自分自身を呼び出す（recursive call）

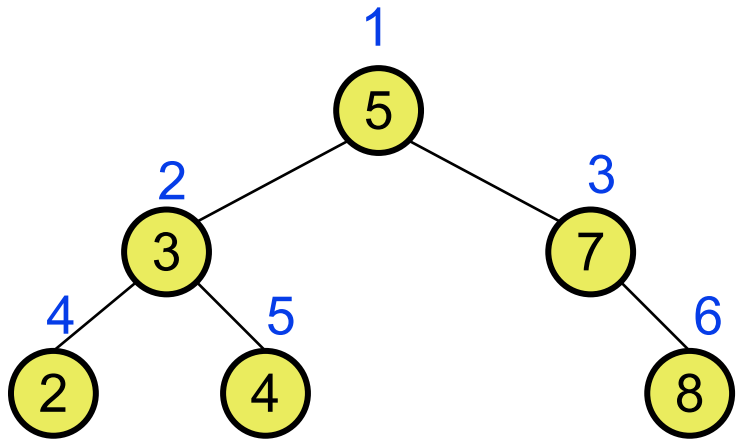
探索の例

- Search(木の根1, 検索値4)
  - 1. If 1 = NIL *no*
  - 3. if 4 < key[1] (=5) *yes*
  - 4. Search(left[1], 4)
- Search(2, 4)
  - 1. If 2 = NIL *no*
  - 3. if 4 < key[2] (=3) *no*
  - 5. Search(right[2], 4)
- Search(5, 4)
  - 1. if 5 = NIL or key[5] = 4 *yes* return 5



例2

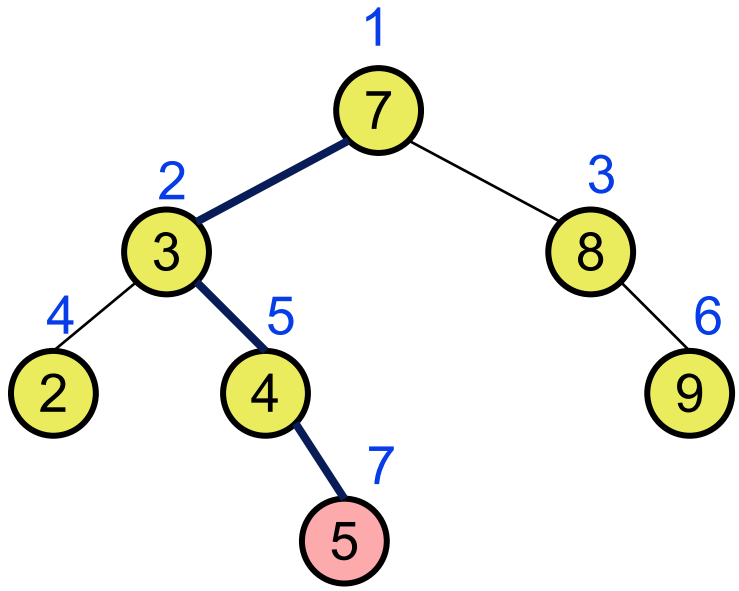
- Search(木の根1, 検索値8)
  - Search(right[1], 8)
  - Search(right[3], 8)
  - Return 6
- Search(1, 3)
  - Search(　　,　)
  - Return
- Search(1, 6)
  - Search(　　,　)
  - Return



要素の挿入

- Insert(T, z)

```
1.  x=root[T]
2.  while x != NIL
3.      y = x
4.      if key[z] < key[x]
5.          then x = left[x]
6.          else x = right[x]
7.  p[z] = y
8.  If key[z] < key[y]
9.     then left[y] = z
10.    else right[y] = z
```



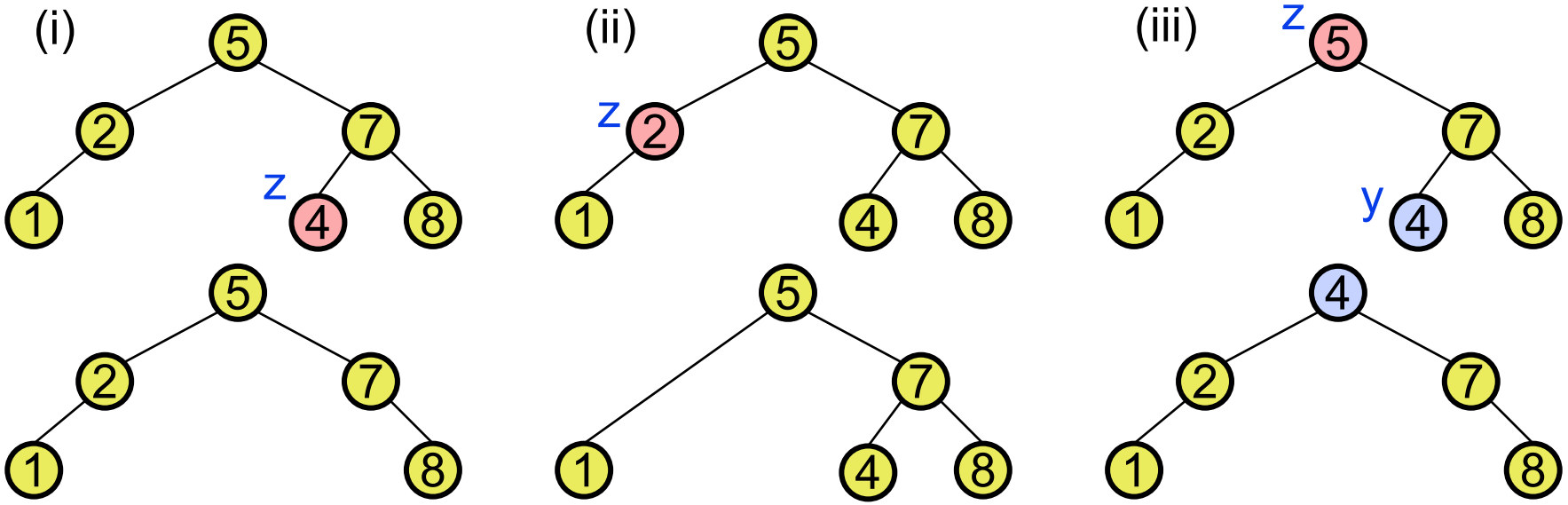
Insert(1, 5)の例
x = 1→2→5
z=7
p[7] = 5
right[5] = 7

例3 （木の構成）

- 1. 空の2分探索木に，次の順でデータを追加した二分探索木を求めよ.
  - A. 8, 12, 5, 3, 10, 7, 6
  - B. 10, 7, 6, 8, 12, 5, 3
  - C. 3, 5, 6, 7, 8, 10, 12
- 2. 最も効率がよいのはどれか？

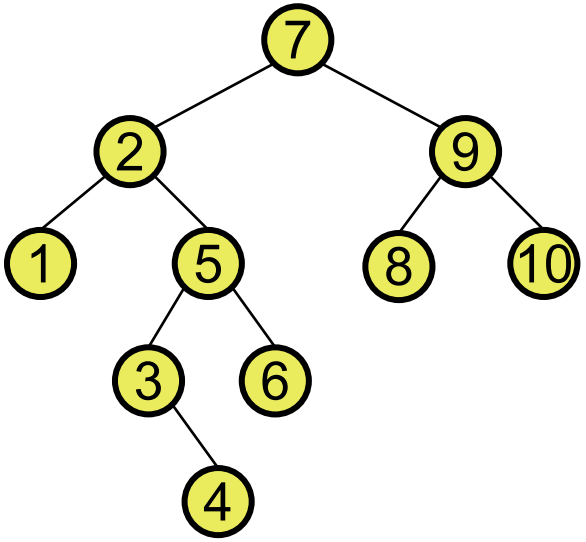
要素の削除

- 節点 zの削除
  - (i) zが子を持たない→zを削除.
  - (ii) zが1個の子を持つ→zを抜く.
  - (iii) zが2個の子を持つ→zの次の要素(右部分木の左端の接点y)と置き換える.



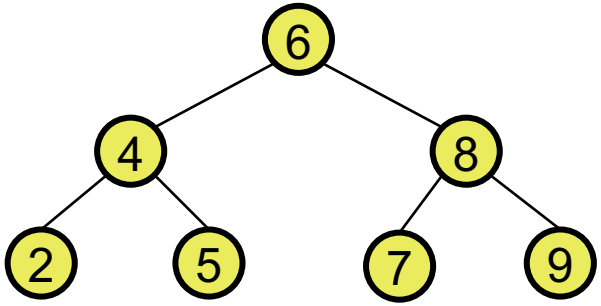
例4 (節点の削除)

- 次の2分探索木から要素zを削除した時，どの要素をzの位置に移動すればよいか？
  - 1. key[z] = 5
  - 2. key[z] = 7
  - 3. key[z] = 2

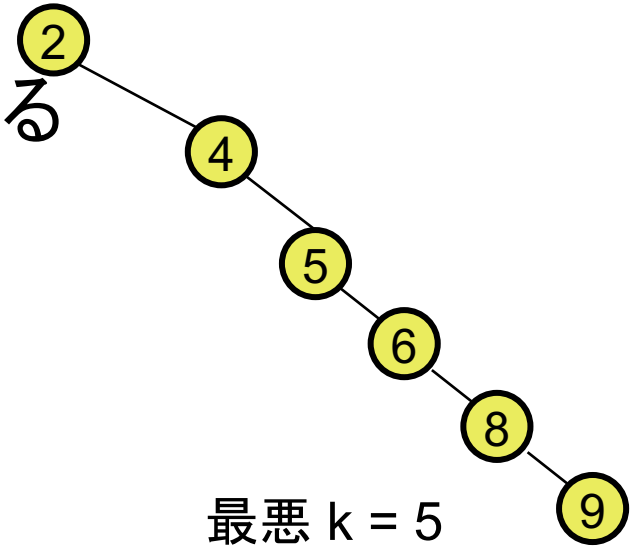


検索の速さ

- 速さ(処理時間)
  - 比較の回数で見積もる
  - 検索: 木の最大の高さkの回数に比例.
  - 追加, 削除も同様.
  - 追加順序で木の形が変わる



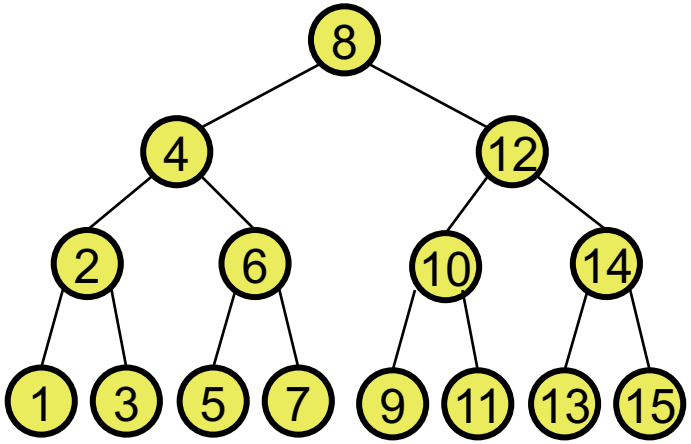
最良 k = 2



最悪 k = 5

木と高さ(深さ)の関係

- 完全2分木
  - 全てのn個の節点が2つの子を持つ.(葉の高さが等しい)
  - 全節点数 $n = 2^{k+1} - 1$ (公比2の等比級数)
  - 検索性能: $k = \log(n+1)-1 = O(\log(n))$



k=0 2^0

k=1 2^1

k=2 2^2

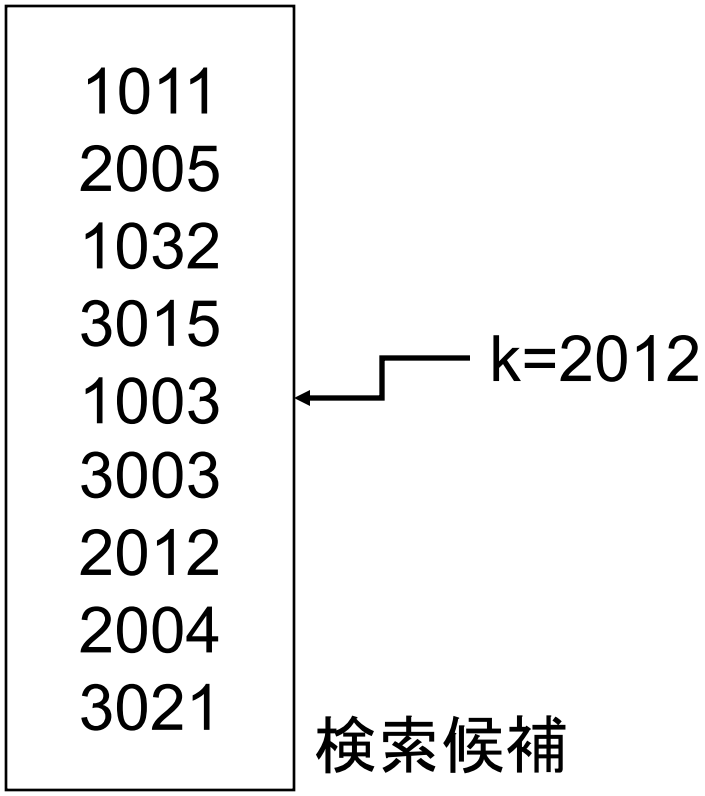
k=3 2^3

例5（木の検索速度）

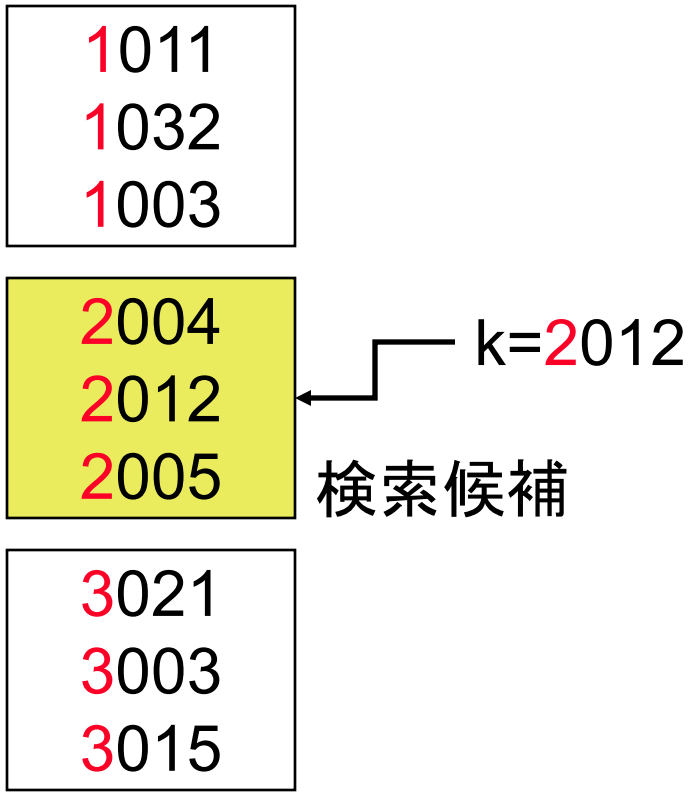
- 節点数15の完全2分木から検索をする．比較する節点の数は最大いくらか？

2. ハッシング(hashing) ★頻出

- 逐次検索



- ハッシング



ハッシュ

- hash
  - [動] 切り刻む，台無しにする
  - [名] 細切れ料理

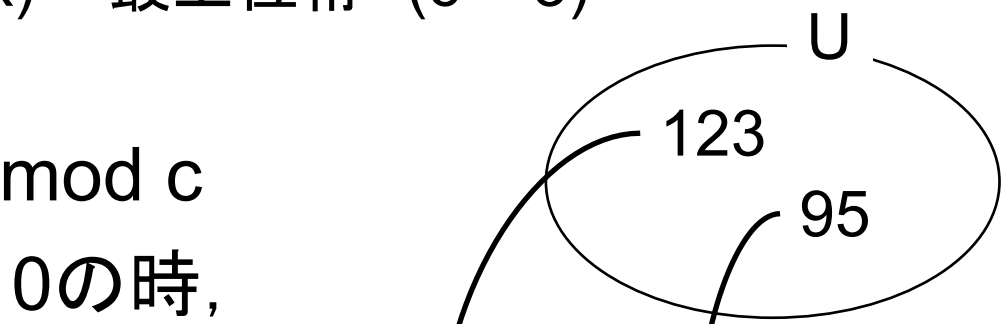
  - ハッシュポテト

- ハッシュテーブル
  - 検索技術（検索エンジン）
  - キーの特徴量（チェックサム，先頭文字）

- (参考) 暗号学的 ハッシュ関数
  - MD5, SHA1, SHA256

ハッシュ関数

- ハッシュ関数
  - 写像 h: U→{0,1,.., c-1}
  - キーxからハッシュ値h(x)を与える関数h
    - » 例）h(x) = 最上位桁　(c = 3)
  - 除算法
    h(x) = x mod c
  - 例) c = 10の時,
    h(123) = 3,
    h(95) = 5

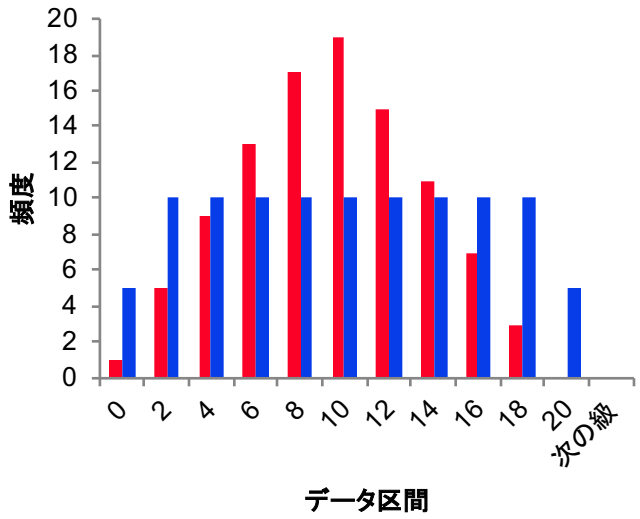


ハッシュ表

0	1	2	3	4	5	6	7	8	9
			123		95				

理想的なハッシュ関数

- 0から99の定義域のキーを0から19までの値域のハッシュ表に登録する．どちらのハッシュ関数がよいか？
  - 1. h1(x) = (x mod 10) + (x/10 mod 10)
    = 1桁目+2桁目
  - 2. h2(x) = x mod 20
  - h1は　　分布
    h2は　　分布



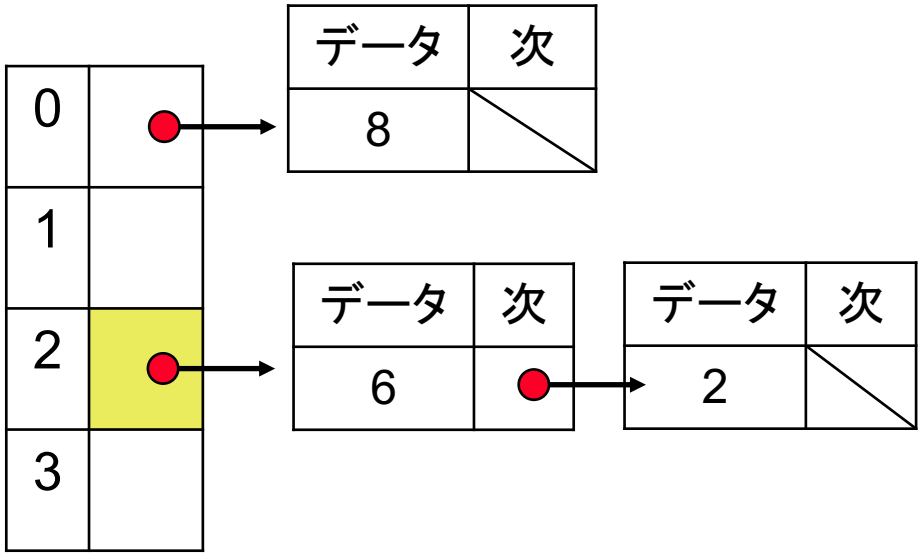
衝突とその解決

- 衝突 (collision)
  - キーが同一のハッシュ値となること. h(x)=h(y)となるx ≠ y
  - h(x) = x mod 4
  - h(8) = 0
  - h(6) = 2
  - h(10) = 2

ハッシュ値	値
0	8
1	
2	6と2
3	

衝突の解決方法

- チェーン法 (chaining)
  - ハッシュ値をリスト構造に登録.



- オープンアドレス
  - 再ハッシュの計算

1. i = 0 (衝突回数)
2. j = h(x, i)
3. If T[j] = NIL 登録する. return
4. Else ++i
5. goto 2

例6

- キー100をハッシュ表に登録する．ハッシュ関数は8進数に基数変換して最下位桁とするとき，登録されるアドレスを求めよ．

例7

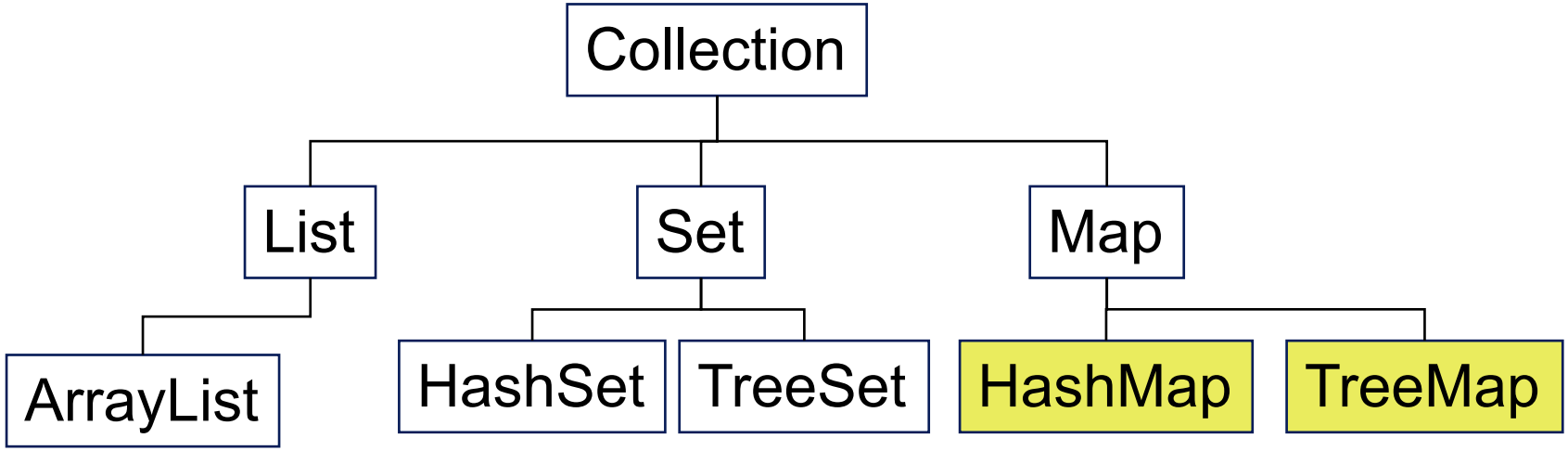
- キーのハッシュ関数 h(x) = x mod 17とする．キー1から100の中に，ハッシュ値h(x) = 3がで衝突するxはいくつ存在するか？

ハッシングの速さ

- 仮定
  - ハッシュ表が十分に大きい
- O(1) の計算量

3. 2分木とハッシュ表の実現

- 概要
  - オブジェクトの集まり(配列，集合，辞書)の操作を行う.
  - Collections Framework



Mapクラス

ソース

- MapDic.pde

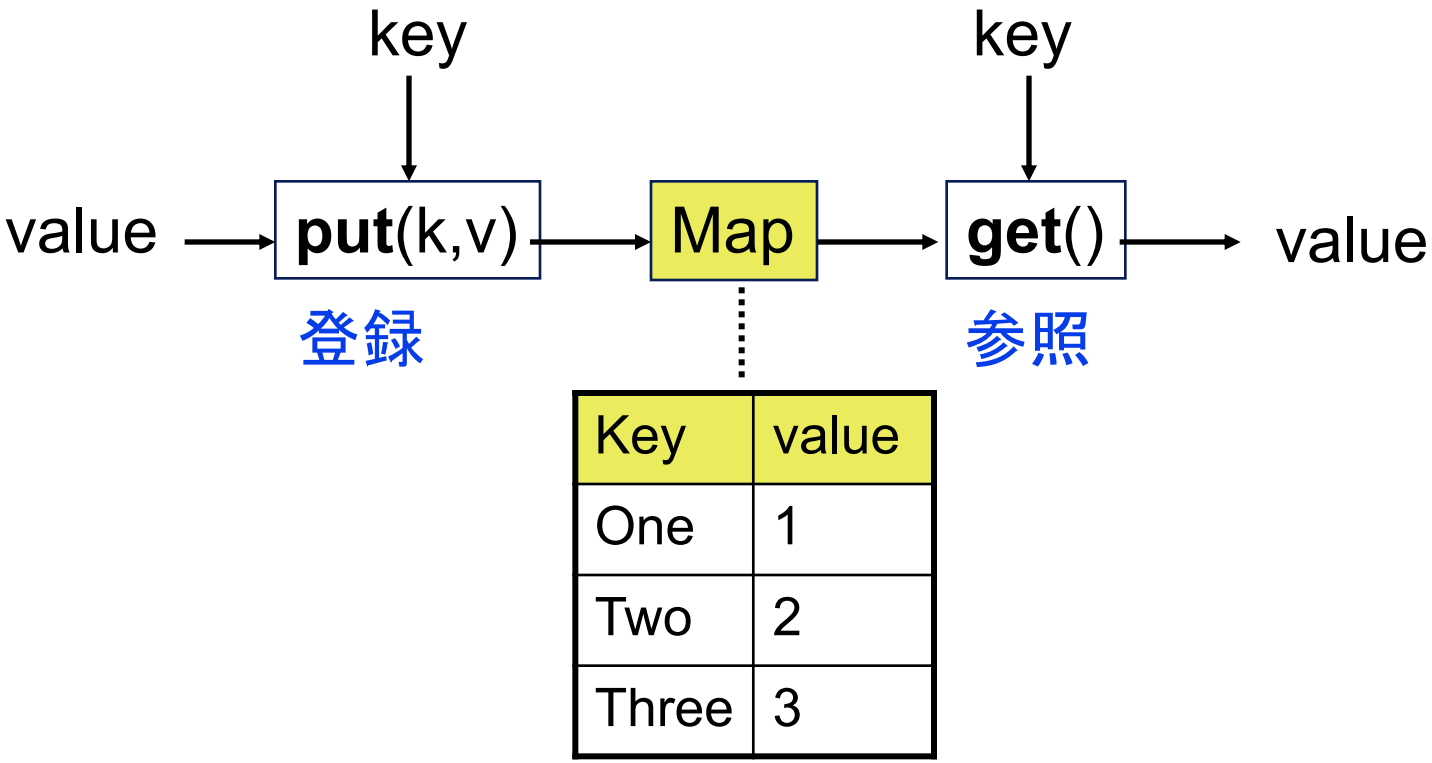
```
1. HashMap<String, Integer> dic = new HashMap<String, Integer>();
2. dic.put("one", 1);
3. dic.put("two", 2);
4. dic.put("three", 3);

5. println(dic);
6. String s[] = {"two", "one", "two", "three"};
7. for(int i = 0; i < s.length; ++i){
8.   print(dic.get(s[i]));
9. }
```

```
{two=2, one=1, three=3}
two one two three =
2, 1, 2, 3,
```

Mapの仕組み

- Key, valueの関係



Key	value
One	1
Two	2
Three	3

宿題

- 練習問題 p.102
  - 1. ハッシュ
  - 3. 2分木
  - 5. ハッシュ

まとめ

- 木構造は，（　　）と枝から成るグラフ構造であり，（　　）を頂点とし末端を葉とする．
- 2分探索木は，（　　）の大きさに応じて木構造に登録する．検索と（　　）と削除を動的に行う．n個のキーを(　　)回以内に検索する．
- ハッシングは（　　）分布をするハッシュ関数でキーからハッシュ表のアドレスを決める．ハッシュ値が一致することを（　　）といい，（　　）構造を用いたチェイニングなどの解決方法がある．