アルゴリズムとデータ構造

第11回　ハッシュテーブル

第11回のキーワード

アルゴリズム関係

- ハッシュテーブル / ハッシュ表（hash table）
- ハッシュ関数
- ハッシュ値（hash value）/ ハッシュコード（hash code）
- バケット（bucket）
- 衝突（collision）
- チェーン法
- オープンアドレス法
- $O(1)$

Java関係

- 文字コード（Unicode）
- str.charAt(i)
- hashCode
- %演算子の結果の範囲

確認問題

- 以下は，配列を用いて1桁の整数を英単語に“翻訳”するプログラムである。
- 空欄を埋めて，配列が「整数とデータの対応表」に使えることを理解せよ。

まず，PCに入力しないで
頭で考えて解答すること

```
import java.util.Scanner;

public class Main {
    final static String[] ewords = {
        "zero", "one", "two", "three", "four",
        "five", "six", "seven", "eight", "nine" };

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("number? ");
        int n = sc.nextInt();
        if (0 <= n && n <= 9)
            System.out.println(ewords[    ]);
    }
}
```

確認問題

- 以下（左）は，任意の文字列を整数に変換する「ハッシュ関数」の一例である。
- PCを使わずに実行過程をトレースし，hash("one") の値を手計算で求めよ。
- 計算に必要な文字コードは，'o'=111, 'n'=110, 'e'=101 である。
- また，右の表のセルC2を埋めて，同じ計算をするExcelシートを完成させよ。

```
public static int hash(String key) {
    final int X = 37, HASHSIZE = 31;
    int L = key.length();
    int h = 0;
    for (int i = 0; i < L; i++) {
        int c = key.charAt(i);
        h = h * 37 + c;
    }
    return Math.abs(h % 31);
}
```

プログラムを実行しないで
頭で考えて解答すること
（計算をPCでするのは可）

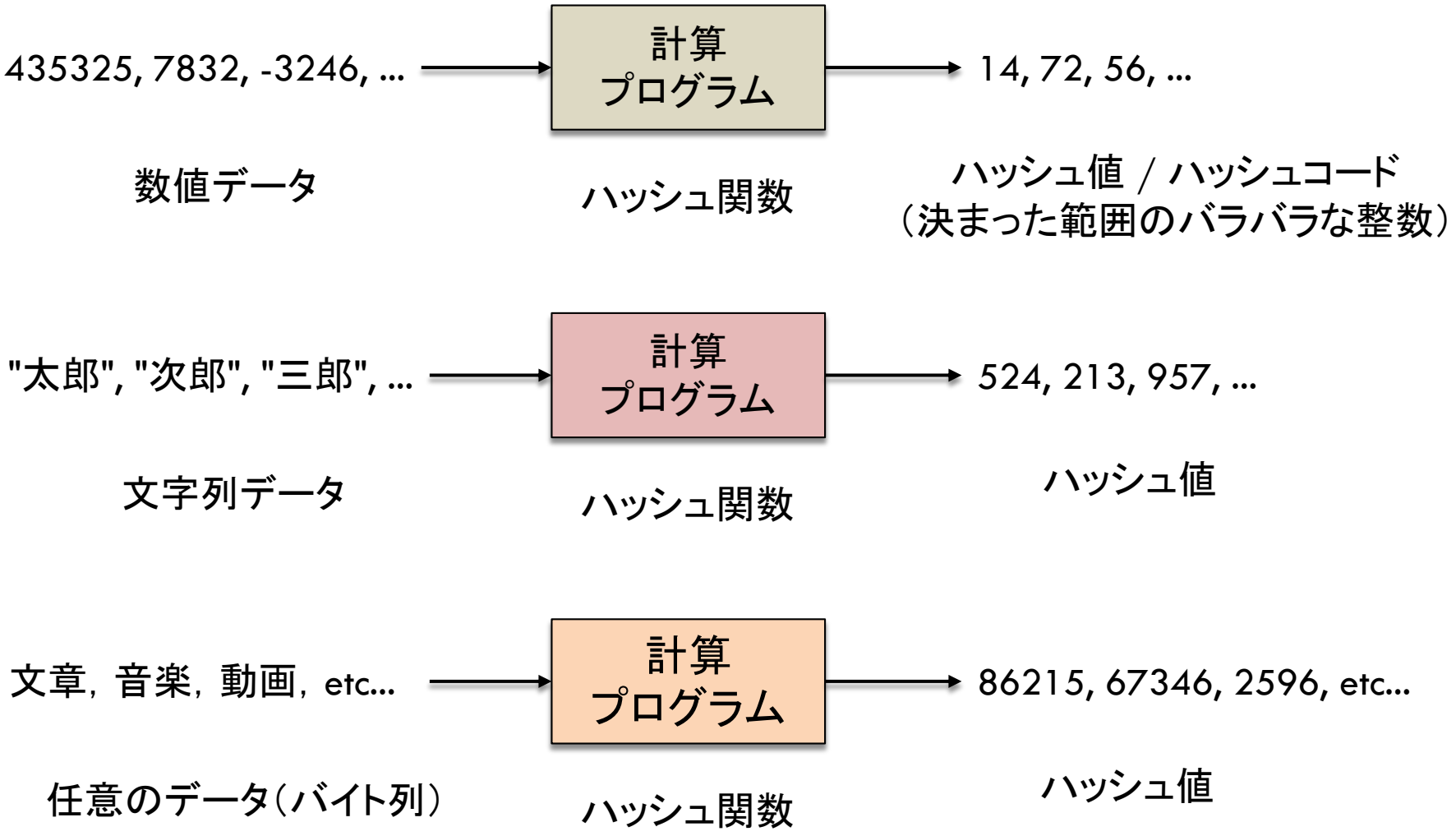
	A	B	C	D
1			0	
2	o	=UNICODE(A2)		=ABS(MOD(C2,31))
3	n	↓	↓	↓
4	e	↓	↓	↓

ハッシュテーブル

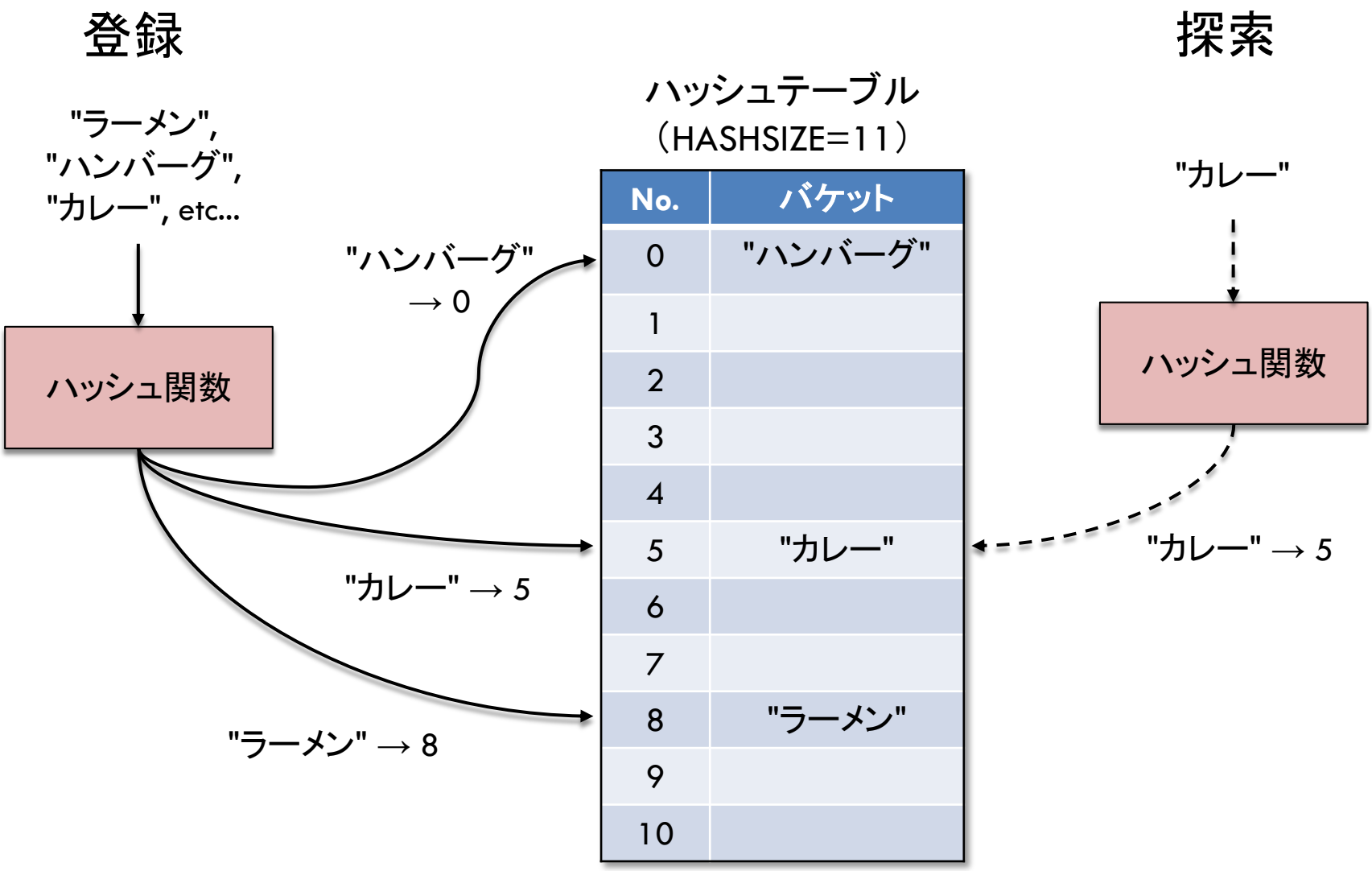
- ハッシュテーブル（ハッシュ表）
  - データ自体から，それを格納・探索する場所（配列の中の位置）を即時に計算で決める探索アルゴリズム
  - データを「ハッシュ関数」によって整数 h に変換し，配列の要素 a[h] をデータの格納場所とする
  - 「衝突」がなければ，データ数によらず極めて高速（$O(1)$）
  - 配列の中身を順に埋めるのではなくて，スカスカに使う

- ハッシュ関数
  - データから指定範囲のバラバラな整数を計算する関数
  - 衝突とは：異なるデータから計算したハッシュ値が偶然に一致すること → 格納場所が重なってしまうので困る…

ハッシュ関数の概念


435325, 7832, -3246, ...
計算
プログラム
14, 72, 56, ...
数値データ
ハッシュ関数
ハッシュ値 / ハッシュコード
（決まった範囲のバラバラな整数）
"太郎", "次郎", "三郎", ...
計算
プログラム
524, 213, 957, ...
文字列データ
ハッシュ関数
ハッシュ値
文章，音楽，動画，etc...
計算
プログラム
86215, 67346, 2596, etc...
任意のデータ（バイト列）
ハッシュ関数
ハッシュ値

ハッシュテーブル


登録
探索
"ラーメン",
"ハンバーグ",
"カレー", etc...
ハッシュ関数
"ハンバーグ"
→ 0
"カレー" → 5
"ラーメン" → 8
ハッシュテーブル
（HASHSIZE=11）
No.
バケット
0
"ハンバーグ"
1
2
3
4
5
"カレー"
6
7
8
"ラーメン"
9
10
"カレー"
ハッシュ関数
"カレー" → 5

確認問題

- 演習課題の2,4
  - 実際にプログラムを実行して解答する
  - 問題文のソースコードだけを抜き出したものが，テキストファイルでアップロードされているので利用してよい

ハッシュ関数の作り方

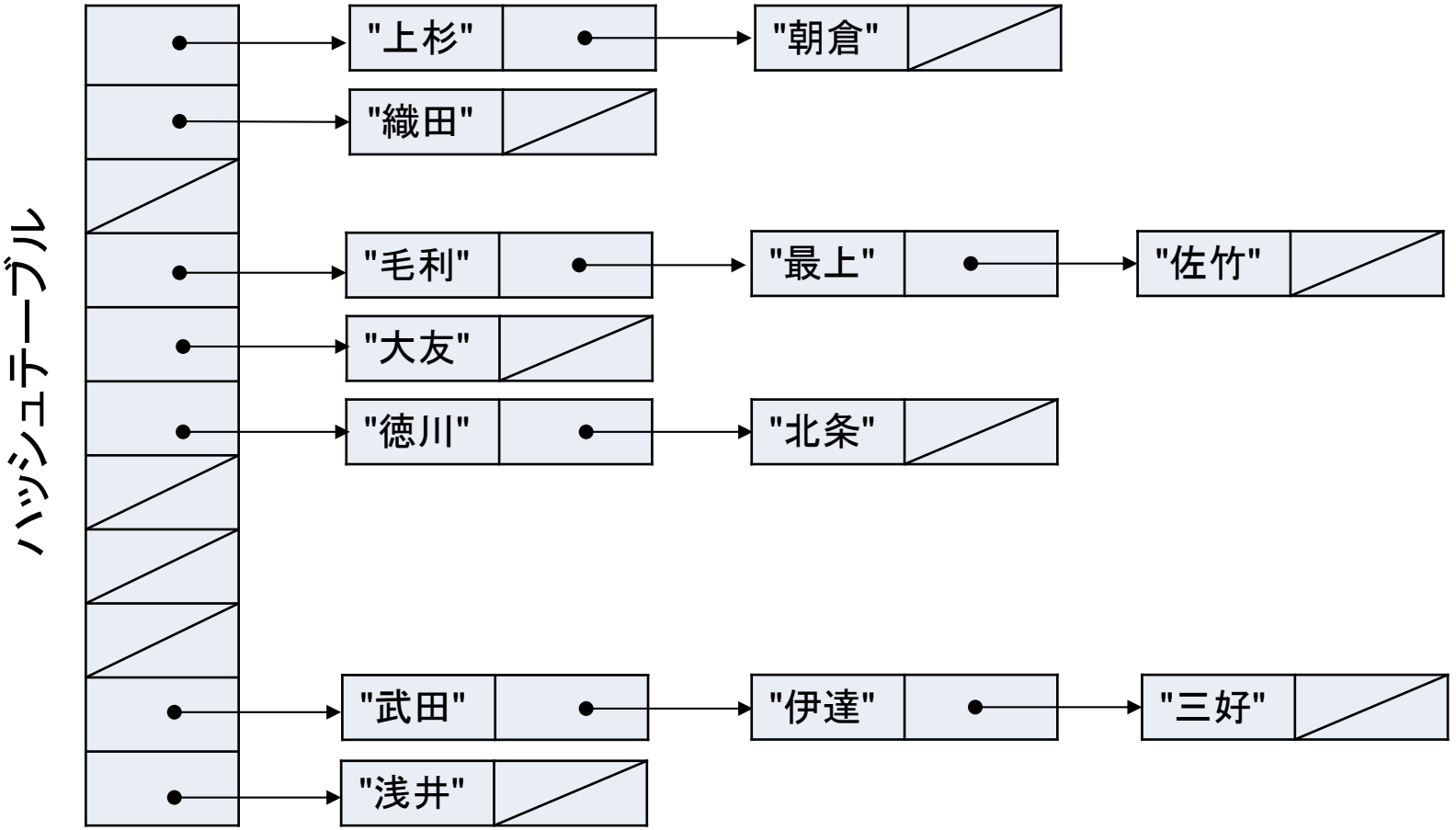
- よいハッシュ関数の条件
  - データが少しでも異なれば，ハッシュ値も異なる
  - ハッシュ値が異なれば，データは必ず異なる
  - ハッシュ値が分散し，バケットを全体的にまんべんなく使う
- ダメなハッシュ関数の例
  - 先頭文字だけを使う ⇒ 残りの文字が異なっても値が衝突
  - ハッシュ値が必ず偶数になる ⇒ バケットの半数が無駄
- ハッシュ関数の作り方
  - データの全体（または十分な長さ）を使って算出する
  - 値を分散させるには，素数による乗除算を繰り返すとよい

衝突への対処方法

- 完全ハッシュ
  - 全てのデータが分かっている場合に，ハッシュ関数の計算アルゴリズムを，必ず衝突がないように設計しておく
- オープンアドレス法
  - もし衝突が起きたら，別のアルゴリズムで計算しなおした要素（単純な方法の場合，単に次の要素）に格納する
- チェーン法
  - ハッシュテーブルの各要素（バケット）を連結リストにする
  - 要素の例） class Node { String data; Node next; }
  - 配列の例） Node[] hashtable = new Node[HASHSIZE];

チェーン法

- 各バケットを連結リストにして複数の値を格納する



課題の解答方法について

- クラス名（Nodeなど）は、変更しないでください
- IntelliJ IDEAでは、プロジェクトの中に「モジュール」を作ることができます