## アルゴリズムとデータ構造⑦

~探索問題(ハッシュ)~

鹿島久嗣

DEPARTMENT OF INTELLIGENCE SCIENCE
AND TECHNOLOGY

#### 探索問題:

#### データ集合から所望の要素を見つける

- ■探索問題は、データの集合から所望のデータを見つけてくる
  - ―データは「キー」と「(データの)内容」からなる
  - ―与えられたキーに一致するキーをもったデータを見つける
- ■2分探索木やハッシュ等によって実現可能

#### ハッシュ表:

- 0(1)で探索するためのデータ構造
- ■データ集合から、あるキーをもつデータを0(1)で発見する
- ■単純な実現:
  - キーに対して自然数を割り当てる(1~N)
    - キーがアルファベット6文字なら  $N = 26^6 \simeq 3 \times 10^8$
  - サイズの配列Nを準備する
  - -キーを自然数に変換して、配列のその位置に格納する
- ■問題点:
  - -長さNの配列を使うと大きすぎる
  - -M個のデータを格納するのに、M << Nなのでムダが多い

# ハッシュ関数: ハッシュ表を省スペースで実現する

- ■ハッシュ関数 h(x): {1,2,...,N} → {1,2,...,B}
- キー x を持つデータをh(x)の位置に格納する
  - -異なるxが同じ位置に格納されうるので、衝突したら (例えば)次の場所に格納
- h(x)のデザインは色々考えられるが、なるべく均等に格納されるものがよい
  - $-例: x = a_1 a_2 \dots a_6$  (アルファベット6文字) に対する ハッシュ関数  $h(x) = \sum_{i=1,2,\dots,6} c(a_i) \bmod B: c$ は文字 コード

#### ハッシュの衝突: 内部ハッシュと外部ハッシュによって衝突を回避

- ■異なるキーxとyに対してh(x) = h(y)となることがある
- ■衝突の回避法:内部ハッシュと外部ハッシュ
  - -外部ハッシュ:衝突回避のためにリストを格納
  - -内部ハッシュ:衝突時に別のハッシュ関数を用いる

#### 外部ハッシュ: 衝突回避のためにリストを格納する

- ■配列にデータを直接格納せず、h(x)の位置にリスト(へのポインタ)を格納
  - -M>B でもよい
- 計算量:ハッシュが一様ならば平均的に 0(1)
  - -ハッシュ関数を用いて配列にアクセスするところまでO(1)
  - -そこから先の計算量はリストの長さℓに依存
    - •ハッシュが一様ならば $\ell \approx \frac{M}{B}$ ,これを定数とみなせば平均的にO(1)

#### 内部ハッシュ: 衝突したときのために複数のハッシュ関数を用意

- ■配列にデータを直接格納する
  - M ≤ B である必要がある
- ■ハッシュ関数の列 $h_0, h_1, h_2, ...$ を用意して、 $h_i$ が衝突したら、次の $h_{i+1}$ を調べる
  - -例: $h_i = h(x) + i \mod B$

### 内部ハッシュの計算手間: 挿入は平均的に0(1)

- ■新しい要素を挿入する手間:空きがみつかるまでの期待ステップ数は、現在M個のデータがランダムに入っていて、かつハッシュもランダムだとすると、 $\frac{B}{B-M} = \frac{1}{1-\alpha}$ 
  - $\alpha = \frac{M}{B}$  (占有率)
- ハッシュ表を最初からつくる (M個の要素を登録する) と

$$-\sum_{m=0}^{M-1} \frac{B}{B-m} \approx \int_{0}^{M-1} \frac{B}{B-m} dm = B \log \frac{B}{B-M+1}$$

$$-$$
 要素あたり平均  $\frac{B}{M}\log\frac{B}{B-M+1}\approx-\frac{1}{\alpha}\log(1-\alpha)$ 

#### 内部ハッシュの計算手間: 探索と削除

- ■探索・削除の手間
  - -xが表にないならば挿入と同じ
  - -xが表にあるならば作成の一要素あたり手間と同じ
    - xを含むM個の要素をもつハッシュ表をつくることを考える
    - xがm番目に挿入された場合、それは $\frac{B}{B-m}$ ステップで見つかる位置に置かれる
- •占有率 $\alpha = \frac{1}{2}$ にしておけば  $-\frac{1}{\alpha}\log(1-\alpha) \approx 1.39$