計算機ソフトウェア
第四回

電気電子工学科
黒橋禎夫

探索(search)

- 探索(search)、検索(retrieval)

  データの中から要求(request, query)に合うものを探してくる操作

- 二分探索(binary search)

  人間が辞書を引く場合と同じ方法

  真ん中、真ん中・・・とみていく　O(log n)

  予めソートして二進木にしておくことが必要

ハッシュ法(hashing)

- データの値（レコードのフィールド）そのものをキーにして格納しておけば、その番号を探しにいくだけ

- キーの範囲は有限なので工夫が必要

  ハッシュ関数：大きな範囲の値を小さな範囲のキーへマップする

  衝突（collision）へどう対処するかが問題

ハッシュ法

- 長所

  とにかく速い。計算時間 O(1)

  「1回」という意味ではない。「n」と関係ない有限回の計算で取ってこれるということ

- 短所

  テーブルのサイズに余裕が必要

  経験的には「容量 ＝ データ個数 × 2」

Collision対策(1) チェイン法

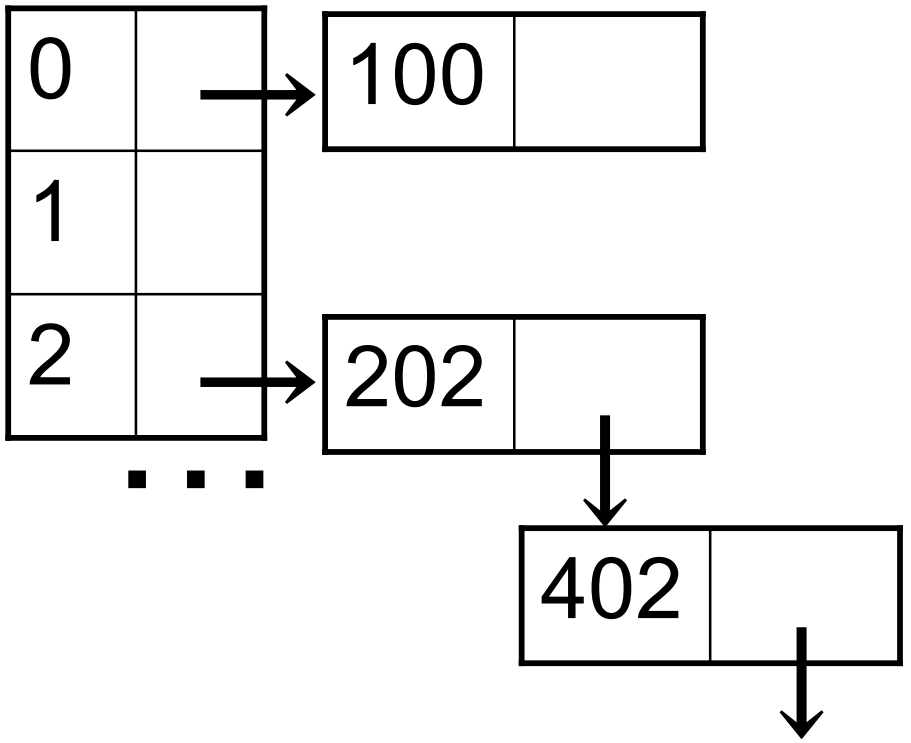
- 線形リストの先頭アドレスをテーブルに格納
- 同じキーを持つ値はリストの後ろへ繋ぐ

Ex.　値の範囲：0～999

ハッシュ関数：

「100で割った余り」

キーの範囲：0～99



Collision対策(2) 開番地法

- セルが埋まっているときは他のセルへ廻す

Ex.　ハッシュ関数h(x)でk回衝突した場合

- 線形走査法

  $h_k(x) = h(x) + k$

- 平方走査法

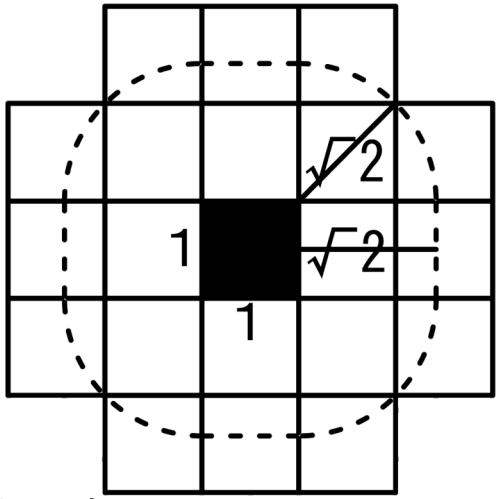
  $h_k(x) = h(x) + k^2$

- 二重ハッシュ法（別のハッシュ関数を用意）

  $h_k(x) + k \cdot g(x)$

バケット法

- O(1)アルゴリズム
- 平面上の正方形中に含まれるn個の点の中から与えられた座標に最も近い点を探す
- マス目（バケット）に切る
  およそ1マスに1点が入る按排
- ２１マス調べればよい
  nの大きさと無関係に平均O(1)
  （指定座標と同じマスに点がある場合）



バケットソート bucket sort

- ソートにバケット法を適用する

- 桁数が限定できるならばO(n)のアルゴリズム
  実際にO(n･logn)より速いとは限らない
  ∵ 実計算時間 ＝ n･a　（aは桁数できまる定数）
  のとき、a と logn ･ b のどちらが大きいかによる