探索　（search）

データ列の中から，特定のキーを持つデータを探す.

線形探索
　先頭から順番に，キーを調べる.

２分探索
　キー値に順序があり，順序に従い要素が並んでいる.
　列の中央のキーを調べ，その値により，列の前半，後半
　を探索対象とする.

計算量：　要素数ｎ
　線形探索：　Ｏ（ｎ）
　２分探索：　Ｏ（ｌｏｇ_2　ｎ）

ハッシュ　（hash）

連想リスト：

キーと，キーに対応つけたデータの組を要素とした，
連結リスト.
リストをたどり，特定のキーに対する値を探索する
ことができた.
線形探索には，Ｏ（ｎ）かかった.

ハッシュ法：

効率的に，キーを探すための，１つの方法

ハッシュ法の考え方：

相異なる８文字のアルファベット名のキー.

２６＾８＝　２０８，８２７，０６４，５７６

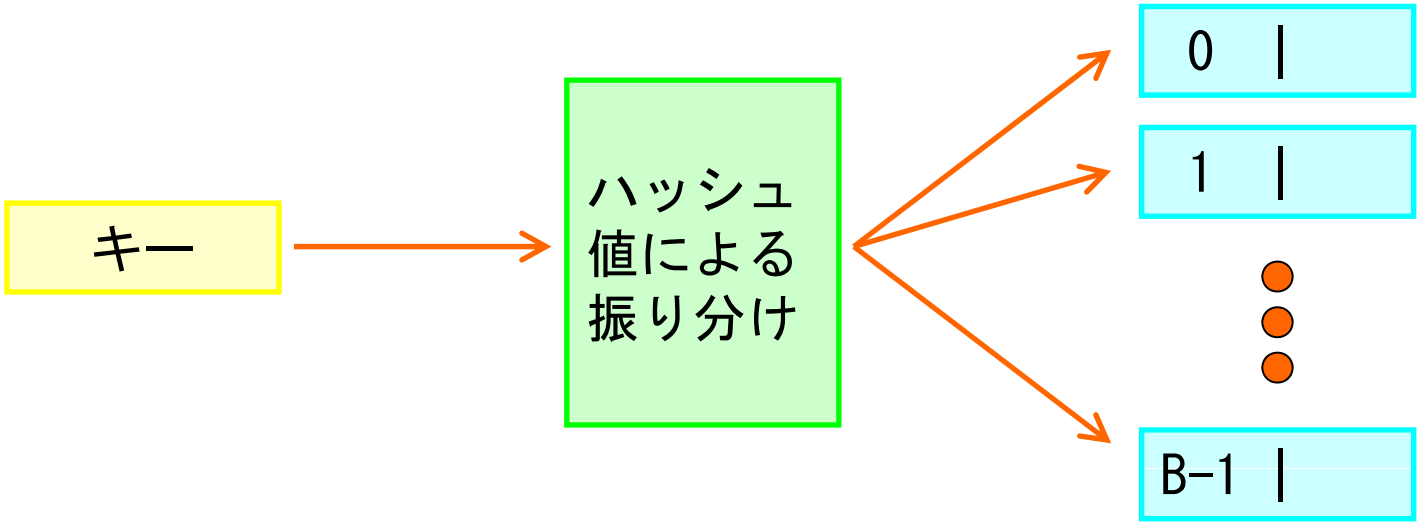
＝　２００Ｇ（ギガ）種類

実用的には，データ数　ｎ　＜＜　２．０×１０＾１１

ハッシュ表：　Ｂ個のバケット

（bucket：バケツ，入れ物）を持つ表

ハッシュ値：　０～（Ｂ－１）　の値



ハッシュ関数　ｈ：

アルファベット名からハッシュ値を得る関数.

ｈ：｛アルファベット名｝→｛０，・・・，Ｂ－１｝

ハッシュ関数の例

アルファベットに数値をつける（数値化）：

ａ→０，ｂ→１，・・・，ｚ→２５.

８文字のアルファベットの数値の和　ｓｕｍ　を求める（数値化により計算が出来る）.

ハッシュ値　＝　ｓｕｍ　％　Ｂ

（０　～　Ｂ－１　の値）

抽象データ型としてのハッシュの操作：

ハッシュの初期化

ハッシュ値に応じた，データの追加，削除，探索

同義語（synonym），衝突（collision）：

　異なるキーに対して，ハッシュ値が同じになること.

同義語，衝突の解消：

　(1) バケットに，連結リストをつける.

　　名称：　オープンハッシュ法，チェイン法，

　　　　外部ハッシュ法

　(2) ハッシュ表の空きバケツが見つかるまで，

　　ハッシュ値を再計算する（rehash）.

　　名称：　クローズドハッシュ法，オープンアドレス法，

　　　　内部ハッシュ法

(1) の方法

各バケットに，平均 ｎ／Ｂ 個のデータが，連結リストとしてつながる.

計算量： ハッシュ値の計算と，データ追加・削除・探索のためのリストの走査

バケットの数が多ければ，走査する計算量は小さい.

（例）Ｂ＝８

key0 → 1

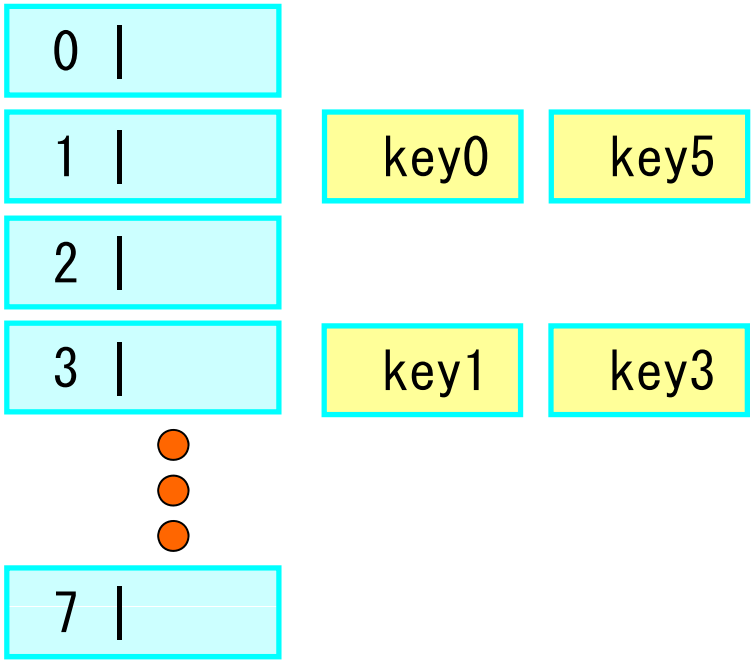
key1 → 3

key2 → 5

key3 → 3

key4 → 4

key5 → 1



(2) の方法

(a) １，２，･･･，ｉ回目の再ハッシュ関数の例（1）

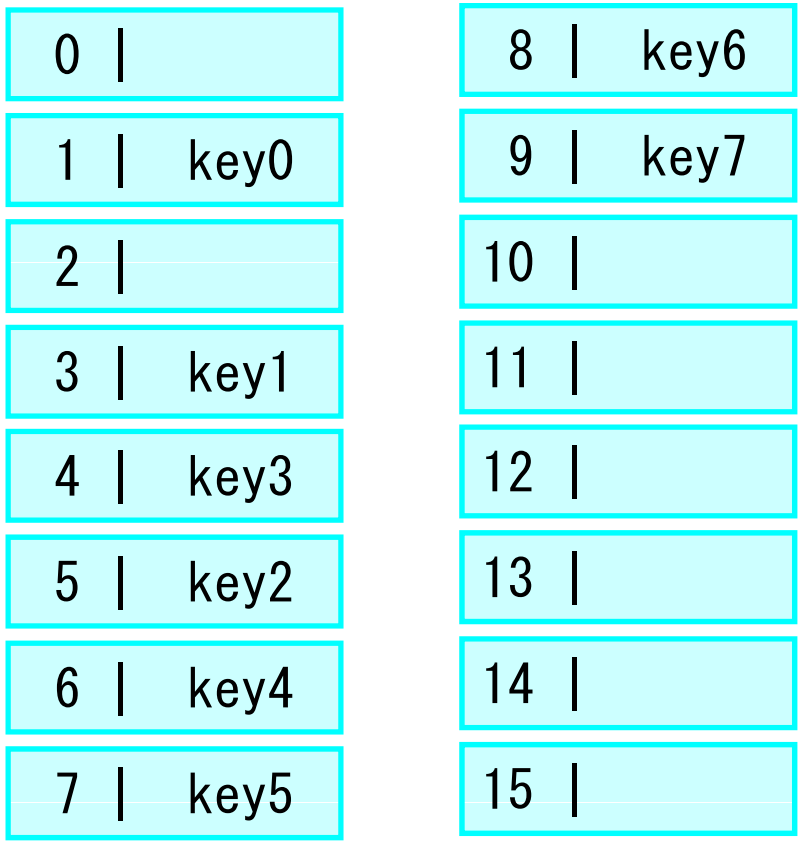
ｈ（ｘ）＝ ｘ ％ Ｂ

ｈｉ（ｘ）＝（ｈ（ｘ）＋ｉ）％ Ｂ

Ｂ＝１６

key0 → 1
key1 → 3
key2 → 5
key3 → 3
key4 → 5
key5 → 3
key6 → 5
key7 → 5

塊ができてしまった.



(2) の方法

探索：

（i）　１回目ハッシュ値が同じキー（同義語）の塊で，目的のキーを捜す．

（ii）　同じキーが現れるまで，再ハッシュ値の評価を繰り返す．

(b)　i 回目の再ハッシュ関数の例（2）

h（x）＝ x % B

hi（x）＝（h（x）＋ki）% B

k1, k2, ・・・は, １から（B－１）の値の, ランダムな並び.

同義語ごとに, キーと, 要素追加に成功したランダム値を保持　→　同義語集合内の探索で利用.

計算量：　２n ≦ B であれば, 再ハッシュは, せいぜい１回　（経験則）