データ構造と アルゴリズム I

第9回

11. ハッシュ表

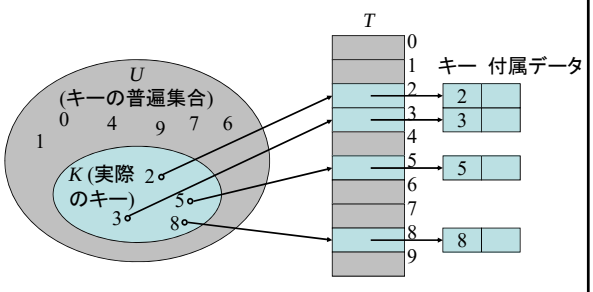
- 辞書操作 (INSERT, DELETE, SEARCH) を効率よく実現するデータ構造
- 応用:C言語のコンパイラでの記号表の管理
  - キー:変数名などの文字列
- ハッシュ表は実際的な場面では極めて速い
  - 妥当な仮定の下で, SEARCHの時間の期待値は O(1)
  - 最悪の場合 $\Theta(n)$

キーの自然数としての解釈

- キーは自然数として与えられることを仮定
- 文字列 (e.g. 変数名) を どのように自然数に変換するか?
  - 文字列 “pt”
  - ASCIIコード (0から128までの自然数で文字を表現) ではp=112, t=116
  - 文字列 “pt” を 10進数の組(112, 116)で表現
  - これを基数128の整数と思えば, “pt” = 112*128+116=14452

11.1 直接アドレス表

- 出現する可能性のあるキーの全集合 (普遍集合, universal set) が大きくない場合にうまく働く
- キーが普遍集合 $U = \{0,1,\ldots, m-1\}$ から選択され, どの2つの要素も同じキーをもたないと仮定する
- 直接アドレス表 (direct-access table) T を用いて動的集合を表現する
- 配列 $T[0..m-1]$ の各要素が U のキーに対応
- $T[k]$ は, キー k を持つ要素を指す. そのような要素がなければ $T[k]$ = NIL
- $T[k]$ をスロット k と呼ぶ



直接アドレス表の欠点

- キーの普遍集合 U が大きい場合は非現実的
  - 表 T をメモリに格納できない
  - T に割り付けた領域のほとんどが無駄になる
- 通常は, 辞書に格納されているキーの集合 K は U に比べて十分に小さい (e.g.,任意の文字列の数>> 実際に使われる変数名の数)
- もっと工夫した方法が必要!

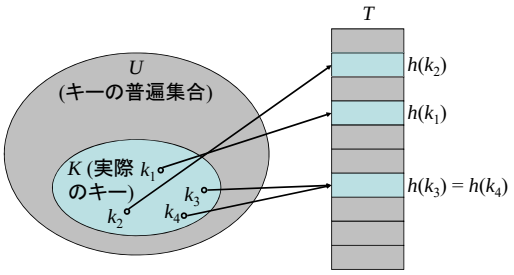
11.2 ハッシュ表

- 直接アドレス表では，キー k はスロット k に格納
- ハッシュ表 $T[0..m-1]$ では，スロット $h(k)$ に格納
- h: ハッシュ関数 (hash function)
  - $h: U \rightarrow \{0,1,\ldots,m-1\}$
- 必要な領域: $\Theta(|K|)$
  --- Kは辞書に格納されているキーの集合
- 要素の探索: O(1)　(平均時間)

343

ハッシュ関数の衝突

- 衝突 (collision): 2つのキーが同じスロットにハッシュされること



344

衝突の回避方法

- 別のハッシュ関数を用いる
  - $|U| > m$ なので完全に回避することは不可能
  - m+1個の要素を格納しようとすると，必ず衝突が生じる⇒鳩の巣原理
- チェイン法
- オープンアドレス法

345

鳩の巣原理

- 鳩がn+1羽，鳩の巣となる穴がnとすると，少なくとも一つの穴には二羽の鳩がいる
  - 当たり前のようだが，この原理を使って簡潔な証明が可能な事例が多数存在

346

鳩の巣原理の利用法(I)

命題：福岡市の住人中で，全く同じ本数の髪の毛を持つ人のペアが存在する

- 髪の毛の本数は15万本程度
- 100万本以上の髪の毛を持っている人間はいないと考えてよい
- 福岡市の人口は100万を超える.
- もし、髪の毛の本数ごとに鳩の巣を割り当て、巣に人を割り当てるなら，少なくとも同じ髪の毛の本数を持った2人の人間が必ず存在

347

鳩の巣原理の利用法(II)

命題：平面上に5つの異なる格子点を選び，それらを結んだ場合，これらの線の中点の少なくとも一つは格子点となる

- 格子点の座標を (x, y) とする. x, yが共に偶数の点をtype 1，共に奇数の点をtype 2，xが奇数でyが偶数の点をtype 3，xが偶数でyが奇数の点をtype 4と呼ぶ
- 同じタイプ同士を結ぶ線の中点は格子点
- 5つ点を選べば，少なくとも一組は同じタイプ

348

鳩の巣原理の利用法(III)

命題: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}から7個の相異なる数を選ぶ. そのうち適切な2個を選ぶと, 必ず合計を12にすることができることを示せ.

命題:一辺の長さが3cmの正方形中に, 10個の点を任意に選ぶ. そのうち適切な2個を選ぶと, 必ず2点間の距離を$\sqrt{2}$ 以下にできることを示せ.

349

チェイン法による衝突解決

- 同じスロットにハッシュされたすべての要素を連結リストに格納
- CHAINED_HASH_INSERT(T,x)
  - リスト $T[h(\text{key}(x))]$ の先頭に x を挿入, O(1) 時間
- CHAINED_HASH_SEARCH(T,k)
  - リスト $T[h(k)]$ の中からキー k を持つ要素を探索
- CHAINED_HASH_DELETE(T,x)
  - リスト $T[h(\text{key}(x))]$ から x を削除, 双方向リストを用いれば O(1) 時間

350

11.3 ハッシュ関数

- 良いハッシュの条件 = 単純一様ハッシュ
  - 各要素は m 個のスロットに同じ程度にハッシュされる
- 各キーは U から確率分布 P に従って独立に取り出されると仮定すると, 条件は

  と書ける $\sum_{k:h(k)=j} P(k) = \frac{1}{m} \qquad (j = 0,1,\ldots,m-1)$
- ただし, 一般に P は未知
- キーがランダムな実数 k ($0 \le k < 1$)で一様独立のとき, $h(k) = \lfloor km \rfloor$ は上の条件を満たす

351

ハッシュ関数の設計

- 現実的には, 必ずうまくいくとは証明できないが, 多くの場合にうまくいくことが経験的に分かっている方法 (ヒューリスティックな方法) を用いることが多い

望ましい性質:

- なるべく各ビットが まんべんなくハッシュ値に影響する
  - 似た文字列でも違う値になる ("pt", "pts"等)

352

11.3.1 除算法

- キー k を m で割った剰余をハッシュ値とする
  - $h(k) = k \bmod m$
- 利点: ハッシュ関数の計算が高速
- m は 2 のべき乗に近くない素数がいい
  - $m = 2^p$ のとき, $h(k)$ は k の最下位 p ビット
    - $m=128^2$ だと文字列の最後の2文字
  - $m = 2^p - 1$ で k が基数 2^p の文字列のとき, キーは文字列の各文字のコードの和になってしまう
    - 文字を並び替えてもハッシュ値は同じ

353

例

- n = 2000 個の文字列を格納する場合
- 占有率 (ハッシュ表が埋まっている割合) を 3 に近くするには, m = 701 にすればいい
- 701 は素数で, 2のべき乗には近くない
- $h(k) = k \bmod 701$ とすればいい
- このハッシュ関数が実際のデータでうまく働くことを確かめるべき

354

演習

- 整数キーの集合K={4, 8, 10, 22, 49, 72}を格納するハッシュ表を考える
- ハッシュ関数を h(x) = x mod 7 とする

問1：Kの要素を小さい順に挿入した後のハッシュ表を書け．衝突はチェイン法を用いて解消し，新しい要素はリストの先頭に挿入するものとする

問2: h'(x) = x mod m を用いて，Kに衝突が生じないようにしたい．そのようなmのうちで最小のものを求めよ．

演習の解答

K={4, 8, 10, 22, 49, 72}

問1：

0: (49)
1: (22, 8)
2: (72)
3: (10)
4: (4)
5:
6:

問2: 11

11.3.2 乗算法

- まず，キー k にある定数 A ($0 < A < 1$) を掛け，その小数部分 $kA - \lfloor kA \rfloor$ を取り出す
- 次に，その値に m を掛け，小数部分を切り捨てる
  $h(k) = \lfloor m(kA - \lfloor kA \rfloor) \rfloor$
- m の値はあまり重要ではない
  - 2のべき乗にすると計算が簡単
- $A \approx (\sqrt{5}-1)/2 = 0.6180339887\ldots$ が良いと言われる
- Aに十分な精度がないと，大きなkで0になってしまうことに注意

ハッシュ関数の応用 (I)

- パスワードの保存：
  - サーバにユーザのパスワードを直接保存すると，情報が漏洩した場合に深刻な問題となる
  - パスワードではなく，パスワードのハッシュ値を保存する
  - ハッシュ値が一致すれば正しいとみなす
  - 同じハッシュ値になる別のパスワードを見つけるのは（うまく設計されたハッシュ関数を用いた場合には）大変

ハッシュ関数の応用 (II)

- 電話でコイン投げ
  - 電話で話をしているAlice, Bobの二人の間で，コイン投げで勝敗を決めたい
  - Aliceがコインを投げて，Bobが予想をする場合：
    - Aliceがコイン投げの結果を先にBobに告げると，Bobが嘘を付くかも知れない
    - Bobが自身の予想を先にAliceに告げると，Aliceが嘘を付くかも知れない
- Aliceは適当な自然数を選ぶ (コイン投げに相当)
- Aliceは選んだ自然数のハッシュ値をBobに告げる
- BobはAliceが選んだ数が偶数か奇数かを予測して，Aliceに告げる
- Aliceは選んだ自然数の値をBobに告げる

ハッシュ関数の応用 (III)

証拠の保持（コミットメント）

- Aliceが，ある時点で，ある情報を持っていること (例えば，ある株が値上がりする／値下がりする等) の証拠をBobに対して残したい（情報そのものは，まだBobに対して公開したくない)
- Aliceは情報＋適当に選んだ乱数のハッシュ値を計算してBobに公開
- 後日，Aliceは情報と乱数を公開
- ハッシュ値が一致すれば，前の時点でAliceがこの情報を持っていたことが証明できる
  - Bobにとっては，ハッシュ値のみではAliceの持っている情報を推測することは困難 (乱数なしでは簡単)
  - Aliceにとっては，後出しで情報を変更することは困難（異なる情報に対して，同じハッシュ値を与える乱数を見つけることは困難）

11.4 オープンアドレス指定法

- オープンアドレス指定法 (open addressing) では，要素は連結リストではなくハッシュ表の中に格納される.
- ハッシュ表が埋まるとそれ以上挿入できない
  - 占有率は1以下
- 連結リストを用いないため，スペースが小さい
- ハッシュ関数を拡張して衝突を回避する
  - 引数: キーと探査番号
  - h: $U \times \{0,1,...,m-1\} \rightarrow \{0,1,...,m-1\}$
- 探査列 $\langle h(k,0), h(k,1),..., h(k,m-1)\rangle$ は $\langle 0,1,...,m-1\rangle$ の並び替えでなければならない

衝突回避の方法

理想的な場合:

- 一様ハッシュ (uniform hashing) :
  各キーに対する探査列として，{0,1,...,m−1} の m! 通りの順列のどれもが同程度に現れる

現実的には近似的な一様ハッシュを用いる

- 線形探査
- 2次関数探査
- ダブルハッシュ法

線形探査

- 通常のハッシュ関数 h': $U \rightarrow \{0,1,...,m-1\}$ に対し $h(k,i) = (h'(k)+i) \bmod m$ (i=0,1,...,m−1) を用いる
- 探査されるスロット: $T[h'(k)]$, $T[h'(k)+1]$,..., $T[m-1]$, $T[0]$, $T[1]$,..., $T[h'(k)-1]$
- 異なる探査列は m 通りしかない (開始位置で決定)
- 問題点: 主クラスタ化 (primary clustering) が起きる
- 直前の i 個のスロットが使用中である空きスロットが選択される確率は $(i+1)/m$ であるため，連続する使用中のスロットは常に大きくなる

演習

- 整数キーの集合K={4, 8, 10, 22, 49, 72}を格納するハッシュ表を考える
- ハッシュ関数を h(x) = x mod 7 とする

Kの要素を小さい順に挿入した後のハッシュ表を書け.
衝突は線形探査を用いて解消するものとする

演習の解答

K={4, 8, 10, 22, 49, 72}
0: (49)
1: (8)
2: (22)
3: (10)
4: (4)
5: (72)
6:

2次関数探査

- 2次関数探査 (quadratic probing) では $h(k,i) = (h'(k)+c_1 i+c_2 i^2) \bmod m$ (i=0,1,...,m−1) を用いる
- c_1, c_2, m は適切に選ぶ必要がある
- 異なる探査列は m 通りしかない (開始位置で決定)
- $h(k_1,0) = h(k_2,0)$ の場合は $h(k_1,i) = h(k_2,i)$ となってしまう (副クラスタ化, secondary clustering)

ダブルハッシュ法

- $h(k,i) = (h_1(k) + h_2(k) \times i) \bmod m$
- 探査列は，初期位置と次に探査する位置までの距離の両方が k に依存している
- $h_2(k)$ の値は m と互いに素である必要がある
  - $h_2(k)$ と m の最大公約数が d のとき，ハッシュ表の1/dしか検査しない
- 例1: m を2のべき乗，h_2 は常に奇数
- 例2: m を素数，h_2 は m 未満の非負整数
  $h_2(k)=1+(k \bmod m')$
  --- m' はmよりわずかに小さい数
- $\Theta(m^2)$ 個の探査列が利用できる

367

要素の削除

- 削除したい要素のあるスロットを NIL にすると，他の要素が検索できなくなる
  - NIL スロットが見つかると検索は終了するため
- 削除するときは NIL でなく特別な値 DELETED を格納する
- SEARCHではDELETEDが現れても探索を続ける
- INSERTではNILまたはDELETEDの場所に挿入
- 問題点: DELETEDで占有されてしまう
- 要素を削除する必要がある場合はチェイン法が好まれる

368