Lecture 7: Phân tích khấu hao

- Bài toán: ước lượng tổng thời gian chạy của một chuỗi các **n** thao tác lên một cấu trúc dữ liệu cho trước.
  - VD: Stack, Queue, Heap,…
- Khó có thể xác định được một tổng thời gian chạy chính xác của một chuỗi **n** thao tác, do đó người ta đi xác định cận trên của nó.

Phân tích khấu hao

- Gọi $T(n)$ là cận trên thời gian cần thiết để thực thi một chuỗi n thao tác lên một cấu trúc dữ liệu.
  - Ví dụ: thực thi một chuỗi PUSH, POP, MULTIPOP lên một stack.
- *Phân tích khấu hao* (amortized analysis):
  - Phân tích khấu hao là đi xác định phí tổn khấu hao cho mỗi thao tác trong chuỗi **n** các thao tác.
  - Yêu cầu là tổng phí tổn khấu hao phải là cận trên thời gian cần thiết để thực hiện chuỗi **n** thao tác
  - Thời gian trung bình để thực hiện một thao tác là: $T(n)/n$,

    được gọi là *phí tổn khấu hao* (Đây chỉ là một trong các định nghĩa của phí tổn khấu hao, các định nghĩa khác sẽ được trình bày trong các phần sau)

Các phương pháp xác định phí tổn khấu hao

- Ba phương pháp để xác định phí tổn khấu hao:
  - *Gộp chung* (the aggregate method)
  - *Kế toán* (the accounting method)
  - *Thế năng* (the potential method)
- Mỗi phương pháp đưa ra định nghĩa phí tổn khấu hao riêng và chứng minh phí tổn khấu hao tổng cộng là chặn trên của phí tổn thực sự tổng cộng.
- Chúng ta sẽ minh họa các phương pháp trên thông qua việc phân tích các cấu trúc dữ liệu sau:
  - stack
  - bộ đếm nhị phân dài k bits
  - bảng động.

Cấu trúc dữ liệu stack

- Các thao tác lên một stack S
  - PUSH(S, x)
  - POP(S)
  - MULTIPOP(S, k)

```
MULTIPOP(S, k)
1   while not STACK-EMPTY(S) and k ≠ 0
2         do POP(S)
3            k ← k – 1
```

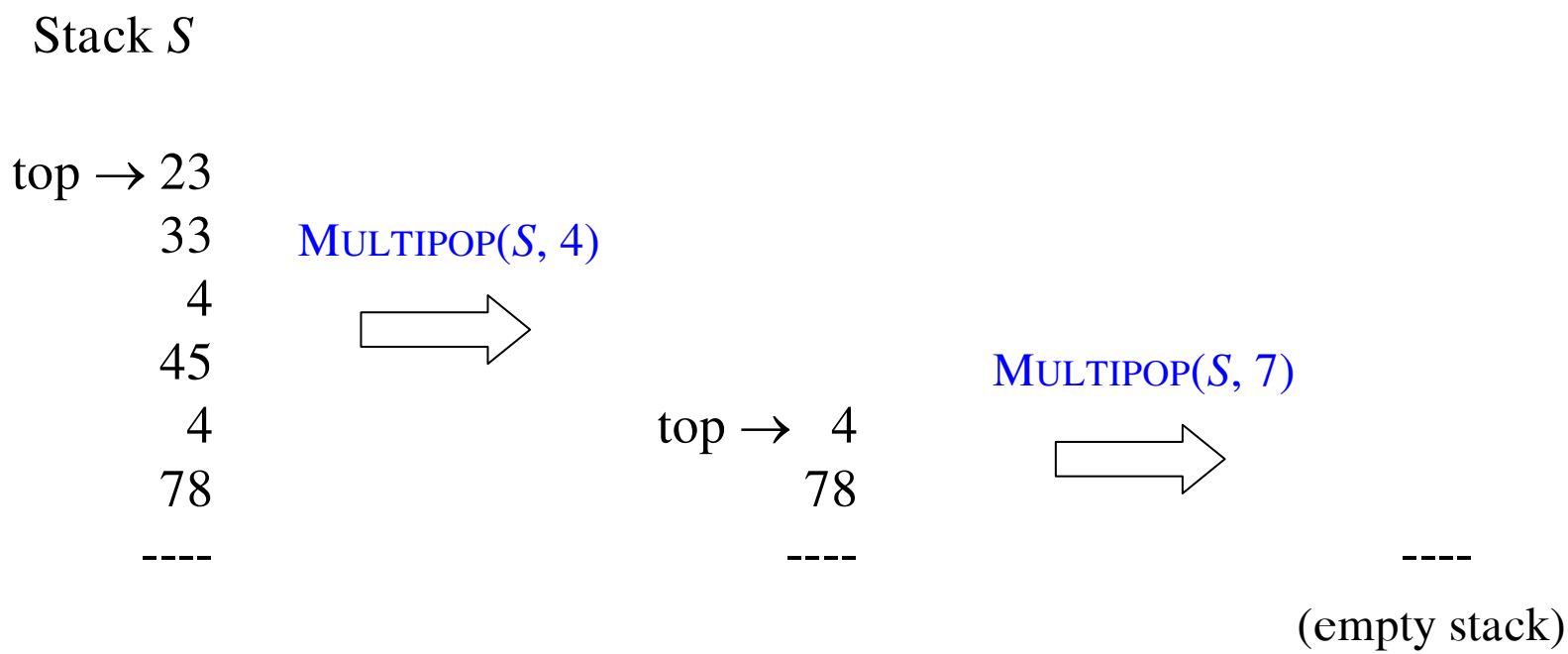
k4

Giới thiệu cấu trúc Satck

khmt-hung, 12/4/2006

Cấu trúc dữ liệu stack (tiếp)

- Minh họa thao tác MULTIPOP



Bộ đếm nhị phân dài k bit

- Bộ đếm nhị phân dài k-bit (k-bit binary counter)
  - là một mảng $A[0..k-1]$ của các bit 0/1
  - có độ dài, $length[A]$, là k.

k5

Giới thiệu cấu trúc bộ đếm nhị phân dài k bít

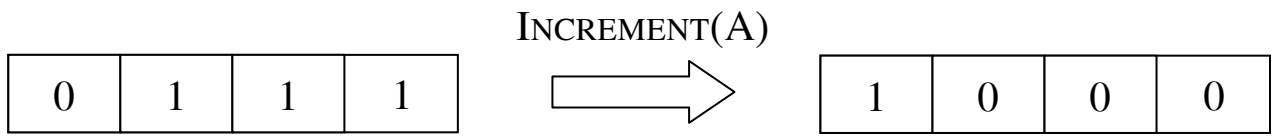
khmt-hung, 12/4/2006

Bộ đếm nhị phân dài k bit (tiếp)

- Dùng bộ đếm để lưu trữ một số nhị phân x:

  $x = A[k-1] \cdot 2^{k-1} + \ldots + A[0] \cdot 2^0$

  – INCREMENT: cộng 1 vào giá trị đang có trong bộ đếm (modulo 2^k)



  – Thủ tục INCREMENT

```
INCREMENT(A)
1   i ← 0
2   while i < length[A] and A[i] = 1
3         do A[i] ← 0
4            i ← i + 1
5   if i < length[A]
6      then A[i] ← 1
```

Thời gian chạy (phí tổn) của thủ tục INCREMENT tỉ lệ với số lần mà các bits bị đảo ngược.

Phân tích cấu trúc stack

- Bài toán: xác định thời gian chạy của một chuỗi n thao tác lên một stack (ban đầu stack là trống).

  Giải bằng phân tích "thô sơ":

  - Phí tổn trong trường hợp xấu nhất của MULTIPOP là $O(n)$.
  - Trong trường hợp xấu nhất thì cả n thao tác trên đều là MULTIPOP.
  - Do đó phí tổn của một chuỗi n thao tác là $O(n^2)$.

◦ Nhận xét: Chặn $O(n^2)$ tìm được là quá thô.

- Ta phải tìm chặn trên tốt hơn!
  - Dùng phân tích khấu hao.

Phương pháp gộp chung (Aggregate)

- Định nghĩa: Trong phương pháp *gộp chung* (*aggregate*) của phân tích khấu hao, chúng ta gộp thời gian mà một chuỗi n thao tác cần trong trường hợp xấu nhất (worst case) thành $T(n)$.
  - Như vậy trong phương pháp gộp chung ta đi tìm cận trên T(n), sau đó xác định c*hi phí khấu hao* (*amortized cost*) của mỗi thao tác được định nghĩa là chi phí trung bình cho mỗi thao tác, tức là $T(n)/n$.

Phương pháp gộp chung: phân tích stack

- Phân tích một chuỗi n thao tác lên một stack S (mà khi bắt đầu thì stack là trống), chuỗi gồm các loại thao tác
  - PUSH(S, x)
  - POP(S)
  - MULTIPOP(S, k)
- Dùng phương pháp gộp chung để xác định chi phí khấu hao của mỗi thao tác:
  - Nhận xét: Mỗi đối tượng có thể được lấy ra(popped) tối đa là một lần sau khi nó được đẩy vào (pushed).
  - Chuỗi thao tác của chúng ta có n thao tác => Số lần đẩy vào (PUSH) tối đa là n, vậy số lần POP được gọi (kể cả từ MULTIPOP) là n.
  - Vậy phí tổn của chuỗi n thao tác PUSH, POP, và MULTIPOP là $O(n)$.
- Do đó phí tổn khấu hao của mỗi thao tác là $O(n)/n = O(1)$.

Phương pháp gộp chung: phân tích bộ đếm nhị phân

- Phân tích một chuỗi gồm n thao tác INCREMENT lên một bộ đếm nhị phân có chiều dài k bit
- Nhận xét: Phân tích thô sơ
  - Khi mọi bit của bộ đếm là 1, thao tác INCREMENT có thể cần thời gian xấu nhất là $\Theta(k)$.
  - Do đó một chuỗi n thao tác INCREMENT cần thời gian xấu nhất là $nO(k) = O(nk)$.
- Dùng phương pháp gộp chung để xác định chi phí khấu hao của mỗi thao tác.
  - Nhận xét (giả sử trị ban đầu của bộ đếm là 0)

    bit $A[0]$ số lần bị đảo ngược n

    bit $A[1]$ $\lfloor n/2 \rfloor$

    bit $A[2]$ $\lfloor n/4 \rfloor$

    ...

$n = 5, k = 4$

0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1

Phương pháp gộp chung: phân tích bộ đếm nhị phân

Tổng quát:

bit $A[i]$ số lần bị đảo ngược $\lfloor n/2^i \rfloor$, $i = 0,..., \lfloor \lg n \rfloor$

bit $A[i]$ không bị đảo ngược nếu $i > \lfloor \lg n \rfloor$.

– Tính được tổng của $\lfloor n/2^i \rfloor$ từ $i = 0$ đến $\lfloor \lg n \rfloor$ là $\leq 2n$:

$$\sum_{i=0}^{\lfloor \lg n \rfloor} \frac{n}{2^i} < n \sum_{i=0}^{\infty} \frac{1}{2^i}$$

$$= 2n$$

$$\sum_{i=0}^{k} x^i = \frac{x^{k+1} - 1}{x - 1}$$

Phương pháp gộp chung: phân tích bộ đếm nhị phân

- Dùng phương pháp gộp chung để xác định chi phí khấu hao của mỗi thao tác (tiếp)
  - Vậy thời gian xấu nhất cho một chuỗi n thao tác INCREMENT lên một bộ đếm (mà trị ban đầu là 0) là $O(n)$.

    $\Rightarrow$ Thời gian khấu hao cho mỗi thao tác là $O(n)/n = O(1)$.

Phương pháp kế toán

- Trong phương pháp *kế toán* của phân tích khấu hao, chúng ta trả chi phí (charge) khác nhau cho các thao tác khác nhau.

  Ta có thể chi phí cho một thao tác nhiều hơn hay ít hơn phí tổn thực sự của nó.

  – Định nghĩa: Số lượng mà ta chi phí cho một thao tác được gọi là *phí tổn khấu hao* của nó.

  – Định nghĩa *chi phí trả trước* hay *credit* là chi phí khấu hao trừ chi phí thực sự.

    - Để cho chi phí khấu hao tổng cộng là chặn trên của chi phí thực sự tổng cộng thì tại mọi thời điểm credit tổng cộng phải ≥ 0.
    - (Nhưng credit cho một thao tác cá biệt có thể < 0 .)

Phương pháp kế toán: phân tích stack

- Các thao tác lên một stack
  - PUSH(S, x)
  - POP(S)
  - MULTIPOP(S, k) (POP k phần tử từ stack S)
- Dùng phương pháp kế toán để xác định chi phí khấu hao của mỗi thao tác lên một stack (ban đầu stack là trống).
  - Nhắc lại: phí tổn thực sự của các thao tác là
    - PUSH 1
    - POP 1
    - MULTIPOP min(k, s), (s là số phầntử trong S khi được gọi)
  - Ta trả cho các thao tác các phí tổn khấu hao như sau:
    - PUSH 2
    - POP 0
    - MULTIPOP 0
  - Cần phải chứng minh được tổng chi phí khấu hao là cận trên của tổng chi phí thực sự

Phương pháp kế toán: phân tích stack (tiếp)

– Diễn giải như sau: Có thể trả chi phí thực sự cho một chuỗi thao tác bất kỳ khi định ra phí tổn khấu hao (đã cho như trên).

Mô hình hóa cấu trúc dữ liệu stack bằng một chồng đĩa. (Dùng 1 đồng để tượng trưng 1 đơn vị phí tổn.):

- Giả sử thao tác là PUSH: trả 2 đồng, trong đó
  - dùng 1 đồng để trả cho phí tổn thực sự
  - dùng 1 đồng còn lại để trả trước phí tổn cho POP hay MULTIPOP sau này (nếu có). Để đồng này trong đĩa được pushed vào chồng đĩa.
- Giả sử thao tác là POP: trả 0 đồng, và
  - dùng 1 đồng đã được trả trước (khi trả cho PUSH, đồng này nằm trong đĩa được popped khỏi chồng đĩa.) để trả cho POP.
- Giả sử thao tác là MULTIPOP: trả 0 đồng, và
  - dùng các đồng đã được trả trước (khi trả cho PUSH) để trả cho MULTIPOP.

Phương pháp kế toán: phân tích stack (tiếp)

- Kết luận:
  - Cho một chuỗi bất kỳ gồm n thao tác PUSH, POP, và MULTIPOP, phí tổn khấu hao tổng cộng là một cận trên cho phí tổn thực sự tổng cộng vì tại mọi thời điểm tổng chi phí trả trước >=0 (bằng số phần tử đã được chen vào Stack).
  - Vì mỗi thao tác có phí tổn khấu hao là $O(1)$ nên một cận trên cho phí tổn thực sự tổng cộng là $O(n)$.

Phương pháp kế toán: phân tích một bộ đếm nhị phân

- Phân tích một chuỗi các thao tác INCREMENT lên một bộ đếm nhị phân dài *k*-bit mà giá trị ban đầu là 0.

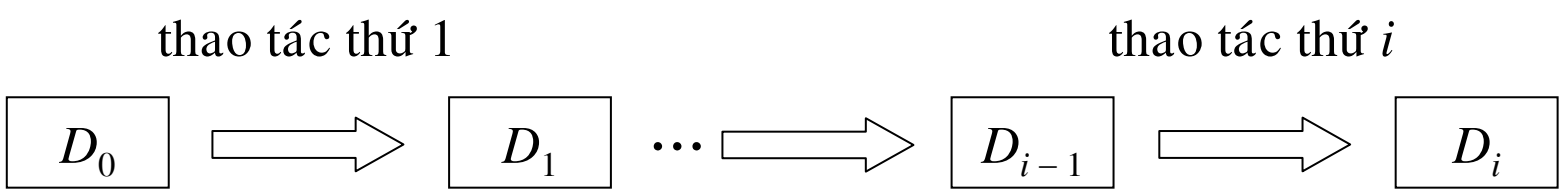
- Dùng phương pháp kế toán để xác định chi phí khấu hao của INCREMENT
  - Chúng ta đặt ra quy định như sau: dùng chi phí khấu hao là 2 đồng để set một bit thành 1.
    - dùng 1 đồng để trả phí tổn thực sự
    - dùng 1 đồng còn lại để trả trước cho phí tổn để reset bit này thành 0 sau này (nếu có).

Phương pháp kế toán: phân tích một bộ đếm nhị phân

- Xác định phí tổn khấu hao của INCREMENT (tiếp)
  - Phí tổn khấu hao của INCREMENT:
    - Phí tổn cho việc resetting các bits trong vòng lặp while được trả bằng các đồng đã được trả trước khi bit được set.
    - Trong INCREMENT thì nhiều nhất là 1 bit được set bằng 1.
  - *Do đó phí tổn khấu hao cho một thao tác INCREMENT tối đa là 2 đồng.*
- Vậy phí tổn khấu hao cho *n* thao tác INCREMENT liên tiếp (với giá trị ban đầu của bộ đếm là 0) là 2n hay $O(n)$.
  - Vì tổng số tiền trả trước không bao giờ âm (= số các bit có trị là 1 trong bộ đếm) nên chi phí khấu hao tổng cộng là cận trên cho chi phí thực sự tổng cộng.

Phương pháp thế năng

- Phân tích một chuỗi các thao tác lên một cấu trúc dữ liệu.
  - Cho một cấu trúc dữ liệu mà có n thao tác thực thi lên trên đó. Ban đầu cấu trúc dữ liệu là D_0
  - Gọi c_i là chi phí thực sự của thao tác thứ i, với $i = 1,..., n$.
  - Gọi D_i là cấu trúc dữ liệu có được sau khi áp dụng thao tác thứ i lên cấu trúc dữ liệu D_{i-1} , với $i = 1,..., n$.



Phương pháp thế năng (tiếp)

- Định nghĩa hàm số

  $\Phi : \{D_0, ..., D_n\} \rightarrow \Re$, với $\Re$ là tập hợp các số thực.

  Hàm Φ được gọi là (*hàm*) *thế năng* (*potential function*).

  Trị $\Phi(D_i)$ được gọi là *thế năng* của cấu trúc dữ liệu ở trạng thái D_i , $i = 0,...,n$.

- Định nghĩa: *phí tổn khấu hao* (*amortized cost*) của thao tác thứ i là:

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$$

Phương pháp thế năng (tiếp)

- Điều kiện đủ đối với thế năng để phí tổn khấu hao tổng cộng là cận trên lên phí tổn thực sự tổng cộng:
  - Ta có từ

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} (c_i + \Phi(D_i) - \Phi(D_{i-1}))$$

$$= \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0)$$

  - Vậy phí tổn khấu hao tổng cộng là cận trên của phí tổn thực sự tổng cộng nếu $\Phi(D_n) - \Phi(D_0) \geq 0$. Vì không biết trước n nên ta có điều kiện đủ sau

$$\Phi(D_i) - \Phi(D_0) \geq 0 \qquad \text{cho mọi } i$$

Phương pháp thế năng: phân tích một stack

- Phân tích một chuỗi gồm n thao tác lên một stack
  - PUSH(S, x)
  - POP(S)
  - MULTIPOP(S, k).
- Áp dụng phương pháp thế năng để xác định chi phí khấu hao của mỗi thao tác
  - Đặt giá trị thế năng Φ của một stack là số đối tượng trong stack.

top → 23
33
4
45
4
78
---

stack có thế năng là 6

Phương pháp thế năng: phân tích một stack (tiếp)

– Nhận xét:

- Khi bắt đầu thì stack trống nên $\Phi(D_0) = 0$.
- $\Phi(D_i) \geq 0$, vậy $\Phi(D_i) \geq \Phi(D_0)$ cho mọi i.

Vậy phí tổn khấu hao tổng cộng là cận trên của phí tổn thực sự tổng cộng.

- Áp dụng phương pháp thế năng để xác định chi phí khấu hao của mỗi thao tác:

– Giả sử thao tác thứ i lên stack là PUSH

(stack có kích thước là s)

- Hiệu thế là $\Phi(D_i) - \Phi(D_{i-1}) = (s + 1) - s$

$= 1$

- Vậy phí tổn khấu hao của PUSH

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$$
$$= 1 + 1 = 2 .$$

Phương pháp thế năng: phân tích một stack

- Áp dụng phương pháp thế năng để xác định chi phí khấu hao của mỗi thao tác (tiếp)
  - Giả sử thao tác thứ i lên stack là MULTIPOP(S, k)

    (stack có kích thước là s)
    - Phí tổn thực sự là $k' = \min(k, s)$
    - Hiệu thế là $\Phi(D_i) - \Phi(D_{i-1}) = -k'$
    - Vậy phí tổn khấu hao của MULTIPOP là

$$\begin{aligned} \hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}) \\ &= k' - k' \\ &= 0 . \end{aligned}$$

Phương pháp thế năng: phân tích một stack

- Áp dụng phương pháp thế năng để xác định chi phí khấu hao của mỗi thao tác (tiếp)
  - Phí tổn khấu hao của POP là 0.

- Vậy phí tổn khấu hao tổng cộng của một chuỗi n thao tác lên stack là $O(n)$.

  Do $\Phi(D_i) \geq \Phi(D_0)$ cho mọi i, vậy phí tổn trong trường hợp xấu nhất của n thao tác là $O(n)$.

Phương pháp thế năng: phân tích bộ đếm nhị phân dài *k* bits

- Phân tích một chuỗi các thao tác lên một bộ đếm nhị phân dài *k*-bit.
- Dùng phương pháp thế năng để xác định chi phí khấu hao của mỗi thao tác
  - Định nghĩa *thế năng* Φ *của bộ đếm* sau khi thực hiện thao tác INCREMENT thứ i là b_i - số các bits bằng 1 trong bộ đếm.

0	1	1	1

Thế năng là 3

Phương pháp thế năng: phân tích bộ đếm nhị phân dài k bits

- Dùng phương pháp thế năng để xác định chi phí khấu hao của mỗi thao tác (tiếp)
  - Tính phí tổn khấu hao của một thao tác INCREMENT
    - INCREMENT thứ i resets t_i bits về 0 và đặt một bít bằng 1.
      Vậy phí tổn thực sự của thao tác INCREMENT thứ i là $t_i + 1$.
    - Hiệu thế là
      $\Phi(D_i) - \Phi(D_{i-1}) = b_i - b_{i-1}$, mà $b_i = b_{i-1} - t_i + 1$. Vậy
      $\Phi(D_i) - \Phi(D_{i-1}) = 1 - t_i$.
    - Phí tổn khấu hao là
      $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = t_i + 1 + 1 - t_i = 2.$

Phương pháp thế năng: phân tích bộ đếm nhị phân dài k bits

- Giá trị của bộ đếm bắt đầu bằng 0, nên $\Phi(D_0) = 0$, do đó $\Phi(D_i) \geq \Phi(D_0)$. Vậy chi phí khấu hao tổng cộng là chặn trên cho chi phí thực sự tổng cộng.

  $\Rightarrow$ Phí tổn trong trường hợp xấu nhất của n thao tác là $O(n)$.

So sánh các phương pháp kế toán và thế năng

	Kế toán	Thế năng
Chi phí khấu hao =	Các thao tác khác nhau được trả chi phí khác nhau.	Chi phí thực sự + hiệu thế năng.
(Chi phí khấu trừ – chi phí thực sự) =	Chi phí trả trước	Hiệu thế năng.
Điều kiện đủ	Chi phí trả trước tổng cộng ≥ 0	Mọi hiệu thế năng $\Phi(D_i) - \Phi(D_0) \geq 0$

Bảng động

- Trong một số ứng dụng, dùng một "bảng" để lưu trữ các đối tượng mà không biết trước bao có nhiêu đối tượng sẽ được lưu trữ. Do đó:
  - khi bảng hiện thời không có đủ chỗ cho các đối tượng mới, cần một bảng mới với kích thước lớn hơn.
  - khi bảng hiện thời dư nhiều chổ trống do xoá nhiều đối tượng, cần một bảng mới với kích thước nhỏ hơn.

- Các thao tác lên một bảng
  - TABLE-INSERT: chèn một item vào bảng
  - TABLE-DELETE: xóa một item khỏi bảng.

Hệ số sử dụng của một bảng

- Định nghĩa *hệ số sử dụng* (load factor) của một bảng T là $\alpha(T)$:

  $\alpha(T)$ = số khe (slots) có chứa đối tượng trong bảng chia cho số khe của bảng.

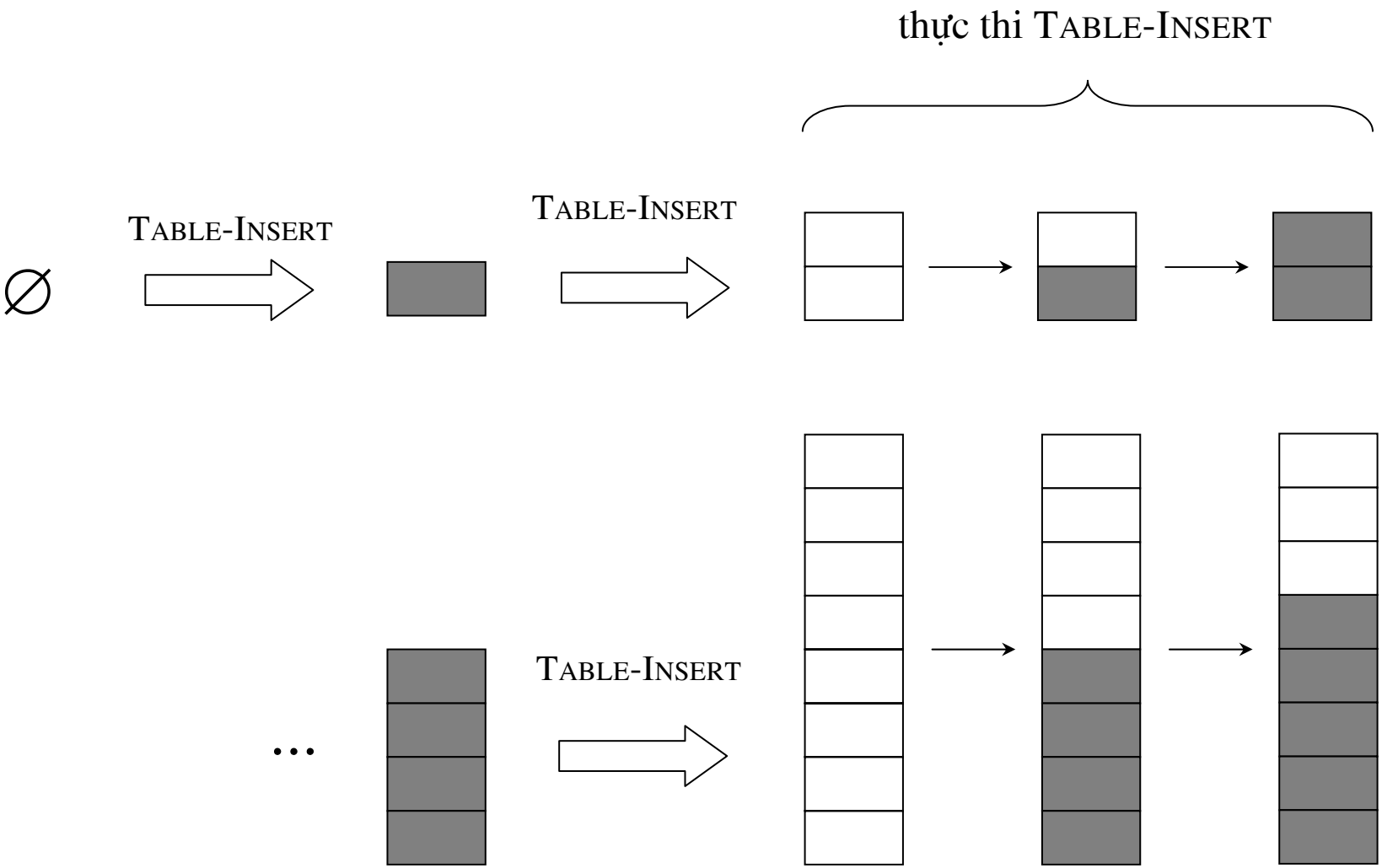
- Bài toán: Xác định chiến lược *mở rộng* và *thu nhỏ* bảng sao cho: hệ số sử dụng của bảng *cao (*được chặn dưới bởi một hằng số). Biết chi phí của TABLE-INSERT và TABLE-DELETE là $O(1)$.

Chiến lược mở rộng một bảng

- Chiến lược khi nào thì mở rộng một bảng được diễn tả trong giải thuật sau.

```
TABLE-INSERT(T, x)
 1      if size[T] = 0
 2         then allocate table[T] with 1 slot
 3              size[T] ← 1
 4      if number[T] = size[T]
 5         then allocate new-table with 2 · size[T] slots
 6              insert all items in table[T] into new-table
 7              free table[T]
 8              table[T] ← new-table
 9              size[T] ← 2 · size[T]
10      insert x into table[T]
11      Number[T] ← number[T] + 1
```

Chiến lược mở rộng một bảng (tiếp)



Phân tích một chuỗi TABLE-INSERT

- Giả sử:
  - Thời gian thực thi của TABLE-INSERT tỉ lệ với thời gian chèn từng item ("chèn sơ đẳng") vào bảng ở dòng 6 và 10.
  - Chi phí của một chèn sơ đẳng là 1.

◦ Chúng ta đi phân tích chi phí của một chuỗi gồm n INSERT lên một bảng động dùng bằng các phương pháp
  - gộp chung
  - kế toán
  - thế năng.

k6

Phân tích chuỗi TABLE-INSERT bằng phương pháp gộp chung

- Dùng phương pháp gộp chung để xác định chi phí khấu hao của INSERT
  - Chi phí thực sự c_i của thao tác Insert thứ i
    - là i nếu $i - 1 = 2^j$ *(trường hợp phải mở rộng bảng)*
    - là 1 trong các trường hợp còn lại.
  - Chi phí của n thao tác TABLE-INSERT
    - là n + tổng các 2^j từ $j = 0$ đến $\lfloor \lg_2 n \rfloor$

      $\leq n + 2n = 3n = T(n)$.

$$\sum_{0}^{\log_2(n)} 2^j = \frac{2^{\log_2(n)+1} - 1}{2 - 1} <= 2n$$

- Vậy chi phí khấu hao của INSERT là T(n)/n bằng $3n / n = 3$.

k6

khmt-hung, 12/4/2006

Phân tích chuỗi TABLE-INSERT bằng phương pháp kế toán

- Dùng phương pháp kế toán để xác định chi phí khấu hao của TABLE-INSERT
  - Chi phí khấu hao của TABLE-INSERT là 3.
  - Trả như sau:
    - 1 đồng để trả cho chi phí thực sự cho riêng nó
    - 1 đồng để trả trước cho chính nó một khi nó được di chuyển sang bảng nới rộng
    - 1 đồng để trả trước cho một item khác trong bảng mà không còn tiền trả trước (vì đã di chuyển).

k2 Giả sử ngay sau khi mở rộng bảng có kích thước m và số tiến trả trước là 0 đồng. Hiện tại số phần tử trong bảng động là m/2. Thêm một phần tử nữa với chi phí khấu hao là 3 (1 đồng là chi phí thực sự, một đồng là tiến trả trước cho việc di dời, 1 đồng là tiền trả trước cho m/2 phần tử cần đến khi di dời). Như vậy cho đến lúc bảng kích thước m đầy thì m/2 phần tử mỗi phần tử đều có 1 đồng trả trước để di dời.

khmt-hung, 12/4/2006

Phân tích chuỗi TABLE-INSERT bằng phương pháp thế năng

- Dùng phương pháp thế năng để phân tích một chuỗi gồm n thao tác INSERT lên một bảng
  - Định nghĩa thế năng Φ là

    $\Phi(T) = 2number[T] - size[T]$.
  - Nhận xét
    - Ngay *sau* khi nới rộng (dòng 9 của TABLE-INSERT thực thi xong)

      $Number[T] = size[T] / 2$

      $\Rightarrow \Phi(T) = 0$.
    - Ngay *trước* khi nới rộng $number[T] = size[T]$

      $\Phi(T) = number[T]$.
    - $\Phi(0) = 0$, $\Phi(T) \geq 0$. Vì vậy tổng của các chi phí khấu hao của n thao tác TABLE-INSERT là một chận trên lên tổng của các chi phí thực sự.

Phân tích chuỗi TABLE-INSERT bằng phương pháp thế năng

(tiếp)

– Xác định chi phí khấu hao của mỗi thao tác

• Giả sử thao tác thứ i không gây nới rộng. Ta có $size_i = size_{i-1}$. và chi phí thực sự bằng 1.

Chi phí khấu hao của thao tác là

$$\begin{aligned} \hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\ &= 1 + (2num_i - size_i) - (2num_{i-1} - size_{i-1}) \\ &= 1 + (2num_i - size_i) - (2(num_i - 1) - size_i) \\ &= 3 . \end{aligned}$$

Phân tích chuỗi TABLE-INSERT bằng phương pháp thế năng

– Xác định chi phí khấu hao của mỗi thao tác (tiếp)

- Giả sử thao tác thứ *i* gây nới rộng. Ta có

  $size_i / 2 = size_{i-1} = number_{i-1} = number_i - 1$. Và chi phí thực sự bằng $number_i$

Chi phí khấu hao của thao tác là:

$$\begin{aligned} \hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\ &= number_i + (2number_i - size_i) - (2number_{i-1} - size_{i-1}) \\ &= number_i + (2number_i - (2number_i - 2)) - (2(number_i - 1) - (number_i - 1)) \\ &= number_i + 2 - (number_i - 1) \\ &= 3 . \end{aligned}$$

Xóa một item khỏi bảng

- Thêm thao tác "*Xóa một item khỏi bảng*": TABLE-DELETE.
  - Hệ số sử dụng của bảng có thể trở nên quá nhỏ.
- Nhắc lại định nghĩa của *hệ số sử dụng* là

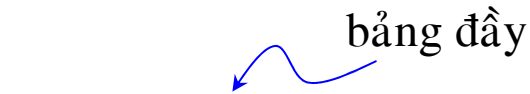
  $\alpha(T) = num[T] / size[T]$.
- Ta muốn
  - giữ hệ số sử dụng cao, tức là nó được chặn dưới bằng một hằng số.
  - chi phí khấu hao của một thao tác lên bảng được chặn trên bằng một hằng số.
- Giả sử chi phí thực sự được đo bằng số lần chèn hay xoá item sơ đẳng.

Chiến lược nới rộng và thu nhỏ bảng

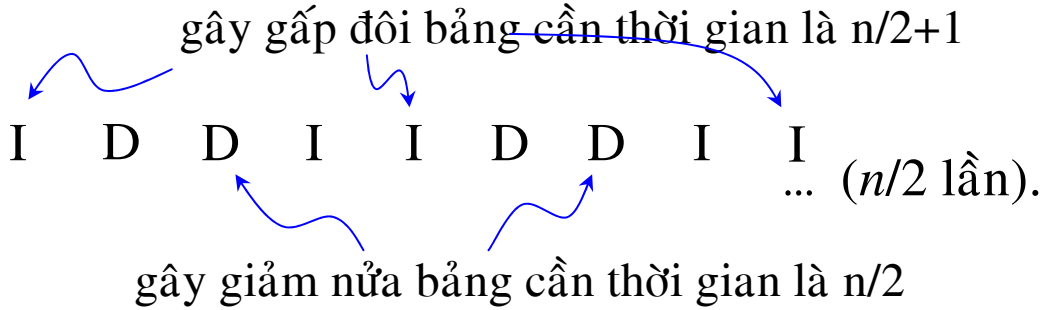
- Một chiến lược tự nhiên cho nới rộng và thu nhỏ bảng là
  - Gấp đôi bảng khi chèn một item vào một bảng đã đầy.
  - Giảm nửa bảng khi xóa một item khỏi một bảng chỉ đầy nửa bảng.
- => Chiến lược trên bảo đảm $\alpha(T) \geq 1/2$.

Chiến lược nới rộng và thu nhỏ bảng

- Phân tích
  - Phí tổn khấu hao của mỗi thao tác có thể rất lớn:
    - Cho n có dạng 2^m
    - Xét một chuỗi n thao tác (I là một insert và D là một delete) theo kịch bản sau:



I ... I ($n/2$ lần), khi đó $number[T] = size[T] = n/2$. Các thao tác tiếp theo là

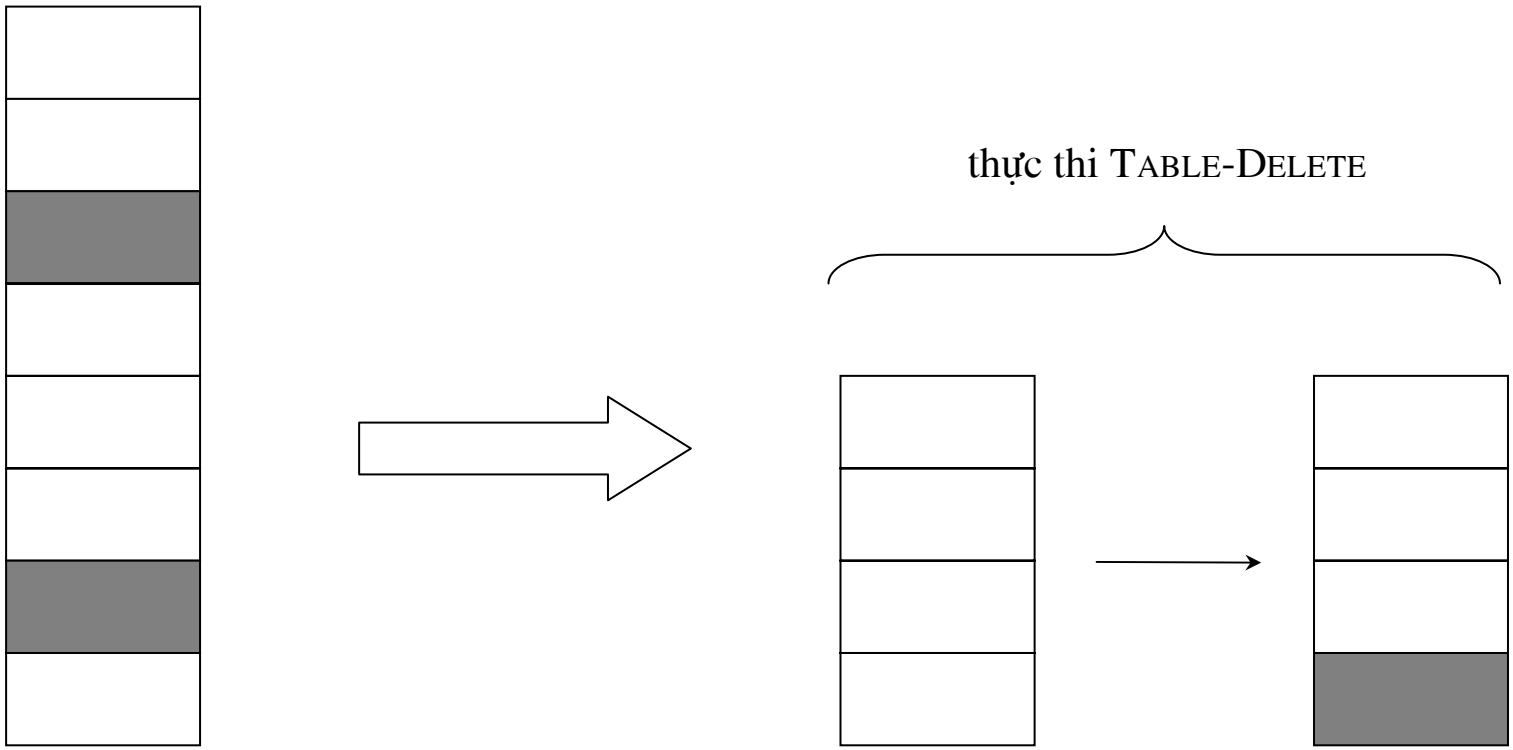


    - Chuỗi n thao tác này có phí tổn là $\Theta(n^2)$, do đó phí tổn khấu hao của mỗi thao tác là $\Theta(n)$.

Chiến lược nới rộng và thu nhỏ bảng (tiếp)

- Cải tiến chiến lược trên bằng cách cho phép hệ số sử dụng có thể trở nên nhỏ hơn 1/2:
  - Nếu $\alpha(T) = 1$, thì thao tác TABLE-INSERT sẽ gấp đôi bảng.
  - Nếu $\alpha(T) = 1/4$, thì thao tác TABLE-DELETE sẽ giảm nửa bảng.

Chiến lược thu nhỏ một bảng (tiếp)



$\alpha(T) = 1/4$

bảng mới =
1/2 bảng cũ

Phương pháp thế năng

- Dùng phương pháp thế năng để phân tích một chuỗi gồm n thao tác TABLE-INSERT và TABLE-DELETE lên một bảng động với chiến lược mở rộng và thu nhỏ bảng mới ở trên:
  - Định nghĩa *thế năng* Φ *trên một bảng* là

$$\Phi(T) = 2\,num[T] - size[T] \quad \text{nếu } \alpha(T) \geq 1/2 \quad (1)$$
$$= size[T] / 2 - num[T] \text{ nếu } \alpha(T) < 1/2 \quad (2)$$

Phương pháp thế năng (tiếp)

– Nhận xét:
  - Φ(bảng trống) = 0, và $\Phi(T) \geq 0$
  - Nếu hệ số sử dụng là 1/2 thì $\Phi(T) = 0$.
  - Nếu hệ số sử dụng là 1 thì $\Phi(T) = num[T]$.
    – Đủ để trả phí tổn một khi có nới rộng bảng do chèn một item.
  - Nếu hệ số sử dụng là 1/4 thì $\Phi(T) = num[T]$.
    – Đủ để trả phí tổn một khi có thu nhỏ bảng do xoá một item.

k1

Phân tích một chuỗi các TABLE-INSERT và TABLE-DELETE

- Xác định chi phí khấu hao của mỗi thao tác
  - Nếu thao tác thứ *i* là TABLE-INSERT, ta phân biệt các trường hợp:
    - $\alpha_{i-1} \geq 1/2$
      - theo mục trước, thì $c\hat{}_i$ là 3.(Chiến lược trên bảo đảm $\alpha(T) \geq 1/2$)
    - $\alpha_{i-1} < 1/2$, ta phân biệt 2 trường hợp:
      - trường hợp $\alpha_i < 1/2$ ***($size_i = size_{i-1}$) hàm thế năng tính theo CT (2)***

$$c\hat{}_i = c_i + \Phi_i - \Phi_{i-1}$$
$$= 1+(size_i/2-num_i) - (size_{i-1}/2-num_{i-1})$$
$$= 1+size_i/2-num_i- size_{i-1}/2+num_i-1 =0 .$$

      - trường hợp $\alpha_i \geq 1/2$ ***($size_i = size_{i-1}$) hàm thế năng tính theo CT (1)***

$$c\hat{}_i = c_i + \Phi_i - \Phi_{i-1}$$
$$= 1+(2*(num_{i-1}+1)-size_i)- (size_{i-1}/2-(num_{i-1}))$$
$$= 3*num_{i-1} -(3/2)\ size_{i-1} +3 = 3\ \alpha_{i-1}\ size_{i-1} -(3/2)\ size_{i-1} +3$$
$$< (3/2)\ size_{i-1} -(3/2)\ size_{i-1}+3 =3.$$

- Vậy chi phí khấu hao của thao tác TABLE-INSERT nhiều nhất là 3.

k1

Alpha (T) = 2 num[T]- size[T] Nếu Alpha(T)>= 1/2
= size[T] / 2 - num[T] Nếu Alpha(T)< 1/2

khmt-hung, 12/4/2006

Phân tích một chuỗi các TABLE-INSERT và TABLE-DELETE

- Xác định chi phí khấu hao của mỗi thao tác (tiếp)
  - Nếu thao tác thứ i là TABLE-DELETE, thì $num_i = num_{i-1} - 1$, ta phân biệt các trường hợp:
    - $\alpha_{i-1} < 1/2$. Có hai trường hợp con
      - không gây thu nhỏ: $c_i=1$; $size_i = size_{i-1}$

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1}$$
$$= 2 .$$

      - gây thu nhỏ: $c_i = num_i + 1$, $size_i / 2 = size_{i-1} / 4 = num_i + 1$

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1}$$
$$= 1 .$$

$$\begin{aligned}\hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\ &= (num_i + 1) + (size_i/2 - num_i) - (size_{i-1}/2 - num_{i-1}) \\ &= (num_i + 1) + ((num_i + 1) - num_i) - ((2 \cdot num_i + 2) - (num_i + 1)) \\ &= 1 .\end{aligned}$$

k3

Alpha (T) = 2 num[T]- size[T] Nếu Alpha(T)>= 1/2
= size[T] / 2 - num[T] Nếu Alpha(T)< 1/2

khmt-hung, 12/4/2006

Phân tích một chuỗi các TABLE-INSERT và TABLE-DELETE

- $\alpha i-1 \geq 1/2$. không gây thu nhỏ bản =>$size_i=size_{i-1}$
  - Giả sử $\alpha_i >= 1/2$

$$\begin{aligned}\hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\ &= 1 + (2 \cdot num_i - size_i) - (2 \cdot num_{i-1} - size_{i-1}) \\ &= 1 + (2 \cdot num_i - size_i) - (2 \cdot (num_i + 1) - size_i) \\ &= -1\end{aligned}$$

  - Giả sử $\alpha_i < 1/2$

$$\begin{aligned}\hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\ &= 1 + (2 \cdot num_i - size_i) - (size_{i-1}/2 - num_{i-1}) \\ &= 3 \cdot num_i - 3/2 \cdot size_i + 2 \\ &= 3\alpha_i \cdot size_i - 3/2 \cdot size_i + 2 \\ &< 3/2 \cdot size_i - 3/2 \cdot size_i + 2 \\ &= 2\end{aligned}$$

Phân tích một chuỗi các TABLE-INSERT và TABLE-DELETE

(tiếp)

- Tổng kết ta thấy chi phí khấu hao của TABLE-DELETE được chặn trên bởi một hằng số.

- Kết luận: Vì chi phí khấu hao của mỗi thao tác TABLE-INSERT và TABLE-DELETE được chặn trên bởi một hằng số, nên thời gian chạy cho một chuỗi bất kỳ gồm n thao tác lên một bảng động là $O(n)$.

Bài tập

- 18.1-2: Show that if a DECREMENT operation were included in the k-bit counter example, n operations could cost as much as (nk) time.