Análise Amortizada

Eduardo Camponogara

Departamento de Automação e Sistemas
Universidade Federal de Santa Catarina

6 de abril de 2009

Introdução

Método de Agregação

Método Contábil

Método da Energia Potencial

Sumário

Introdução

Método de Agregação

Método Contábil

Método da Energia Potencial

Análise Amortizada

Princípios

- Em uma análise amortizada, o tempo de execução de uma operação é a média sobre uma sequência de operações
- Pode ser usada para mostrar que o custo médio é baixo, embora algumas operações sejam onerosas
- Análise amortizada se diferencia da análise probabilística no sentido que probabilidades não estão envolvidas
- A análise amortizada garante *desempenho médio de cada operação no pior caso*

Análise Amortizada

Três Métodos

1) Método da agregação
2) Método contábil
3) Método da energia potencial

Análise Amortizada

Método da Agregação

- Determinamos um limite superior $T(n)$ para o custo total de n operações.
- O custo amortizado por operação é $\frac{T(n)}{n}$

Análise Amortizada

Método da Contábil

- Cada operação tem um custo amortizado diferenciado
- O método contábil sobretaxa algumas operações no início de uma sequência (de operações), armazenando a sobretaxa como "crédito pré-pago" em objetos da estrutura de dados.
- O crédito é usado mais tarde na sequência para subsidiar operações que são taxadas abaixo do custo real.

Análise Amortizada

Método da Energia Potencial

- Determina-se o custo amortizado de cada operação
- Operações iniciais da sequência podem ser sobretaxadas para manter um "crédito" na forma de "energia potencial" da estrutura de dados
- A "energia potencial" é da estrutura de dados como um todo, não sendo associado crédito a objetos individuais como no método contábil

Sumário

Introdução

Método de Agregação

Método Contábil

Método da Energia Potencial

Método de Agregação

Princípios

- Mostra-se que para todo n, uma sequência de n operações tem no pior-caso um custo $T(n)$ total
- Logo, o custo amortizado por operação é $\frac{T(n)}{n}$

Operações com Pilhas

Operações básicas

- *push*(S, x): empilha um objeto x na pilha S
- *pop*(S): remove o elemento que está no topo da pilha S
- Uma vez que cada operação custa $O(1)$, o custo total de uma sequência de n operações (*push* e *pop*) é portanto $\Theta(n)$

Operações com Pilhas

Introduzindo Operação *multipop*(S, k)

- *multipop*(S, k): remove os k elementos no topo da pilha S, ou remove a pilha inteira se a pilha tem menos que k objetos

Pseudo-código para *multipop*(S, k)

```
multipop(S, k)
    while not empty_stack(S) and k ≠ 0
        do pop(S)
            k ← k − 1
```

Operações com Pilhas

Pilha Inicial

23
17
6
39
10
47

$$multipop(S, 4) \Longrightarrow$$

Pilha Final

10
47

Qual é o tempo de execução de $multipop(S, k)$ em uma pilha com s objetos?

- $multipop(S, k)$ leva tempo $\min(s, k) = \min(|S|, k)$

Análise

Análise de uma sequência qualquer de n operações *push*, *pop* e *multipop*.

1. No pior caso, cada operação leva tempo $O(n)$
2. Logo uma sequência com n operações tem custo total $O(n^2)$, uma vez que podemos ter $O(n)$ operações *multipop*, cada uma com custo $O(n)$
3. $O(n^2)$ é correto, mas não define um limite apertado para $T(n)$

Análise Refinada

Na verdade, qualquer sequência de operações *pop*, *push* e *multipop* em um pilha inicialmente vazia custa no máximo $O(n)$.

- Cada objeto x pode ser removido no máximo uma vez para cada vez que ele é empilhado
- O número de vezes que *pop* é executado a partir de uma pilha inicialmente vazia, incluindo *pop*'s dentro de *multipop*'s, é no máximo o número de vezes que *push* foi executado, sendo este no máximo n vezes

Análise Refinada

Conclusão

Qualquer sequência de n operações tem custo total $O(n)$ e, portanto, o custo amortizado por operação é:

$$\frac{T(n)}{n} = \frac{O(n)}{n} = O(1)$$

Contador Binário

- Considere o problema de implementar um contador binário de k-bits que conta de 0 pra cima
- Usamos um vetor $A[0..k-1]$ de bits, onde $length[A] = k$
- O valor do contador é:

$$x = \sum_{i=0}^{k-1} A[i]2^i$$

- Inicialmente, $x = 0$

Contador Binário

Algoritmo

Para adicionar 1 $(\mathrm{mod}\ 2^k)$ ao valor do contador, usamos o procedimento:

$increment(A)$
- $i \leftarrow 0$
- while $i < length[A]$ and $A[i] = 1$
    - do $A[i] \leftarrow 0$
        - $i \leftarrow i + 1$
- if $i < length[A]$
    - then $A[i] \leftarrow 1$

Análise Grosseira

- No pior caso, cada operação *increment* leva tempo $O(k)$, quando todas as entradas de A são 1
- Portanto, uma sequência de n operações *increment* leva no máximo $T(n) = O(nk)$
- Daí concluímos que o custo amortizado por operação é

$$\frac{T(n)}{n} = \frac{O(nk)}{n} = O(k)$$

Exemplo

Exemplo

5	4	3	2	1	0
0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	0	1	1
0	0	0	1	0	0
0	0	0	1	0	1
0	0	0	1	1	0
0	0	0	1	1	1

Análise Refinada

- Por inspeção, $A[0]$ troca a cada vez, enquanto $A[1]$ troca a cada duas vezes
- Em geral, para $i = 0, 1, \ldots, \lfloor \lg n \rfloor$, o bit $A[i]$ troca $\left\lfloor \frac{n}{2^i} \right\rfloor$ vezes em uma sequência de n operações *increment* a partir de um contador zerado.

Resultado

O número total de trocas de bits em uma sequência de tamanho n é:

$$T(n) = \sum_{i=0}^{\lfloor \lg n \rfloor} \left\lfloor \frac{n}{2^i} \right\rfloor < n \sum_{i=0}^{\infty} \frac{1}{2^i} = 2n$$

Análise Refinada

Análise Amortizada

O tempo de execução de n operações *increment* é no pior caso, a partir de um contado zerado, limitado por $O(n)$. Portanto, o custo amortizado por operação é:

$$\frac{T(n)}{n} = \frac{O(n)}{n} = O(1)$$

Sumário

Introdução

Método de Agregação

Método Contábil

Método da Energia Potencial

Método Contábil

Princípios

- No Método Contábil, designamos taxas diferenciadas às operações, sendo algumas operações taxadas mais ou menos do que o custo real
  - A taxa cobrada de uma operação é o **custo amortizado**
- Quando o custo amortizado exceder o custo real, a diferença é creditada em objetos específicos da estrutura de dados
- **Créditos** podem ser usados mais tarde para ajudar no pagamento de operações cujos custos amortizados são inferiores aos custos reais

Método Contábil

Princípios

- Para mostrar que (no pior caso) o custo médio por operação é pequeno, temos que mostrar que o custo amortizado total de uma sequência define um limite superior para o custo real da sequência:
- Além disso, a relação deve ser mantida para todas as sequências de operações
- Note que o crédito total associado com uma estrutura de dados deve ser sempre não negativo

Operações com Pilhas

Custos Reais

- *push* $\longrightarrow O(1)$
- *pop* $\longrightarrow O(1)$
- *multipop* $\longrightarrow \min(k, s)$, onde k é o argumento da operação *multipop* e s é o tamanho da pilha no momento da chamada

Custos Amortizados

- *push* $\longrightarrow 2$
- *pop* $\longrightarrow 0$
- *multipop* $\longrightarrow 0$

Operações com Pilhas

- Note que o custo amortizado de *multipop* é uma constante (0), enquanto que o custo real é uma variável
- Quando empilhamos um objeto na pilha, usamos uma unidade do custo de *push* para executar a operação, deixando a outra unidade sobre o elemento
- A todo instante, uma unidade está associada (depositada) a cada elemento da pilha
- Este crédito é um pré-pagamento para cobrir o custo da operação *pop*
- Não taxamos *pop* e *multipop* pois os custos reais destas operações foram pagos antecipadamente

Operações com Pilhas

Observações

- Nós asseguramos que a quantidade de crédito é sempre não negativa
- Portanto, para qualquer sequência de n operações *push*, *pop* e *multipop*, o custo amortizado é um limite superior para o custo real.
- O custo amortizado total é $O(n)$ e, portanto, o custo total é $T(n) = O(n)$. Logo o custo amortizado por operação é:

$$\frac{T(n)}{n} = \frac{O(n)}{n} = O(1)$$

Análise Amortizada do Contador Binário

Custos Amortizados

- Cobramos 2 unidades para setar um bit em 1.
- Quando um bit é modificado para 1, usamos uma unidade para cobrir o custo da operação em si, mas colocamos a outra unidade sobre o bit em forma de crédito
- Em qualquer momento, todo bit 1 do contador tem uma unidade de crédito associada e, portanto, não precisamos cobrar taxa extra para resetá-lo em 0

Contador Binário

Algoritmo

Para adicionar 1 $\pmod{2^k}$ ao valor do contador, usamos o procedimento:

```
increment(A)
    i ← 0
    while i < length[A] and A[i] = 1
        do A[i] ← 0
           i ← i + 1
    if i < length[A]
        then A[i] ← 1
```

Contador Binário

Observações

- O custo de resetar bits dentro do laço *while* já foi pago com as unidades que estão nos bits a serem resetados
- No máximo um bit é setado, na linha 6 da operação *increment*, portanto o custo amortizado de *increment* é 2
- Uma vez que o número de 1's no contador nunca é negativo, a quantidade de crédito é sempre não negativa

Contador Binário

Observações

Assim, para uma sequência de n operações *increment*, o custo total é $O(n)$, que limita o custo total. Daí concluímos que o custo amortizado por operação é:

$$\frac{T(n)}{n} = \frac{O(n)}{n} = O(1)$$

Sumário

Introdução

Método de Agregação

Método Contábil

Método da Energia Potencial

Método da Energia Potencial

Princípios

- Em vez de representar trabalho pré-pago na forma de crédito em objetos específicos da estrutura de dados, o *método da energia potencial* representa trabalho pré-pago na forma de "*energia potencial*" ou simplesmente "*potencial*", que pode ser liberado para cobrir custos de operações futuras
- O potencial é associado à estrutura de dados como um todo em vez de objetos específicos dentro da estrutura de dados

Método da Energia Potencial

Princípios

Para cada $i = 1, 2, \ldots, n$

- c_i: é o custo real da operação
- D_i: é a estrutura de dados que resulta da aplicação da i-ésima operação na estrutura de dados D_{i-1}

Função Potencial

- A *função potencial* Φ mapeia cada estrutura de dados D_i a um número real $\Phi(D_i)$ com o potencial associado à estrutura de dados
- O *custo amortizado* $\hat{c}_i$ da i-ésima operação com respeito à função potencial Φ é definido por:

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

O custo amortizado é, portanto, o custo atual mais o acréscimo de energia potencial incorrido pela operação

Função Potencial

- A partir da equação anterior, concluímos que o custo amortizado após n operações é:

$$\begin{aligned}\sum_{i=1}^{n} \hat{c}_i &= \sum_{i=1}^{n}(c_i + \Phi(D_i) - \Phi(D_{i-1})) \\ &= \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0)\end{aligned}$$

Se pudermos definir a função potencial Φ tal que $\Phi(D_n) \geq \Phi(D_0)$, então o custo amortizado total $\sum_{i=1}^{n} \hat{c}_i$ induz um limite superior para o custo real

Função Potencial

- Intuitivamente, se a diferença de potencial $\Phi(D_i) - \Phi(D_{i-1})$ da i-ésima operação é positiva, então o custo amortizado foi maior do que o custo real da operação e, portanto, o potencial da estrutura de dados aumenta

Função Potencial e Operações com Pilhas

- Definimos Φ como o número de itens na pilha
- Para uma pilha vazia D_0, temos $\Phi(D_0) = 0$
- Uma vez que o tamanho da pilha nunca é negativo, teremos

$$\Phi(D_i) \geq 0 = \Phi(D_0), \ \ i = 1, 2, \ldots$$

- Logo, o custo amortizado total de n operações com respeito a Φ representa um limite superior para o custo real

Função Potencial e Operações com Pilhas

Operação *Push*

- Quando a operação *push* é executada em uma pilha com s elementos, temos como variação de energia potencial:

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

- Por outro lado, o custo real da operação é $c_i = 1$
- Destes resultados, derivamos que o custo amortizado da operação *push* é:

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

Função Potencial e Operações com Pilhas

Operação *Multipop*

- Suponha que a i-ésima operação executada é $multipop(s, k)$ e $k' = min(s, k)$ objetos são removidos da pilha
- O custo real da operação é $c_i = k'$
- A variação em energia potencial será

$$\Phi(D_i) - \Phi(D_{i-1}) = (s - k') - s = -k'$$

- Daí inferimos que o custo amortizado $\hat{c}_i$ é:

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

Função Potencial e Operações com Pilhas

Operação *Pop*

Pode ser verificado que o custo amortizado da operação *pop* é também 0.

Função Potencial e Operações com Pilhas

Conclusão

- O custo amortizado de cada operação é $O(1)$, logo o custo amortizado total de n operações é $O(n)$
- Uma vez que $\Phi(D_i) \geq \Phi(D_0)$, verificamos que o custo amortizado total é um limite superior para o custo total
- O custo de n operações é no pior caso $O(n)$

Função Potencial e Contador Binário

Definimos o potencial do contador após a i-ésima operação como o *número de bits 1* no contador, sendo denotado por b_i

Função Potencial e Contador Binário

- Suponha que a i-ésima operação reseta t_i bits.
- O custo real da operação é portanto no máximo $t_i + 1$, uma vez que além de resetar t_i bits, setamos um bit em 1
- O número de bits 1 no contador após a i-ésima operação é portanto $b_i = b_{i-1} - t_i + 1$
- A diferença em energia potencial fica

$$\begin{aligned} \Phi(D_i) - \Phi(D_{i-1}) &= (b_{i-1} - t_i + 1) - b_{i-1} \\ &= 1 - t_i \end{aligned}$$

Função Potencial e Contador Binário

- O custo amortizado é portanto:

$$\begin{aligned}
\hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}) \\
&= (t_i + 1) + (1 - t_i) \\
&= 2
\end{aligned}$$

Análise Amortizada

- Fim!
- Obrigado pela presença