Aula 21: Análise amortizada e estruturas de dados

David Déharbe
Programa de Pós-graduação em Sistemas e Computação
Universidade Federal do Rio Grande do Norte
Centro de Ciências Exatas e da Terra
Departamento de Informática e Matemáica Aplicada

Introdução

Análise amortizada

- Método agregado
- Método do contador
- Método do potencial

Arranjos dinâmicos

1. Avaliação de estruturas de dados: análise amortizada
   - Contador binário
   - Arranjos dinâmicos

PPgSC

Análise amortizada

- Medição da complexidade ao longo de n operações
- Custo de execução, no pior caso: $T(n)$
- Custo amortizado: $T(n)/n$
- Exemplos
    1. contador binário
    2. arranjo dinâmico

Método agregado

- Considere uma série de n operações em alguma estrutura de dados, n qualquer.
- Determina o custo $T(n)$ desta série de operações.
- O custo médio de cada operação é $T(n)/n$.
- Exemplo: contador binário

PPgSC

Método agregado: um exemplo

```
INCREMENT(A)
    // A é um arranjo de k bits
    i = 0
    while i < length[A] and A[i] = 1
        A[i] = 0
        i = i + 1
    if i < length[A]
        A[i] = 1
```

- O tamanho da entrada é k;
- O custo da execução da INCREMENT é proporcional ao número de bits invertidos.
- No pior caso, a complexidade é proporcional a k;
- Logo, o custo de uma sequência de n operações é $O(n \times k)$.
- Podemos prover uma análise mais precisa?

Método agregado: um exemplo

Em uma série de n execuções de INCREMENT:

- $A[0]$ é invertido n vezes;
- $A[1]$ é invertido $n/2$ vezes;
- $A[2]$ é invertido $n/4$ vezes;
- $A[k-1]$ é invertido $n/2^{k-1}$ vezes;

O custo total para n operações é

$$
\begin{aligned}
T(n) &= \sum_{i=1}^{\lfloor \lg n \rfloor} n/2^{i-1} \\
T(n) &< \sum_{i=1}^{\infty} n/2^{i-1} \\
T(n) &< n \times \sum_{i=1}^{\infty} 1/2^{i-1} = 2n
\end{aligned}
$$

Logo o custo amortizado de INCREMENT é $T(n)/n = 2 \in \Theta(1)$: é constante.

Método do contador

Quanto pagar para executar n operações?

- cada operação tem um custo real
- a cada operação é atribuído um preço ($\approx$ custo amortizado individual)
- os preços cobrados devem cobrir os custos reais
- se o preço é maior que o custo:
  - crédito é associado a elementos da estrutura de dados
- se o custo é maior que o preço
  - a diferença deve poder ser paga com os créditos disponíveis
- custo amortizado total: soma dos preços cobrados

Critérios de definição do custo

- o custo amortizado total deve ser uma cota superior do custo real
  - a estrutura de dados não “empresta”
- o crédito total associado aos elementos nunca pode ser negativo

Método do contador: o contador binário

- a unidade de custo é inverter um bit
- preço para $0 \to 1$: 2
  - 1 é o custo
  - sobra $2 - 1 = 1$ de crédito, associado ao bit setado a 1
- preço para $1 \to 0$: 0
  - pago com o crédito

Método do contador: o contador binário

```
INCREMENT(A)
    // A é um arranjo de k bits
    i = 0
    while i < length[A] and A[i] = 1
        A[i] = 0
        i = i + 1
    if i < length[A]
        A[i] = 1
```

- O custo do laço é pago usando os créditos dos bits setados a 1.
- no máximo um único bit é setado a 1: custo 1, mais 1 que fica como crédito
- logo, o custo de uma execução da operação INCREMENT é 2
- o custo de n execuções da operação INCREMENT é $O(n)$.

Método do potencial

- crédito $\Longrightarrow$ potencial para executar futuras operações
- estrutura de dados inicia em um estado D_0
- n operações são executadas,
  - custo real $c_1, \ldots c_i, \ldots c_n$
  - levando aos estados $D_1, \ldots D_i, \ldots D_n$
- A função Φ associa um número real Φ_i (potencial) ao estado D_i
- Custo amortizado $a_i = c_i + \Phi_i - \Phi_{i-1}$
- Custo amortizado total

$$\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} c_i + \Phi_i - \Phi_{i-1} = \sum_{i=1}^{n} c_i + \Phi_n - \Phi_0$$

- se $\forall i \cdot \Phi_i \geq \Phi_0$, então $\sum_{i=1}^{n} a_i$ é uma cota superior do custo total real.

- $\Phi_0 = 0$
- mostrar que $\Phi_i \geq 0, \forall i$.
- $\Phi_i > \Phi_{i-1} \approx$ crédito (potencial aumenta)
- $\Phi_i < \Phi_{i-1} \approx$ débito (potencial diminui)

Método do potencial: o contador binário

- a i-ésima chamada de INCREMENT zera t_i bits
- custo real $c_i = t_i + 1$ (zera i, seta 1)
- potencial: $\Phi_i =$ quantidade de bits setados a 1
- naturalmente, $\forall i \cdot \Phi_i \geq 0$
- logo $\Phi_i \leq \Phi_{i-1} - t_i + 1$
- $\Phi_i - \Phi_{i-1} \leq 1 - t_i$
- custo amortizado de uma operação

$$a_i = c_i + \Phi_i - \Phi_{i-1} \leq (t_i + 1) - (1 - t_i) = 2$$

- custo amortizado de n operações $O(n)$

Arranjos dinâmicos

- contêiner tabela, cuja capacidade adapta-se ao tamanho da coleção
  - inserção e remoção: modelo pilha
- inserção em uma tabela está cheia
  - aloca uma nova tabela de capacidade maior
  - copia o conteúdo da tabela original para a nova tabela
  - libera o espaço ocupado pela tabela original
  - insere o novo elemento
- política de expansão: dobra o tamanho
- remoção, quando o fator de carga fica baixo
  - aloca uma nova tabela de capacidade menor
  - copia o conteúdo da tabela original para a nova tabela
  - libera o espaço ocupado pela tabela original
  - remove o elemento
- política de contração: divide a capacidade por dois quando fator de carga chega a $1/4$.

Implementação: dados

- $T.table$ tabela com os elementos
- $T.num$ quantidade de elementos armazenados na tabela
- $T.size$ capacidade máxima

PPgSC

Inserção

```
INSERT(T, k)
if T.size == 0 // alocação inicial
    T.table = ALLOC-TABLE(1)
    T.size = 1
elseif T.num == T.size // expansão
    tab = T.table
    T.table = ALLOC-TABLE(T.size × 2)
    for i = 1 to T.size
        T.table[i] = tab[i]
    T.size = T.size × 2
    FREE-TABLE(tab)
T.num = T.num + 1 // inserção
T.table[T.num] = k
```

Análise

Cenário: n inserções em uma tabela inicialmente vazia

- custo da operação i
  - sem expansão: 1
  - com expansão: i
- Pior caso $O(n^2)$
- Melhoria desta análise com análise amortizada

Método agregado

- Há expansão quando i é uma potência de 2.
- Custo total:

$$\begin{aligned}
\sum_{i=1}^{n} c_i &\leq n + \sum_{j=0}^{\lfloor \log_2 n \rfloor} 2^j \\
&< n + 2n \\
&= 3n
\end{aligned}$$

Método do contador

- Custo 1 para atribuir uma posição de $T.table$
- Preço de uma operação de inserção: 3
- Pagamento de uma operação de inserção, sem expansão:
  - 1 vai para a inserção (custo real) $i = T.size/2 + j$
  - sobra 2 de crédito
  - 1 crédito na posição de inserção i
  - 1 crédito para um outro elemento da tabela j
- Pagamento da expansão:
  - todos os itens tem um crédito
  - o crédito paga a expansão

Método do potencial

- Após cada expansão: $\Phi = 0$
- $\Phi \uparrow$ quando a tabela é preenchida
- até a tabela ser preenchida, e Φ é gasto na expansão
- $\Phi(T) = 2 \times T.num - T.size$
  - Inicialmente, e após cada expansão $\Phi(T) = 0$
  - Antes de uma expansão $\Phi(T) = 2 \times T.num - T.size = T.size$
  - A cada momento $\Phi(T) \geq 0$
  - Logo, a soma dos custos amortizados é uma cota superior dos custos reais.

Método do potencial

- n_i: número de elementos após operação i
- s_i: capacidade após a operação i
- inicialmente $n_0 = s_0 = 0$, $\Phi_0 = 0$
- $n_i = n_{i-1} + 1$

Método do potencial

se a operação i não tem expansão:

- $s_i = s_{i-1}$ e

$$\begin{aligned} a_i &= c_i + \Phi_i - \Phi_{i-1} \\ &= 1 + (2 \cdot n_i - s_i) + (2 \cdot n_{i-1} - s_{i-1}) \\ &= 1 + (2 \cdot n_i - s_i) + (2 \cdot (n_i - 1) - s_i) \\ &= 3 \end{aligned}$$

PPgSC

Método do potencial

se a operação i tem expansão:

- $s_i = 2 \times s_{i-1}$, $s_{i-1} = n_{i-1} = n_i - 1$ e

$$\begin{aligned}
a_i &= c_i + \Phi_i - \Phi_{i-1} \\
&= n_i + (2 \cdot n_i - s_i) - (2 \cdot n_{i-1} - s_{i-1}) \\
&= n_i + (2 \cdot n_i - 2 \cdot (n_i - 1)) - (2 \cdot (n_i - 1) - (n_i - 1)) \\
&= 3
\end{aligned}$$

Remoção com contração de capacidade

- expansão dobra capacidade
- contração ocorre quando capacidade chega a $1/4$
- por quê não escolher $1/2$?

```
Remove(T)
if T.num == 0
    T.size = 0
    Free-Table(T.table)
elseif T.num ≠ 0 and T.num * 4 < T.size // contração
    tab = T.table
    T.table = Alloc-Table(T.size/2)
    for i = 1 to T.num
        T.table[i] = tab[i]
    T.size = T.size/2
    Free-Table(tab)
T.num = T.num − 1 // remoção
```

Análise com o método do potencial

- $\Phi = 0$ logo após contração ou expansão
- Fator de carga $\alpha(T) = T.num/T.size$, se T não for vazia, 1 se for.
- $T.num = \alpha(T) \cdot T.size$
- Função potencial

$$\Phi(T) \quad = \quad \begin{cases} 2 \cdot T.num - T.size & \text{se } \alpha T \geq 1/2 \\ T.size/2 - T.num & \text{se } 1/4 \leq \alpha T < 1/2 \end{cases}$$

A função potencial nunca é negativa: obteremos uma cota superior do custo real

Análise com o método do potencial

Justificativa intuitiva

- antes de uma expansão:
  - $\alpha(T) = 1$
  - $T.num = T.size$,
  - $\Phi(T) = T.num$
  - permite cópia de $T.num$ elementos.
- antes de uma contração:
  - $\alpha(T) = 1/4$
  - $T.num \cdot 4 = T.size$,
  - $\Phi(T) = T.size/2 - T.num = T.num$
  - permite cópia de $T.num$ elementos.

Análise com o método do potencial

Notações

s_i	:	capacidade após operação i
n_i	:	número de elementos após operação i
Φ_i	:	potencial após operação i
α_i	:	fator de carga após operação i

Inicialmente: $s_0 = n_0 = \Phi_0 = 0, \alpha_0 = 1$

A operação i é uma inserção:

- já fizemos a análise no caso $\alpha_{i-1} \geq 1/2$: $a_i = 3$
- se $\alpha_{i-1} < 1/2$, não há expansão
- se $\alpha_i \geq 1/2$:

$$
\begin{aligned}
a_i &= c_i + \Phi_i - \Phi_{i-1} \\
&= 1 + (2 \cdot n_i - s_i) - (s_{i-1}/2 - n_{i-1}) \\
&= 1 + (2 \cdot (n_{i-1} + 1) - s_{i-1}) - (s_{i-1}/2 - n_{i-1}) \\
&= 3 \cdot n_{i-1} - \frac{3}{2} \cdot s_{i-1} + 3 \\
&= 3 \cdot \alpha_{i-1} \cdot s_{i-1} - \frac{3}{2} \cdot s_{i-1} + 3 \\
&< \frac{3}{2} \cdot s_{i-1} - \frac{3}{2} \cdot s_{i-1} + 3 \\
&= 3
\end{aligned}
$$

A operação i é uma inserção:

- já fizemos a análise no caso $\alpha_{i-1} \geq 1/2$: $a_i = 3$
- se $\alpha_{i-1} < 1/2$, não há expansão
- se $\alpha_i < 1/2$:

$$\begin{aligned} a_i &= c_i + \Phi_i - \Phi_{i-1} \\ &= 1 + (s_i/2 - n_i) - (s_{i-1}/2 - n_{i-1}) \\ &= 1 + (s_i/2 - n_i) - (s_i/2 - (n_i - 1)) \\ &= 0 \end{aligned}$$

O custo amortizado de uma inserção é 3 no máximo.

$$n_i = n_{i-1} - 1$$

- se $\alpha_{i-1} < 1/2$,
  - sem contração: $s_i = s_{i-1}$

$$\begin{aligned} a_i &= c_i + \Phi_i - \Phi_{i-1} \\ &= 1 + (s_i/2 - n_i) - (s_{i-1}/2 - n_{i-1}) \\ &= 1 + (s_i/2 - (n_{i-1} - 1)) - (s_i/2 - n_{i-1}) \\ &= 2 \end{aligned}$$

  - com contração: $n_{i-1} = n_i - 1 = s_i/2 = s_{i-1}/4$ e $c_i = n_i + 1$

$$\begin{aligned} a_i &= c_i + \Phi_i - \Phi_{i-1} \\ &= n_i + 1 + (s_i/2 - n_i) - (s_{i-1}/2 - n_{i-1}) \\ &= n_{i-1} + (n_{i-1} - (n_{i-1} - 1)) - (2 \cdot n_{i-1} - n_{i-1}) \\ &= 1 \end{aligned}$$

$$n_i = n_{i-1} - 1$$

- se $\alpha_{i-1} \geq 1/2$,
- não há contração: $s_i = s_{i-1}$, $n_i = n_{i-1} - 1$
- se $\alpha_i \geq 1/2$

$$\begin{aligned} a_i &= c_i + \Phi_i - \Phi_{i-1} \\ &= 1 + (2 \cdot n_i - s_i) - (2 \cdot n_{i-1} - s_{i-1}) \\ &= 1 + (2 \cdot (n_{i-1} - 1) - s_{i-1}) - (2 \cdot n_{i-1} - s_{i-1}) \\ &= -1 \end{aligned}$$

$$n_i = n_{i-1} - 1$$

- se $\alpha_{i-1} \geq 1/2$,
- não há contração: $s_i = s_{i-1}$, $n_i = n_{i-1} - 1$
- se $\alpha_i < 1/2$

$$\begin{aligned}
a_i &= c_i + \Phi_i - \Phi_{i-1} \\
&= 1 + (s_i/2 - n_i) - (2 \cdot n_{i-1} - s_{i-1}) \\
&= 1 + (s_{i-1}/2 - (n_{i-1} - 1)) - (2 \cdot n_{i-1} - s_{i-1}) \\
&= 2 + \frac{3}{2} \cdot s_{i-1} - 3 \cdot n_{i-1} \\
&= 2 + \frac{3}{2} \cdot s_{i-1} - 3 \cdot \alpha_{i-1} \cdot s_{i-1} \\
&\leq 2
\end{aligned}$$

O custo amortizado de uma remoção é 2 no máximo.

Arranjos dinâmicos

Síntese da análise de complexidade amortizada

- inserção tem custo amortizado $O(1)$
- remoção tem custo amortizado $O(1)$

Arranjos dinâmicos

Exercício

1. Projetar uma estrutura de dados para arranjos dinâmicos, que tem apenas operação de inserção, tal que esta tenha complexidade constante, *no pior caso*.
   - hipótese: supõe-se que o custo de uma alocação dinâmica é $\Theta(1)$.
   - dica: pode inspirar-se na aplicação do método do contador desta aula.
   - obs.: a estrutura deve oferecer o acesso a qualquer posição em $\Theta(1)$.
2. Estender a estrutura projetada com uma operação de remoção do último elemento ($\textsc{Pop-Back}$), tal que tanto a inserção quanto a remoção tenham complexidade constante, no pior caso.