Análise amortizada

Prof. Marcelo Keese Albertini
Universidade Federal de Uberlândia

Análise amortizada

- Distribuir custo de uma operação de alto custo em várias de baixo custo
- Métodos de amortização
  - Método da agregação
  - Método da contabilidade
  - Método de Potenciais (inspirada na Física)
- Limitantes amortizados:

$$\sum_{\text{operação}} \text{“custo amortizado”} \geq \sum_{\text{operação}} \text{“custo real”}$$

Método da agregação: exemplo

```
1 incremento(B[]){
2    i = 0;
3    while (B[i] == 1) {
4       B[i] = 0
5       i = i + 1
6    }
7    B[i] = 1
8 }
```

- Quantos bits são invertido em N operações de incremento?
- No máximo, são invertidos $\lfloor \lg N \rfloor + 1$ bits, que é o número de bits na representação binária de N.
- E na média entre as N operações?
- Método da agregação: somar todos os custos e dividir pelo número de operações

Método da agregação: exemplo

- Melhor caso é 1
- Pior caso é $\lfloor \lg N \rfloor + 1 = \Theta(\lg N)$. Ex: incremento em 0111111 faz 7 inversões
- Para contar de 0 a N são necessárias no máximo $2N$ inversões e não $N \times \Theta(\lg N)$
- Sendo que o n-ésimo bit vai inverter $\left\lfloor \frac{N}{2^n} \right\rfloor$ vezes, temos

$$\sum_{n=0}^{\lfloor \lg \rfloor} \left\lfloor \frac{N}{2^n} \right\rfloor < \sum_{n=0}^{\infty} \frac{N}{2^n} = N \frac{1}{1 - 1/2} = 2N$$

- Assim, o custo amortizado (ou seja, médio) de um incremento em N operações é $O(2)$

Método da agregação: rehashing

- Queremos que $N/M < 1$, então quando $N = M$ dobramos o espaço: $M \leftarrow 2M$
- Custo de inserções agregado em N operações é:

$$\Theta(2^0 + 2^1 + 2^2 + \ldots 2^{\lfloor \lg N \rfloor}) = \Theta(N)$$

- Custo amortizado em N operações é:

$$\frac{\Theta(N)}{N} = \Theta 1$$

Método da agregação: Árvores Binárias de Busca 2-3

- Custo de criar árvore vazia: $c = O(1)$
- Custo por inserção: $i = O(\lg N^*)$ com $N^* > n$ para qualquer número n de chaves na árvore
- Custo por remoção : $d = O(\lg N^*)$
- Custo total:

$$O(c + i \lg N^* + d \lg N^*)$$

- Sabendo que $d < i$, podemos reescrever o custo total como:

$$O(c + 2i \lg N^* + d \cdot 0)$$

- Custo por remoção amortizado (nas inserções) : $d = O(0)$, ou seja, ZERO

Amortização: método da contabilidade

- Usar créditos para operações
- Operações podem usar créditos disponíveis (e saírem de graça)
- Custo amortizado = Custo real + Depósitos (compra de créditos) - Retirada (uso de crédito)

Método da Contabilidade: rehashing

- Na inserção depositar valor $c = O(1) \geq 2$ para pagar o custo de dobrar tamanho da tabela e fazer o rehashing
- O valor c será usado para pagar o hashing atual e o hashing futuro para reposicionar na tabela maior
- Custo amortizado do rehashing: $\Theta(N) - cN/2 = 0$ se c é grande o suficiente

Estruturas de dados com desempenho amortizado

- Tabelas hash com rehashing
- Lazy-weight-balanced tree (Overlars, 1980)
- Scapegoat trees (Anderson, Galpeprin, Rivest, 1993)
- Splay tree (Sleator, Tarzan, 1981)

Árvores Balanceadas por Amortização: Método da Reconstrução global

- Inserção/busca
  - Custo $O(\log n)$ usando algoritmo determinístico de auto-balanceamento
- Remoção: usar marcadores
  - Reconstruir árvore inteira se metade está marcada para remoção
  - Possível reconstruir com $O(n)$ operações de inserção
  - Remoção só custa $O(n)$ após n remoções
  - Algumas remoções custam $O(n)$, mas na maior parte custa $O(\log n)$
  - Amortização: embutir parte de $O(n)$ em cada operação que custa $\log n$: custo amortizado é $O(1 + \log n)$

Árvores Balanceadas por Amortização: Método da Reconstrução Parcial

- Reconstruir subárvore α-desbalanceada
- Reconstruir no retorno da recursão se `altura(v)` $> \alpha \log((v))$ com custo $O(|v|)$
- Assim, altura será $O(\alpha \log n$ então, às vezes, inserção tem custo $\Theta(n)$
- Mas na maior parte das vezes é $O(\log n)$
- Custo amortizado da inserção: é $O(\log n)$

Scapegoat tree

- Inserção e remoção amortizados
- subárvores podem ser reconstruídas em inserções
  - Busca: $O(\log n)$ no pior caso
  - Inserção: $O(\log n)$ amortizada
  - Remoção: $O(\log n)$ amortizada

Splay trees

- Usa ajustes locais
  - zig-zag: 2 rotações seguidas em sentidos distintos
  - roller-coaster: 2 rotações seguidas no mesmo sentido
  - splay: rotacionar até ficar no topo
    - custo $O(\mathit{profundidade}(x))$