Análise Amortizada de Complexidade

Algoritmos e Complexidade

LEI-LCC 2010-2011

MBB

Novembro de 2010

Introdução

Pretende analizar-se uma sequência de operações sobre uma estrutura de dados.

Este é, geralmente o caso mais interessante do ponto de vista aplicacional.

Uma análise que assuma o pior caso para todas as operações será pessimista.

Uma análise probabilística depende muito da informação disponível sobre os inputs.

O objectivo é calcular o custo médio, por operação, numa sequência de operações.

A sequência escolhida será a mais desfavorável, dando origem a uma análise no pior caso.

Uma solução com operações individuais menos eficientes pode apresentar um comportamento amortizado mais favorável.

Exemplo: Stack

Vamos assumir que Push e Pop executam em tempo constante.

Analisemos o tempo de execução de uma sequência de m operações.

Cada operação consiste em 0 ou mais Pop e termina com um Push.

Partindo de stack vazia, podemos concluir duas coisas aparentemente contraditórias.

É possível que uma operação demore $m - 1$ unidades de tempo. Porquê?

No total, o número total de operações é, no máximo, $2m$. Porquê?

Vamos ver várias metodologias equivalentes para tratar este tipo de problema.

Análise Agregada

Trata-se de uma aproximação directa.

Calcula-se o custo médio de uma operação fazendo:

- O cálculo do pior caso do custo total da sequência de operações.
- O quociente desse valor pelo número de operações.

No caso da stack, temos que o número total de Pop nunca pode ser superior ao número de Push.

O número total de Push é simplesmente m, dando um custo total no pior caso de $2m$.

O custo amortizado de cada operação é 2.

Visão do banqueiro: “Accounting Argument”

Assumimos que efectuar operações exige crédito.

A cada operação atribuímos um determinado crédito.

Este crédito poderá ser inferior ao que a operação necessitaria individualmente.

Tentaremos demonstrar que a soma do crédito de todas as operações permite executar a sequência completa.

Para isso, usaremos créditos não utilizados de operações anteriores.

Poderemos também necessitar de pedir créditos emprestados, mas não iremos abordar este caso mais complicado.

Visão do banqueiro: “Accounting Argument”

Se provarmos que nunca precisaremos de pedir emprestado, a análise é simples mesmo para sub-sequências: no pior caso será a soma dos custos individuais amortizados.

No caso da stack, alocamos dois créditos por operação e observamos que o número de créditos poupados é sempre igual ao número de elementos armazenados.

Um Pop utiliza créditos poupados.

Um Push utiliza um crédito poupado, e assinala o fim da operação: mais 2 créditos que podem ser poupados.

Nunca precisamos de pedir emprestado: custo total $2m$.

Visão do Físico: “Potential Argument”

Define-se uma função *potencial* Φ, que mapeia uma configuração D da estrutura de dados num número real.

Φ representa a potencial carga computacional de processar D.

Define-se o *tempo amortizado a* de uma operação como $t + \Phi(D') - \Phi(D)$:

- t é o tempo real que a operação leva a executar.
- D é a configuração antes de a operação executar.
- D' é a configuração depois de a operação executar.

Para qualquer sequência de m operações, teremos

$$\sum_{i=1}^{m} t_i = \sum_{i=1}^{m} a_i - \Phi_i + \Phi_{i-1} = \Phi_0 - \Phi_m + \sum_{i=1}^{m} a_i$$

Visão do Físico: “Potential Argument”

O tempo total é a soma dos custos amortizados, afectada da alteração no potencial da configuração da estrutura de dados.

Espera-se que uma operação muito pesada faça decrescer o potencial, permitindo operações seguintes mais eficientes.

Podemos escolher qualquer função potencial.

Nos casos mais interessantes Φ_0 será nulo, e o potencial será sempre positivo.

Voltando ao exemplo da stack, podemos tomar Φ como o número de elementos armazenados.

Neste caso, partindo de um potencial de i, sendo k o número de Pop realizados, temos o custo amortizado de cada operação:

$$(k+1)+(i-k+1)-i=2$$

Se partirmos e chegarmos a uma stack vazia, temos custo total $2m$.

Gestão de Listas

Análise da heurística “Move-to-Front”:

- Consideramos tabela de n elementos e a operação de aceder a um desses elementos.
- A implementação é ligada, e o tempo de acesso ao elemento na posição i é igual a i.
- A lista não está ordenada, mas pode ser re-arranjada entre operações de acesso.
- O re-arranjo pode ser feito trocando um elemento com outro adjacente, o que pode ser feito em tempo constante.

O problema: vale a pena fazer swaps entre operações de acesso?

Esta pergunta só faz sentido se alguns elementos forem acedidos mais vezes que outros.

A heurística “Move-to-Front” coloca o elemento acedido na cabeça da lista.

Gestão de Listas

Esta técnica tão simples é surpreendentemente eficiente:

- Fica a um factor constante de qualquer outra estratégia que implique conhecimento de estatísticas de acesso.
- O mesmo se aplica relativamente a algoritmos que assumem ser possível fazer swaps arbitrários em tempo constante.
- É uma estratégia óptima num sentido muito abrangente!

Adoptamos a visão do físico:

- Assumimos um algoritmo arbitrário A a executar em paralelo com esta estratégia.
- O potencial é definido como $\Phi(D) = 2 * \sum_{x \in D} \Phi_x$, com

$$\Phi_x = \#\{z | z \text{ antes de } x \text{ em } D, z \text{ depois de } x \text{ em } A\}$$

- As configurações são inicialmente iguais.
- Considera-se uma sequência arbitrária de acessos.

Gestão de Listas

Vamos demonstrar o resultado conhecido de que o custo amortizado de uma operação de acesso é, no máximo, 4 vezes superior ao de A.

O custo amortizado, partindo de potencial P, de um acesso à posição i em D é

$$2i + (2(i - 1 - 2x_i) + P) - P = 4i - 2 - 4x_i$$

em que x_i é o número de antecessores do elemento i em D, que aparecem depois dele em A.

Um algoritmo optimizado para aquela sequência de acessos nunca precisará de fazer swaps: tempo de acesso j.

Como $i - x_i \leq j$, temos um custo amortizado máximizado por $4j$.

O potencial final é superior ou igual a zero, pelo que o custo total é O(4*m*N) e está a um factor de 4 do método óptimo!

A razão entre custos é inferior a 4: **análise competitiva**.