MAC 5710 - Estruturas de Dados e sua Manipulação

URL: http://www.ime.usp.br/∼song/mac5710.html

Eficiência ou complexidade de algoritmos

Siang - 2005

(Daremos aqui um tratamento informal sobre o assunto. Maiores detalhes devem ser obtidos em cursos de Análise de Algoritmos.)

Costuma-se analisar um algoritmo em termos de tempo e espaço (ou memória). Para o tempo, diversas medidas são em princípio possíveis:

1. Tempo absoluto (em minutos, segundos, etc.) – não é interessante por depender da máquina.
2. Número de operações consideradas relevantes (por exemplo comparações, operações aritméticas, movimento de dados, etc.)

Três casos podem ser considerados: o melhor caso, o caso médio e o pior caso. O estudo do caso médio depende do conhecimento ou suposição da distribuição das entradas ao problema. Em geral o pior caso é o mais fácil de se analisar.

Por exemplo, se temos 2 métodos para resolver um mesmo problema, cuja entrada tem tamanho n, podemos ter:

Método 1: $T_1(n) = 2n^2 + 5n$ operações
Método 2: $T_2(n) = 500n + 4000$ operações

Dependendo do valor de n, método 1 pode requerer mais ou menos operações que método 2.

Notação O

Para estudarmos o comportamento assintótico – para n grande – interessa-nos o termo de ordem superior:

Método 1: $T_1(n) = O(n^2)$
Método 1: $T_2(n) = O(n)$

A notação O pode ser formalizada como se segue:
Dizemos que $T(n) = O(f(n))$ se existirem inteiro m e constante c tais que

$$T(n) \leq cf(n) \quad \text{para} \quad n > m.$$

Isto é, para valores de n suficientemente grandes, $T(n)$ não cresece mais rapidamente que $f(n)$.

A importância da complexidade pode ser observada pelo seguinte exemplo, que mostra 5 algoritmos A_1 a A_5 para resolver um mesmo problema, de complexidades diferentes. Supomos que uma operação leva 1 ms para ser efetuada. A tabela seguinte dá o tempo necessário por cada um dos algoritmos. (Sempre suporemos logaritmos na base 2.)

	A_1	A_2	A_3	A_4	A_5
n	$T_1(n) = n$	$T_2(n) = n \log n$	$T_3(n) = n^2$	$T_4(n) = n^3$	$T_5(n) = 2^n$
16	0.016s	0.064s	0.256s	4s	1m 4s
32	0.032s	0.16s	1s	33s	46 dias
512	0.512s	9s	4m 22s	1 dia 13h	10^{137} séculos

Podemos muitas vezes melhorar o tempo de execução de um programa (que implementa um algoritmo dado) fazendo otimizações locais (e.g. usar $x + x$ ao invés de $2x$, em máquinas em que a multiplicação é mais demorada que a adição).

Entretanto, melhorias substanciais são muitas vezes obtidas se usarmos um algoritmo diferente, com outra complexidade de tempo, e.g. obter um algoritmo de $O(n \log n)$ ao invés de $O(n^2)$.

Cota superior ou limite superior ("upper bound")

Seja dado um problema, por exemplo, multiplicação de duas matrizes quadradas de ordem n. Conhecemos um algoritmo para resolver este problema (pelo método trivial) de complexidade $O(n^3)$. Sabemos assim que a complexidade deste problema não deve superar $O(n^3)$, uma vez que existe um algoritmo desta complexidade que o resolve. Uma cota superior (ou "upper bound") deste problema é $O(n^3)$. A cota superior de um problema pode mudar se alguém descobrir um outro algoritmo melhor. Isso de fato aconteceu com o algoritmo de Strassen que é de $O(n^{\log 7})$. Assim a cota superior do problema de multiplicação de matrizes passou a ser $O(n^{\log 7})$. Outros pesquisadores melhoraram ainda este resultado. Atualmente o melhor resultado é o de Coppersmith e Winograd – de $O(n^{2.376})$. Notem que a cota superior de um problema depende do algoritmo.

A cota superior de um problema é parecida com o *record mundial* de uma modalidade de atletismo. Ela é estabelecida pelo melhor atleta (algoritmo) do momento. Assim como o *record mundial*, a cota superior pode ser melhorada por um algoritmo (atleta) mais veloz.

"Cota Superior" dos 100 metros rasos:

Ano	Atleta (Algoritmo)	Tempo
1988	Carl Lewis	9s92
1993	Linford Christie	9s87
1993	Carl Lewis	9s86
1994	Leroy Burrell	9s84
1996	Donovan Bailey	9s84
1999	Maurice Greene	9s79
2002	Tim Montgomery	9s78

Cota inferior ou limite inferior ("lower bound")

As vezes é possível demonstrar que para um dado problema, qualquer que seja o algoritmo a ser usado, o problema requer pelo menos um certo número de operações. Essa complexidade é chamada cota inferior ou "lower bound" do problema. Veja que a cota inferior depende do problema mas não do particular algoritmo. Usamos a letra Ω em lugar de O para denotar uma cota inferior.

Para o problema de multiplicação de matrizes de ordem n, apenas para ler os elementos das duas matrizes de entrada leva tempo $O(n^2)$. Assim uma cota inferior trivial é $\Omega(n^2)$.

Na analogia anterior, uma cota inferior de uma modalidade de atletismo não dependeria mais do atleta. Seria algum tempo mínimo que a modalidade exige, qualquer que seja o atleta. Uma cota inferior trivial para os 100 metros rasos seria o tempo que a velocidade da luz leva para percorrer 100 metros no vácuo.

Se um algoritmo tem uma complexidade que é igual a cota inferior do problema, então ele é ótimo. O algoritmo de Coppersmith e Winograd é de $O(n^{2.376})$ mas a cota inferior é de $\Omega(n^2)$. Portanto não podemos dizer que ele é ótimo. Pode ser que esta cota superior possa ainda ser melhorada. Pode também ser que a cota inferior de $\Omega(n^2)$ possa ser melhorada (isto é "aumentada"). Muitos pesquisadores desta área dedicam seu tempo tentando encurtar este intervalo ("gap") até encostar uma da outra.