Notas sobre minorantes de complexidade

Fevereiro 2007

Armando B. Matos

Neste trabalho faremos uma introdução às aplicações da teoria da Informação aos problemas de complexidade mínima, isto é, à procura de *minorantes* da complexidade da classe dos algoritmos que resolvem um determinado problema.
Seja P um problema dado e seja $\mathsf{f}(\mathfrak{n})$ a complexidade (por exemplo, em termos de ordem de grandeza) do algoritmo mais eficiente A que resolve esse problema (eventualmente pode haver mais do que um); chamemos a $\mathsf{f}(\mathfrak{n})$ a complexidade óptima do problema. Normalmente o algoritmo óptimo A é desconhecido mas, tanto do ponto de vista teórico como do prático, é importante conhecer o mais exactamente possível — isto é, limitar a um intervalo tão pequeno quanto possível — a complexidade óptima $\mathsf{f}(\mathfrak{n})$.
A complexidade de qualquer algoritmo que solucione P é obviamente um majorante da complexidade óptima.
Determinar minorantes de $\mathsf{f}(\mathfrak{n})$ é muito mais difícil uma vez que, para se mostrar que $\mathsf{g}(\mathfrak{n})$ é um minorante de $\mathsf{f}(\mathfrak{n})$ é necessário considerar *todos* os algoritmos para P e mostrar que nenhum deles tem uma eficiência melhor do que $\mathsf{f}(\mathfrak{n})$; como o número desses algoritmos é infinito, um ataque directo à determinação de minorantes está fora de questão.
A Teoria da Informação pode ser utilizada para a determinação de minorantes de complexidade. A ideia básica, em linhas gerais é a seguinte. Para que um algoritmo consiga solucionar todas as "instâncias" de um determinado comprimento é necessário que obtenha, da "instância" em questão, informação suficiente para poder determinar a solução correcta. Por outras palavras, o facto de se "seleccionar" deterministicamente, de entre todas as soluções possíveis, a solução correcta, obriga a uma obtenção de informação da "instância" que representa, por si só, uma tarefa que qualquer algoritmo que resolva o problema dado tem que executar.
Neste trabalho discutiremos primeiro problemas em que a informação é obtida por pesagens em balanças de pratos. Seguidamente consideraremos a comparação entre valores numéricos como fonte de informação em problemas como a determinação de máximo, o "merge" de 2 sequências e a ordenação.

1 Um problema

Existem 3 bolas, sendo 2 de peso igual e uma mais pesada; podemos fazer apenas uma "pesagem" numa balança de pratos que nos dá 3 resultados possíveis: ou cai para a esquerda ou para a direita ou fica equilibrada. Qual a bola mais pesada? A resposta é simples: comparemos com a balança o peso da bola 1 com a bola 2:

- Se forem iguais, a bola 3 é a mais pesada de todas.
- Se a bola 1 pesa mais que a 2, então a bola 1 é a mais pesada de todas.
- Se a bola 2 pesa mais que a 1, então a bola 2 é a mais pesada de todas.

Neste caso, o número de soluções que o problema pode ter é 3 (qualquer uma das bolas pode ser a mais pesada) e o número de resultados possíveis das pesagens é também 3. Em geral podemos enunciar o seguinte Teorema.

Teorema 1 *Para que seja sempre possível chegar à solução é condição necessária que* $a \leq b$ *onde* a *representa o número de soluções do problema e* b *representa o número de casos que é possível distinguir com a informação obtida pelo algoritmo.*

Se são permitidas k pesagens, o número de resultados possíveis das pesagens é 3^k.

> **Problema**
> Existem 14 bolas, todas iguais, com excepção de uma que pode ser mais ou menos pesada que as outras; com 3 pesagens determinar qual a diferente e se é mais leve ou pesada que as outras.

Neste caso $a = 28$ pois qualquer uma das 14 bolas pode ser a mais leve ou a mais pesada e $b = 3^3 = 27$; como $a > b$, podemos afirmar que não existe solução!

> **Problema**
> Existem 5 bolas todas de pesos diferentes; determinar, com um máximo de 4 pesagens, a sequência das bolas ordenadas pelos seus pesos.

Temos $a = 5! = 120$, (todas as permutações das 5 bolas) e $b = 3^4 = 81$; mais uma vez o problema não tem solução. Qual o número mínimo de pesagens para que o problema *possa* ter solução? Admitamos que os pesos são tais que a balança nunca fica equilibrada (o que é consistente com o facto de os pesos serem todos diferentes). Cada pesagem só dá origem a 2 decisões possíveis. Assim temos que p, o número de pesagens, deve satisfazer $2^p \geq 120$, ou seja

$$p \geq \lceil \log 120 \rceil = 7$$

2 As 12 bolas

O Teorema 1 anterior não só é útil para determinar casos de impossibilidade mas pode ser também usado para seleccionar as pesagens que podem constituir uma solução. Vejamos um problema com solução e procuremo-la.

> **Problema**
> Existem 12 bolas, todas iguais, excepto uma (que pode ser mais ou menos pesada que as outras); com 3 pesagens determinar a diferente e se é mais leve ou pesada.

Neste caso $a = 24$ e $b = 3^3 = 27$ pelo que é $a \leq b$ e o problema poderá ter ou não solução; o Teorema 1 só exprime uma condição necessária para haver solução.
Procuremos uma possível solução. Utilizaremos apenas pesagens com igual número de bolas em cada prato. Se, por exemplo, colocassemos 4 bolas no prato da esquerda e 3 no da direita, e a balança a caísse para a esquerda, não tiravamos qualquer conclusão: qualquer uma destas 7 bolas podia ser a mais pesada, a mais leve, ou podiam ser todas iguais!

Primeira pesagem

Temos $a = 24$ e $b = 27$; depois da primeira pesagem ficará $b' = 9$ onde b' representa o valor de b para o problema subsequente. Se colocassemos 1 bola em cada prato e a balança a ficasse equilibrada, restavam 20 hipóteses (10 bolas) para serem decididas em 2 pesagens; como $20 > 9$, isso seria impossível, pelo Teorema 1. A mesma conclusão se tira se usarmos 2 (pois $16 > 9$) ou 3 (pois $12 > 9$) bolas em cada prato. Se usarmos 5 bolas em cada prato e a balança a cair para a esquerda, teríamos que decidir uma de 10 hipóteses (uma das 5 bolas da esquerda pode ser mais a pesada, ou uma das 5 bolas da direita pode ser a mais leve) com 2 pesagens; ora $10 > 9$ e isso é impossível; com 6 bolas em cada prato ainda é pior. Conclusão: teremos que colocar 4 bolas em cada prato, seja $(1, 2, 3, 4) - (5, 6, 7, 8)$. Temos a considerar os casos

1. Fica equilibrado
   Uma das restantes 4 bolas, 9, 10, 11 ou 12 é a diferente; temos 8 hipóteses e $8 < 9$, tudo bem, só pode haver um resultado de pesagem desperdiçado. Façamos seguidamente a pesagem $(9, 10, 11) - (1, 2, 3)$, notando que 1, 2 e 3 são bolas iguais.

   (a) Se cai para a esquerda, ou 9 ou 10 ou 11 é a diferente e é mais pesada ($3 \leq 3$); com uma nova pesagem entre 9 e 10 a situação fica resolvida.

   (b) Se cai para a direita é análogo: ou 9 ou 10 ou 11 é a diferente e é mais leve ($3 \leq 3$); com uma nova pesagem entre 9 e 10 a situação fica resolvida.

   (c) Se fica equilibrada, a bola 12 é diferente; comparada com, por exemplo, a bola 1, determinamos se é mais leve ou mais pesada (aqui a balança a não pode ficar equilibrada: é o resultado desperdiçado).

2. Cai para a esquerda
   Há 8 hipóteses: ou 1 ou 2 ou 3 ou 4 é a mais pesada; ou 5 ou 6 ou 7 ou 8 é a mais leve ($8 \leq 9$) Façamos a pesagem $(1, 2, 5) - (3, 4, 6)$.

   (a) Cai para a esquerda: 1 ou 2 é a mais pesada ou 6 é a mais leve; uma nova pesagem decide a situação $(1) - (2)$.

   (b) Cai para a direita; análogo: 3 ou 4 é a mais pesada ou 5 é a mais leve; fazemos a pesagem $(3) - (4)$.

   (c) Fica equilibrada: a mais leve é 7 ou 8; pesemos $(7) - (8)$ (não pode dar igual).

3. Cai para a direita
   Raciocínio análogo ao caso 2.

Exercício 1
O problema análogo com 13 bolas é insolúvel (por isso é que se diz que 13 dá azar). Provar esse facto. ◇

3 Entropia e informação

Se existem n hipóteses para uma situação e p_i é a probabilidade de o facto i ser verdadeiro a entropia deste estado de conhecimento é o seguinte valor que nunca é negativo

$$S = -\sum_{i=1}^{n} p_i \log p_i$$

Esta entropia mede a falta de informação ou incerteza deste estado de conhecimento; admitimos que os logaritmos são na base 2 embora isso não seja importante aqui. Uma entropia grande significa que é necessária uma grande quantidade de informação para descrever a situação. Uma entropia nula quer basicamente dizer que o conhecimento da situação é completo.
Vejamos em 2 situações o valor tomado pela entropia.

- Se temos a certeza que a hipótese 1 é válida, $p_1 = 1$ e $p_i = 0$ para $i \geq 2$. Resulta que o valor limite de S quando todos os p_i, com $i \geq 2$, tendem para 0 (e, consequentemente, p_1 tende para 1)

  $$S = 1 \times \log 1 = 0$$

  levando em consideração que $\lim_{p \to 0} p \log p = 0$.

- Se qualquer das n hipóteses é igualmente provável, $p_i = 1/n$ para todos os $1 \leq i \leq n$. Temos pois

  $$S = -n \frac{1}{n} \log \frac{1}{n} = -\log \frac{1}{n} = \log n$$

  Por exemplo, se $n = 16$, vem $S = 4$; a entropia (usando o logaritmo na base 2) mede, o número de bits de informação necessários para especificar completamente uma das hipóteses.

A fórmula da entropia não "caiu do céu". Shannon em *the Mathematical Theory of Information* demonstra que a *única* função $S(p_1, \cdots, p_n)$ que satisfaz os seguintes pressupostos razoáveis

- S é contínua em todos os p_i.
- Se todos os $p_i = 1/n$, a função $S(n)$ deverá crescer monotonicamente com n. Mais hipóteses (=acontecimentos) representam mais incerteza.
- Se uma escolha se desdobrar em duas escolhas sucessivas, a função ao original S deve ser a soma pesada dos S de cada escolha (ver o trabalho referido de Shannon para mais pormenores).

é do tipo

$$S = -k \sum_{i=1}^{n} p_i \log p_i$$

onde $k > 0$.

4 Informação e pesagens

Uma pesagem do tipo considerado pode ter 3 resultados; seja:

n o número de hipóteses possíveis antes da pesagem.

n_1, n_2 e n_3 os números de hipóteses que restam após cada pesagem.

Supondo probabilidades iguais para as situações possíveis, a entropia é diminuida de, respectivamente, $\log(n/n_1)$, $\log(n/n_2)$ e $\log(n/n_3)$. Ora, como pretendemos limitar o número máximo de pesagens em todas as situações, a situação mais favorável seria quando a divisão fosse disjunta (em hipóteses) e igual, isto é $n_1 = n_2 = n_3 = n/3$. Nesta hipótese a pesagem daria um "trit" de informação:

$$\text{Um trit} = \log_2 3 \text{ bits} \approx 1.585 \text{ bits}$$

Em termos de informação expressa em trits, a possível solução do problema das 12 bolas (ver o teorema1) resulta da desigualdade:

$$3 \text{ trits} \geq \log_3 24 \text{trits} \approx 2.893 \text{ trits}$$

Nota: Nota: a desigualdade $a \leq b$ deve ser válida em todos os nós da árvore.

Exercício 2

Construir a árvore da solução dada para as 12 bolas, representando em cada nó:

- *A pesagem efectuada.*
- *A entropia (antes da pesagem) expressa em trits.*
- *A entropia máxima, dado o número que falta de pesagens (0, 1, 2 ou 3).*

◇

5 Informação e Comparações

5.1 Ordenação

Vejamos agora um problema que tem a ver com os importantes algoritmos de ordenação.

Problema (Ordenação)
Sejam n bolas de pesos diferentes; pretende-se colocá-las por ordem crescente de pesos usando apenas operações de comparação entre 2 bolas, isto é, pesagens com uma bola num prato e outra noutro. Note-se que a balança a nunca fica equilibrada uma vez que não existem 2 bolas de igual peso.

Aplicando directamente o Teorema 1 temos.

O número de hipóteses possíveis é $a = n!$.

O número de situações que se podem discriminar com c comparações é $b = 2^c$.

Resulta que qualquer algoritmo de ordenação que obtenha a informação através de comparações deve satisfazer $a \leq b$, ou seja,

$$2^c \geq n!$$

ou ainda

$$c \geq \lceil \log(n!) \rceil$$

Exercício 3

Determine minorantes de c para os primeiros valores de n (por exemplo entre 1 e 6) e procure algoritmos de ordenação que se aproximem tanto quanto possível desses valores (no caso mais desfavorável). ◇

Usando a fórmula de Stirling como aproximação para o factorial

$$n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$$

temos (podiamos ser mais rigorosos; o sinal $\approx$ pode ser substituido por $>$ e este facto pode ser utilizado no que se segue).

$$2^c \geq \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$$

ou seja (os logaritmos são, como é usual, na base 2)

$$c \geq \frac{1}{2}\log(2\pi n) + n\log n - n\log e$$

Vemos pois que qualquer algoritmo de ordenação baseado em comparações deve fazer (no pior caso) pelo menos cerca de $n \log n$ comparações, isto é deve ser pelo menos de ordem $O(n \log n)$.

Exercício 4

Considere o algoritmo de ordenação "mergesort". Compare para $n = 1, 2, 4, \ldots, 1024$ o minorante teórico do número de comparações ($\lceil \log(n!) \rceil$) com o majorante número de comparações calculado para o algoritmo (ver apontamentos teóricos de Análise de Algoritmos). ◇

Concluimos pois que, em termos de ordens de grandeza (a afirmação não é válida se considerarmos os valores exactos do número de comparações), são conhecidos algoritmos de ordenação óptimos. Um exemplo é o "heapsort"; o "quicksort" não é óptimo, uma vez que as nossas conclusões são em termos do comportamento no pior caso (para o qual o "quicksort" é $O(n^2)$).

O teorema 1 dá para cada problema um limite mínimo da complexidade dos algoritmos respectivos; pode ser expresso, por outras palavras, da forma seguinte: o algoritmo tem que

obter pelo menos tanta informação quanta é necessária para distinguir o caso particular do problema em questão. Para certos problemas — como o da ordenação — existem algoritmos próximos deste limite teórico; para outros, os melhores (teoricamente possiveis) algoritmos estão muito longe dele e há que recorrer a outros métodos, em geral mais complicados, para obter melhores minorantes.

5.2 Máximo

Um outro problema, mais simples que o da ordenação é o seguinte.

Problema (Máximo)
Determinar o maior de n valores distintos.

Neste caso a aplicação do Teorema fornece um resultado bem fraco: são precisas pelo menos $\log n$ comparações (o número de soluções possí veis é n); ora não é difícil mostrar que são necessárias $n-1$ comparações!

Exercício 5
Mostre que qualquer algoritmo baseado em comparações para a determinação do máximo entre n elementos tem de efectuar, pelo menos, $n-1$ comparações. ◇

O problema de não se conseguir fazer melhor pode, de certo modo, ser explicado pelo facto de não ser possível eliminar cerca de $n/2$ elementos em cada comparação; de facto, no algoritmo usual — o maior dos elementos encontrados até à altura é comparado com o elemento corrente — fazem-se $n-1$ comparações, eliminando apenas um candidato em cada comparação.

A situação é semelhante para a procura do elemento de ordem $\lfloor n/2 \rfloor$ (a meio da tabela). Aqui existem algoritmos que fazem cerca de $3n$ comparações.

Vejamos contudo duas aplicações mais "fortes" do teorema:

Exercício 6
Temos n bolas; uma pesa 1, outra 2,... e a outra 2^{n-1} Mostre que é possível determinar a mais pesada em não mais de cerca de $\log n$ pesagens (aproximando-se do limite dado pelo teorema). ◇

Exercício 7
Analise a complexidade mínima do seguinte problema. Existem n elementos distintos v_1, v_2,...,v_n ordenados por ordem crescente; é dado um valor x e pergunta-se: qual o menor i tal que $v_i < x$ (sendo $i = 0$ se x é menor que todos os elementos)? Por outras palavras, onde deveria x ser colocado na lista dos elementos?

Notas sobre a solução

Aqui existem $n+1$ respostas possíveis; o teorema diz-nos que o número de comparações deve satisfazer: $2^c \geq n+1$, ou seja $c \geq \lceil \log(n+1) \rceil$.

Ora existe um algoritmo — o da pesquisa binária — que faz exactamente este número de comparações; não pode haver melhor! ◇

Exercício 8
Em cada um dos casos seguintes pergunta-se quantas comparações são necessárias pelo

teorema? E na realidade? Qual a entropia da situação inicial?

1. *Eu "penso" num inteiro entre 1 e 1000; tu tentas adivinhar qual é, tentando de cada vez um número; eu respondo: é maior, menor ou é esse.*

2. *Existem n inteiros entre 1 e n; determinar com comparações o seu valor.*

◇

5.3 "Merge"

Consideremos agora o algoritmo do "merge".

Problema ("Merge")
São dadas 2 sequências de ordenadas de inteiros, uma com n e a outra com m elementos. Pretende-se obter a sequência ordenada constituída pelos $m + n$ elementos.

Um algoritmo de resolver este problema é baseado na aplicação iterativa do seguinte passo. Os dois menores elementos das sequências dadas são comparados; o menor deles é retirado e colocado na sequência resultado. Quando uma das sequências dadas "chegar ao fim", o resto da outra é "passado" para a sequência resultado. Supondo todos os elementos distintos, é fácil ver que são feitas no máximo $m + n - 1$ comparações.
Vejamos se este algoritmo se aproxima do limite teórico; o número de situações possíveis diferentes entre as duas sequências é:

$$\binom{m+n}{n}$$

Este número, igual a $(m+n)!/(n!m!)$, pode ser aproximado usando a fórmula de Stirling; vamos supor apenas o caso $m = n$.

$$\binom{2n}{n} = \frac{(2n)!}{(n!)^2} \approx (\pi n)^{-1/2} 2^{2n}$$

Sendo c o número de comparações efectuadas, deverá ser pois:

$$2^c \geq \lceil (\pi n)^{-1/2} 2^{2n} \rceil$$

ou seja:

$$c \geq \left\lceil 2n - \frac{1}{2}\log(\pi n) \right\rceil$$

Ora, como dissemos, o algoritmo do "merge" faz neste caso $2n - 1$ comparações na pior hipótese. Vemos que, para n grande, o algoritmo é quase óptimo. Por exemplo, para $n = 100$, o algoritmo faz 199 comparações, enquanto o minorante (confiando na aritmética do yap) é $\lceil 195.852 \rceil = 196$.