CI165 — Cotas Inferiores - Ordenação e Busca

André Vignatti

24 de outubro de 2011

O problema da ordenação - cota inferior

- Estudamos alguns algoritmos de ordenação.
- Eles têm algo em comum: usam somente comparações entre dois elementos do vetor para definir a posição relativa desses elementos.
- Isto é, o resultado de comparar x_i com x_j, define se x_i será posicionado antes ou depois de x_j.
- O MergeSort e o HeapSort executam $\Theta(n \log n)$ comparações no pior caso.

O problema da ordenação - cota inferior

- Será que é possível projetar um algoritmo de ordenação baseado em comparações ainda mais eficiente?
- Veremos a seguir que não!
- É possível provar que qualquer algoritmo que ordena n elementos baseado apenas em comparações de elementos efetua **no mínimo** $\Omega(n \log n)$ comparações no pior caso.
- Para demonstrar isso, vamos representar os algoritmos de ordenação em um modelo computacional abstrato, denominado árvore (binária) de decisão.

Árvores de Decisão - Modelo Abstrato

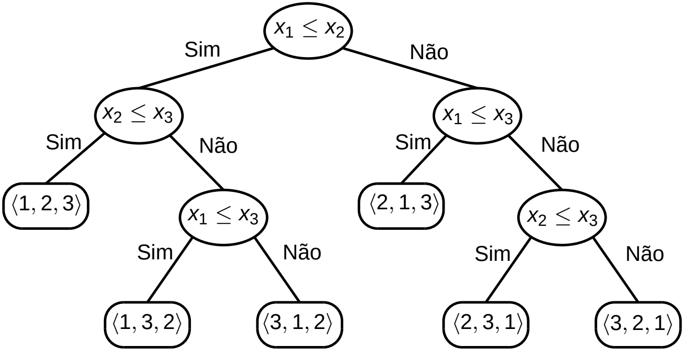
- Os nós internos de uma árvore de decisão representam comparações feitas pelo algoritmo.
- As subárvores de cada nó interno representam possibilidades de continuidade das ações do algoritmo após a comparação.
- No caso das árvores binárias de decisão, cada nó possui apenas duas subárvores. Tipicamente, as duas subárvores representam os caminhos a serem seguidos conforme o resultado (verdadeiro ou falso) da comparação efetuada.
- As folhas são as respostas possíveis do algoritmo após as decisões tomadas ao longo dos caminhos da raiz até as folhas.

Árvores de decisão para o problema da ordenação

É possível representar um algoritmo para o problema da ordenação através de uma árvore de decisão:

- Os nós internos representam comparações entre dois elementos, digamos $x_i \leq x_j$.
- As ramificações representam os possíveis resultados da comparação: verdadeiro se $x_i \leq x_j$, ou falso se $x_i > x_j$.
- As folhas representam possíveis soluções: as diferentes permutações dos n índices.

Esta a árvore de decisão que representa o comportamento do *Insertion Sort* para 3 elementos:



Árvores de decisão para o problema da ordenação

- Ao representarmos um algoritmo de ordenação qualquer baseado em comparações por uma árvore binária de decisão, todas as permutações de n elementos devem ser possíveis soluções.
- Assim, a árvore binária de decisão deve ter $\geq n!$ folhas, podendo ter mais (nada impede que duas seqüências distintas de decisões terminem no mesmo resultado).
- O caminho mais longo da raiz a uma folha representa o pior caso de execução do algoritmo.
- A altura mínima de uma árvore binária de decisão com pelo menos $n!$ folhas fornece o número mínimo de comparações que o melhor algoritmo de ordenação baseado em comparações deve efetuar.

Cota inferior

- Qual a altura mínima, em função de n, de uma árvore binária de decisão com pelo menos $n!$ folhas?
- Uma árvore binária de decisão T com altura h tem, no máximo, 2^h folhas.
- Portanto, se T tem pelo menos $n!$ folhas, então $n! \leq 2^h$, ou seja, $h \geq \log_2 n!$.
- Mas,

$$\begin{aligned}
\log_2 n! &= \textstyle\sum_{i=1}^{n} \log i \\
&\geq \textstyle\sum_{i=\lceil n/2 \rceil}^{n} \log i \\
&\geq \textstyle\sum_{i=\lceil n/2 \rceil}^{n} \log n/2 \\
&\geq (n/2 - 1) \log n/2 \\
&= n/2 \log n - n/2 - \log n + 1 \\
&\geq n/4 \log n, \ \text{para } n \geq 16.
\end{aligned}$$

- Então, $h \in \Omega(n \log n)$.

Conclusão

- Provamos então que $\Omega(n \log n)$ é uma cota inferior para o problema da ordenação.
- Portanto, os algoritmos *Mergesort* e *Heapsort* são algoritmos ótimos.
- Existem algoritmos lineares para ordenação, ou seja, que têm complexidade $O(n)$. (Como???)

Algoritmos de ordenação NÃO baseados em comparações devem **assumir algum propriedade sobre a entrada**:

- COUNTING SORT: assume que cada um dos n elementos é um inteiro entre 0 e $k \Rightarrow \Theta(n+k)$.
- RADIX SORT: assume que cada dígito dos n elementos assume d valores distintos $\Rightarrow \Theta(d(n+k))$, com $k =$ número de elementos distintos.
- BUCKET SORT: assume que os n números estão distribuídos de acordo com a distribuição uniforme $\Rightarrow$ tempo **esperado** $\Theta(n)$.

Cotas inferiores de problemas

- Em geral é muito difícil provar cotas inferiores não triviais de um problema.

Um problema com entrada de tamanho n tem como cota inferior trivial $\Omega(n)$ (Se tivermos que ler toda a entrada).

- São pouquíssimos problemas para os quais se conhece uma cota inferior que coincide com a cota superior (i.e., um algoritmo).

Busca em vetor ordenado

Dado um vetor crescente $A[p \dots r]$ e um elemento x, devolver um índice i tal que $A[i] = x$ ou -1 se tal índice não existe.

Busca-Binária(A, p, r, x)

  se $p \le r$
      então $q \leftarrow \lfloor (p + r)/2 \rfloor$
          se $A[q] > x$
            então **devolva** Busca-Binária$(A, p, q - 1, x)$
          se $A[q] < x$
            então **devolva** Busca-Binária$(A, q + 1, r, x)$
          devolva q $\triangleright A[q] = x$
      senão
        devolva -1

Número de comparações: $O(\lg n)$.

- É possível projetar um algoritmo mais rápido?

Não, se o algoritmo baseia-se em comparações do tipo $A[i] < x$, $A[i] > x$ ou $A[i] = x$.

- A cota inferior de comparações para o problema da busca em vetor ordenado é $\Omega(\lg n)$.
- Pode-se provar isso usando o modelo de árvore de decisão.

Cota inferior

Todo algoritmo para o problema da busca em vetor ordenado baseado em comparações pode ser representado através de uma árvore de decisão.

- Cada nó interno corresponde a uma comparação com o elemento procurado x.
- As ramificações correspondem ao resultado da comparação.
- As folhas correspondem às possíveis respostas do algoritmo. Então tal árvore deve ter pelo menos $n + 1$ folhas.
- Logo, a altura da árvore é pelo menos $\Omega(\lg n)$.