Algoritmos de Ordenação: Cota Inferior

ACH2002 - Introdução à Ciência da Computação II

Delano M. Beder

Escola de Artes, Ciências e Humanidades (EACH)
Universidade de São Paulo
dbeder@usp.br

10/2008

Material baseado em slides dos professores Cid de Souza e Cândida da Silva

O problema da Ordenação – Cota Inferior

- Estudamos diversos algoritmos para o problema da ordenação.
- Todos eles têm algo em comum: usam **somente comparações** entre dois elementos do conjunto a ser ordenado para definir a posição relativa desses elementos.
- Isto é o resultado da comparação de x_i com x_j , $i \neq j$, define se x_i será posicionado antes ou depois de x_j no conjunto ordenado.
- Todos os algoritmos dão uma **cota superior** para o número de comparações efetuadas por um algoritmo que resolva o problema da ordenação.
- A **menor** cota superior é dada pelos algoritmos *Mergesort* e o *Heapsort*, que efetuam $\Theta(n \log n)$ 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 efetua no mínimo $\Omega(n \log n)$ comparações no **pior caso**.
- Para demonstrar esse fato, 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 – Problema da Ordenação

- Considere a seguinte definição alternativa do problema da ordenação:

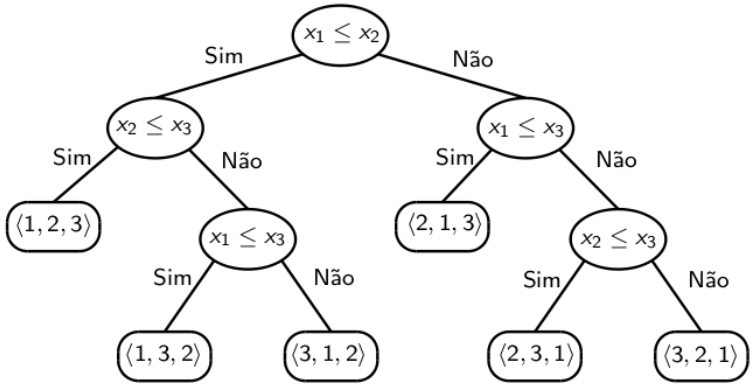
Problema da Ordenação:

Dado um conjunto de n inteiros $x_1, x_2, \ldots, x_n$ encontre uma permutação p dos índices $1 \leq i \leq n$ tal que $x_{p(1)} \leq x_{p(2)} \leq \cdots \leq x_{p(n)}$.

- É possível representar um algoritmo para o problema da ordenação através de uma árvore de decisão da seguinte forma:
  - Os nós internos representam comparações entre dois elementos do conjunto, 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.

Árvores de Decisão – Problema da Ordenação

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



Árvores de Decisão – 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 pelo menos $n!$ **folhas**, podendo ter mais (nada impede que duas sequências 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 dá o número mínimo de comparações que o melhor algoritmo de ordenação baseado em comparações deve efetuar.

A Cota Inferior

- Uma árvore binária de decisão T com altura h tem, no máximo, 2^h folhas.
- Portanto, se T tem pelo menos $n!$ folhas, $n! \leq 2^h$, ou seja $h \geq \log_2 n!$.

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

- Então, $h \in \Omega(n \log n)$.

Observações finais

- Provamos que $\Omega(n \log n)$ é uma **cota inferior** para o problema de ordenação.
- Portanto, os algoritmos *MergeSort* e *HeapSort* são algoritmos *ótimos*.

Referências utilizadas:

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest & Clifford Stein. *Algoritmos - Tradução da 2a. Edição Americana*. Editora Campus, 2002.