Algoritmos de Ordenação

Cota inferior

Professora:

Fátima L. S. Nunes

SISTEMAS DE INFORMAÇÃO

EACH

Algoritmos de Ordenação

- Algoritmos de ordenação que já conhecemos:

Algoritmos de Ordenação

- Algoritmos de ordenação que já conhecemos:
  - *Insertion Sort* (Ordenação por Inserção)
  - *Selection Sort* (Ordenação por Seleção)
  - *Bubble Sort* (Ordenação pelo método da Bolha)
  - *MergeSort* (Ordenação por intercalação)
  - *QuickSort* (Ordenação rápida)
  - *HeapSort* (Ordenação por monte)

Algoritmos de Ordenação

- Algoritmos de ordenação que já conhecemos:
  - *Insertion Sort* (Ordenação por Inserção)
  - *Selection Sort* (Ordenação por Seleção)
  - *Bubble Sort* (Ordenação pelo método da Bolha)
  - *MergeSort* (Ordenação por intercalação)
  - *QuickSort* (Ordenação rápida)
  - *HeapSort* (Ordenação por monte)

Quais suas complexidades?

Algoritmos de Ordenação

- Algoritmos de ordenação que já conhecemos:
  - *Insertion Sort* (Ordenação por Inserção) $O(n^2)$
  - *Selection Sort* (Ordenação por Seleção) $O(n^2)$
  - *Bubble Sort* (Ordenação pelo método da Bolha) $O(n^2)$
  - *MergeSort* (Ordenação por intercalação) $O(n \lg n)$
  - *QuickSort* (Ordenação rápida) $O(n \lg n)$ ***no caso médio***
  - *HeapSort* (Ordenação por monte) $O(n \lg n)$

Quais suas complexidades?

Algoritmos de Ordenação

- Algoritmos compartilham uma propriedade interessante:
  - sequência ordenada que determinam se baseia somente em comparações entre os elementos de entrada.
  - por isso, são chamados e ordenação por comparação.
  - veremos que qualquer ordenação por comparação deve efetuar **$\Omega(n \lg n)$** comparações no pior caso para ordenar ***n*** elementos.
  - O que significa isso?

Algoritmos de Ordenação

- Veremos que ualquer ordenação por comparação deve efetuar $\Omega(n \lg n)$ comparações no pior caso para ordenar ***n*** elementos.
  - O que significa isso?
  - o *MergeSort* e o *HeapSort* são algoritmos assintoticamente ótimos: não existe nenhuma ordenação por comparação que seja mais rápida por mais de um fator constante.

Limites inferiores para ordenação

- Ordenação por comparação:
  - usamos apenas comparações entre elementos para obter informações de ordem sobre uma sequência de entrada $<a_1, a_2, \ldots, a_n>$
  - dados dois elementos de entrada a_i e a_j, usamos um testes para determinar sua ordem relativa no conjunto de dados:

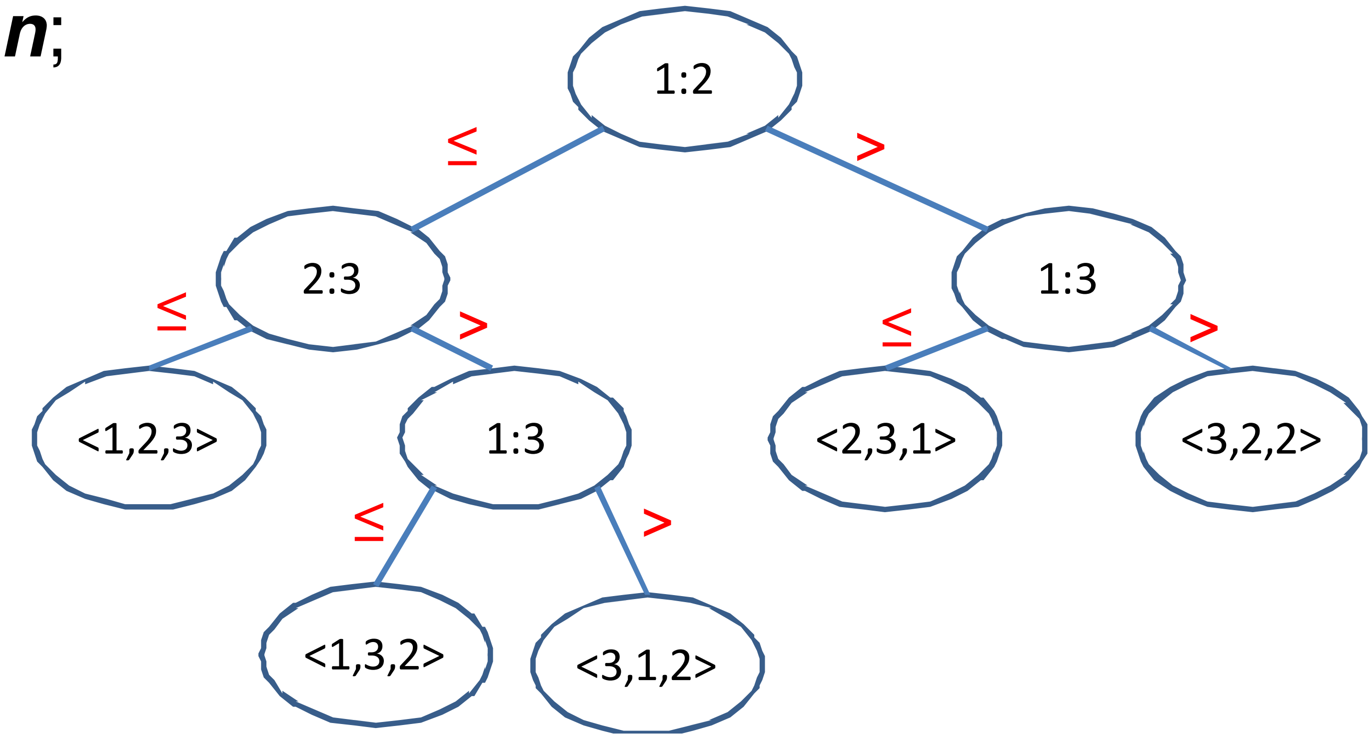
    $a_i < a_j$, $a_i \leq a_j$, $a_i = a_j$, $a_i \geq a_j$, $a_i > a_j$.

Limites inferiores para ordenação

- Ordenação por comparação:
  - vamos supor que todos os elementos são distintos $\Rightarrow$ eliminam-se comparações $a_i = a_j$
  - as demais comparações são equivalentes porque produzem a mesma informação: ordem relativa de a_i e a_j
  - Então, supomos que todas as comparações são do tipo $a_i \leq a_j$

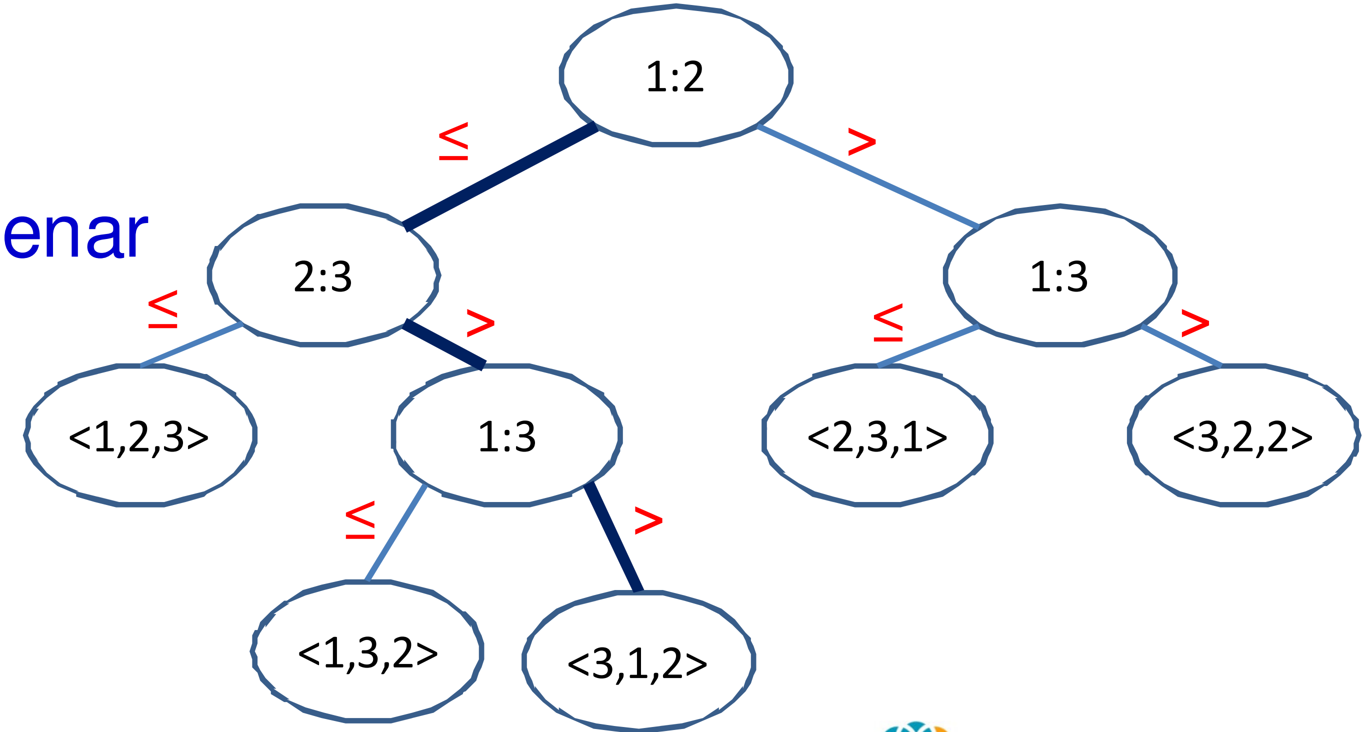
Modelo de árvore de decisão

- Ordenações por comparação podem ser vistas como árvores de decisão:
  - árvore binária cheia que representa as comparações executadas pelo algoritmo sobre uma entrada de dados de tamanho ***n***;
  - outros aspectos do algoritmo são ignorados.
  - cada nó da árvore é anotado por *i:j* para *i,j* no intervalo $\boldsymbol{1 \leq i, j \leq n}$;



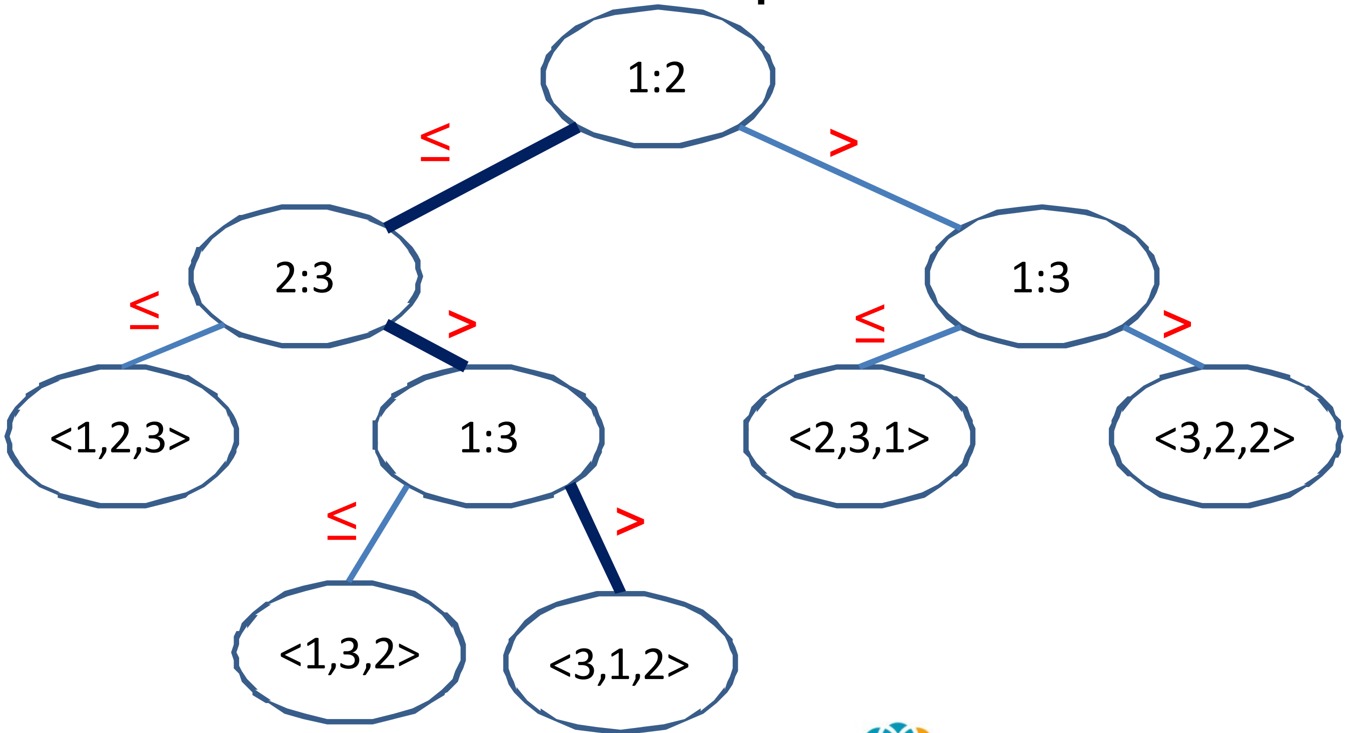
Modelo de árvore de decisão

- Ordenações por comparação podem ser vistas como árvores de decisão:
  - cada folha é anotada por uma permutação < π(1), π(2), ..., π(3)>: decisões após as comparações executadas pelo algoritmo.
  - execução do algoritmo: traçar um caminho deste a raiz até um nó folha
  - Exemplo:
    caminho para ordenar
    $<a_1=6,a_2=8,a_3=5>$



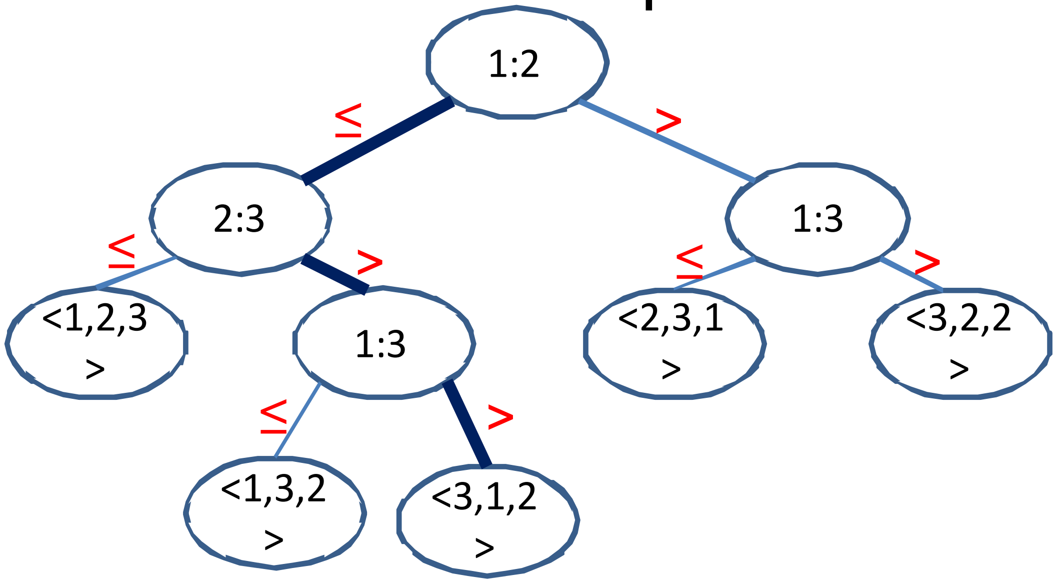
Modelo de árvore de decisão

- Ao final do algoritmo por comparação, sempre se atingirá uma folha:
- Condições necessárias para algoritmo estar correto:
  - cada uma das ***n!*** permutações sobre ***n*** elementos deve aparecer como uma das folhas da árvore de decisão;
  - cada uma das folhas deve ser acessível por um caminho a partir da raiz.



Limite inferior para o pior caso

- Dada uma árvore de decisão, qual seria o tamanho do pior caso para um algoritmo de ordenação por comparação?
  - tamanho do caminho mais longo deste a raiz até qualquer uma de suas folhas.
  - Então: número de comparações do pior caso = altura da árvore.
  - Limite inferior sobre a altura de todas as árvores de decisão em que cada permutação aparece como uma folha acessível é um limite inferior sobre o tempo de execução do algoritmo



Limite inferior para o pior caso

- Teorema:

  Qualquer algoritmo de ordenação por comparação exige **Ω(n lg n)** comparações no pior caso.

- Prova:
  - a partir do exposto anteriormente, basta determinar a altura de uma árvore de decisão em que cada permutação aparece como uma folha acessível.
  - considerando uma árvore de altura ***b*** com ***l*** folhas acessíveis:
    - cada uma das ***n!*** permutações da entrada aparece como alguma folha $\Rightarrow$ temos $n! \leq l$;
    - árvore binária de altura ***h*** tem no máximo 2^h folhas;
    - então: $n! \leq l \leq 2^h$
    - $h \geq lg(n!) = \Omega(n \lg n)$

Limite inferior para o pior caso

- Corolário:

  O HeapSort e o MergeSort são ordenações por comparação assintoticamente ótimas

- Prova:
  - Os tempos $O(n \lg n)$ limites superiores para o HeapSort e o MergeSort correspondem ao limite inferior $\Omega(n \lg n)$ do pior caso do teorema anterior.

Referências

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

Algoritmos de Ordenação

Cota inferior

Professora:

Fátima L. S. Nunes