Aula 13 – Algoritmos de Ordenação: Modelo Incremental e Cota Inferior

Norton Trevisan Roman
norton@usp.br

27 de setembro de 2018

Projeto por Indução Simples

Terceira Alternativa

Projeto por Indução Simples

Terceira Alternativa

- **Base**: $n = 1$. Um único elemento está ordenado

Projeto por Indução Simples

Terceira Alternativa

- **Base**: $n = 1$. Um único elemento está ordenado
- **H.I.**: Sei ordenar um conjunto de $n - 1 \geq 1$ valores

Projeto por Indução Simples

Terceira Alternativa

- **Base**: $n = 1$. Um único elemento está ordenado
- **H.I.**: Sei ordenar um conjunto de $n - 1 \geq 1$ valores
- **Passo**:

Projeto por Indução Simples

Terceira Alternativa

- **Base**: $n = 1$. Um único elemento está ordenado
- **H.I.**: Sei ordenar um conjunto de $n - 1 \geq 1$ valores
- **Passo**:
  - Seja S um conjunto de $n \geq 2$ valores, e x o maior elemento de S

Projeto por Indução Simples

Terceira Alternativa

- **Base**: $n = 1$. Um único elemento está ordenado
- **H.I.**: Sei ordenar um conjunto de $n - 1 \geq 1$ valores
- **Passo**:
  - Seja S um conjunto de $n \geq 2$ valores, e x o maior elemento de S
  - Então x certamente é o último elemento da sequência ordenada de S. Por hipótese de indução, sabemos ordenar os demais $S - x$ elementos, e assim obtemos S ordenado

Projeto por Indução Simples

Terceira Alternativa

- **Base**: $n = 1$. Um único elemento está ordenado
- **H.I.**: Sei ordenar um conjunto de $n - 1 \geq 1$ valores
- **Passo**:
  - Seja S um conjunto de $n \geq 2$ valores, e x o maior elemento de S
  - Então x certamente é o último elemento da sequência ordenada de S. Por hipótese de indução, sabemos ordenar os demais $S - x$ elementos, e assim obtemos S ordenado
- Variação do Método da Seleção

Projeto por Indução Simples

- No entanto, se implementarmos de uma forma diferente a seleção e o posicionamento do maior elemento, obteremos o algoritmo da Bolha (Bubble Sort):

Projeto por Indução Simples

- No entanto, se implementarmos de uma forma diferente a seleção e o posicionamento do maior elemento, obteremos o algoritmo da Bolha (Bubble Sort):

```
Bolha(A, n):
  Entrada: Arranjo A de n valores
  Saída: Aranjo A ordenado.

para i = n - 1 até 1 faça:
  para j = 1 até i faça:
    se A[j-1] > A[j] então
      t = A[j-1]
      A[j-1] = A[j]
      A[j] = t
```

Projeto por Indução Simples

Método da Bolha (iterativo)

- Usando a notação O, quantas comparações no arranjo são feitas no pior caso?

```
Bolha(A, n):
  para i = n - 1 até 1 faça:
    para j = 1 até i faça:
      se A[j-1] > A[j] então
        t = A[j-1]
        A[j-1] = A[j]
        A[j] = t
```

Projeto por Indução Simples

Método da Bolha (iterativo)

- Usando a notação O, quantas comparações no arranjo são feitas no pior caso?
- $O(n)$

```
Bolha(A, n):
  para i = n - 1 até 1 faça:
    para j = 1 até i faça:
      se A[j-1] > A[j] então
        t = A[j-1]
        A[j-1] = A[j]
        A[j] = t
```

Projeto por Indução Simples

Método da Bolha (iterativo)

- Usando a notação O, quantas comparações no arranjo são feitas no pior caso?
- $O(n) \times O(n)$

```
Bolha(A, n):
  para i = n - 1 até 1 faça:
    para j = 1 até i faça:
      se A[j-1] > A[j] então
        t = A[j-1]
        A[j-1] = A[j]
        A[j] = t
```

Projeto por Indução Simples

Método da Bolha (iterativo)

- Usando a notação O, quantas comparações no arranjo são feitas no pior caso?

```
Bolha(A, n):
  para i = n - 1 até 1 faça:
    para j = 1 até i faça:
      se A[j-1] > A[j] então
        t = A[j-1]
        A[j-1] = A[j]
        A[j] = t
```

- $O(n) \times O(n) = O(n^2)$

Projeto por Indução Simples

Método da Bolha (iterativo)

- E quantas trocas de elementos são feitas no arranjo no pior caso?

```
Bolha(A, n):
  para i = n - 1 até 1 faça:
    para j = 1 até i faça:
      se A[j-1] > A[j] então
        t = A[j-1]
        A[j-1] = A[j]
        A[j] = t
```

Projeto por Indução Simples

Método da Bolha (iterativo)

- E quantas trocas de elementos são feitas no arranjo no pior caso?

```
Bolha(A, n):
  para i = n - 1 até 1 faça:
    para j = 1 até i faça:
      se A[j-1] > A[j] então
        t = A[j-1]
        A[j-1] = A[j]
        A[j] = t
```

- $O(n)$

Projeto por Indução Simples

Método da Bolha (iterativo)

- E quantas trocas de elementos são feitas no arranjo no pior caso?

```
Bolha(A, n):
  para i = n - 1 até 1 faça:
    para j = 1 até i faça:
      se A[j-1] > A[j] então
        t = A[j-1]
        A[j-1] = A[j]
        A[j] = t
```

- $O(n) \times O(n)$

Projeto por Indução Simples

Método da Bolha (iterativo)

- E quantas trocas de elementos são feitas no arranjo no pior caso?

```
Bolha(A, n):
  para i = n - 1 até 1 faça:
    para j = 1 até i faça:
      se A[j-1] > A[j] então
        t = A[j-1]
        A[j-1] = A[j]
        A[j] = t
```

- $O(n) \times O(n) = O(n^2)$

Cota Inferior

Ordenação: Cota Inferior

- Vimos diversos algoritmos para o problema da ordenação

Ordenação: Cota Inferior

- Vimos diversos algoritmos para o problema da ordenação
- Todos eles têm algo em comum:

Ordenação: Cota Inferior

- Vimos diversos algoritmos para o problema da ordenação
- Todos eles têm algo em comum:
  - Usam somente comparações entre elementos do conjunto a ser ordenado para definir a posição relativa desses elementos

Ordenação: Cota Inferior

- Vimos diversos algoritmos para o problema da ordenação
- Todos eles têm algo em comum:
  - Usam somente comparações entre elementos do conjunto a ser ordenado para definir a posição relativa desses elementos
  - Ou seja, 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

Ordenação: Cota Inferior

- Vimos diversos algoritmos para o problema da ordenação
- Todos eles têm algo em comum:
  - Usam somente comparações entre elementos do conjunto a ser ordenado para definir a posição relativa desses elementos
  - Ou seja, 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 ao resolver o problema da ordenação

Ordenação: Cota Inferior

- A menor cota superior dada pelos algoritmos até então é $O(n^2)$, relativa ao número de comparações no pior caso

Ordenação: Cota Inferior

- A menor cota superior dada pelos algoritmos até então é $O(n^2)$, relativa ao número de comparações no pior caso
- Será que é possível projetar um algoritmo de ordenação baseado em comparações assintoticamente mais eficiente que isso?

Ordenação: Cota Inferior

- A menor cota superior dada pelos algoritmos até então é $O(n^2)$, relativa ao número de comparações no pior caso
- Será que é possível projetar um algoritmo de ordenação baseado em comparações assintoticamente mais eficiente que isso?
- Sim

Ordenação: Cota Inferior

- A menor cota superior dada pelos algoritmos até então é $O(n^2)$, relativa ao número de comparações no pior caso
- Será que é possível projetar um algoritmo de ordenação baseado em comparações assintoticamente mais eficiente que isso?
- Sim
  - Veremos que **qualquer** algoritmo que ordena n elementos, baseado apenas em comparações, efetua no mínimo $\Omega(nlog(n))$ comparações no pior caso

Ordenação: Cota Inferior

- Para isso, vamos representar os algoritmos de ordenação em um modelo computacional abstrato: a **árvore (binária) de decisão**

Ordenação: Cota Inferior

- Para isso, vamos representar os algoritmos de ordenação em um modelo computacional abstrato: a **árvore (binária) de decisão**

Árvores de Decisão

- Uma árvore de decisão é uma representação gráfica formada por um conjunto de nós, unidos entre si por arestas

Ordenação: Cota Inferior

- Para isso, vamos representar os algoritmos de ordenação em um modelo computacional abstrato: a **árvore (binária) de decisão**

Árvores de Decisão

- Uma árvore de decisão é uma representação gráfica formada por um conjunto de nós, unidos entre si por arestas
- Cada nó representa um ponto de decisão

Ordenação: Cota Inferior

Árvores de Decisão

- Cada aresta saindo do nó representa a decisão tomada, levando a outro nó, ou seja, a outro ponto de decisão

Ordenação: Cota Inferior

Árvores de Decisão

- Cada aresta saindo do nó representa a decisão tomada, levando a outro nó, ou seja, a outro ponto de decisão
- Toda a representação começa em um nó inicial, a **raiz** da árvore, que representa o primeiro ponto de decisão

Ordenação: Cota Inferior

Árvores de Decisão

- Cada aresta saindo do nó representa a decisão tomada, levando a outro nó, ou seja, a outro ponto de decisão
- Toda a representação começa em um nó inicial, a **raiz** da árvore, que representa o primeiro ponto de decisão
- Terminando então nos resultados finais das decisões, as **folhas**, a partir das quais mais nenhuma decisão é tomada

Ordenação: Cota Inferior

Árvores de Decisão

- No caso da ordenação, os nós internos representam comparações feitas pelo algoritmo

Ordenação: Cota Inferior

Árvores de Decisão

- No caso da ordenação, os nós internos representam comparações feitas pelo algoritmo
- As arestas representam os possíveis resultados dessas comparações

Ordenação: Cota Inferior

Árvores de Decisão

- No caso da ordenação, os nós internos representam comparações feitas pelo algoritmo
- As arestas representam os possíveis resultados dessas comparações
- E as subárvores de cada nó interno representam possibilidades de continuidade das ações do algoritmo após a comparação

Ordenação: Cota Inferior

Árvores de Decisão

- No caso das árvores binárias de decisão, cada nó possui apenas duas subárvores

Ordenação: Cota Inferior

Árvores de Decisão

- 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

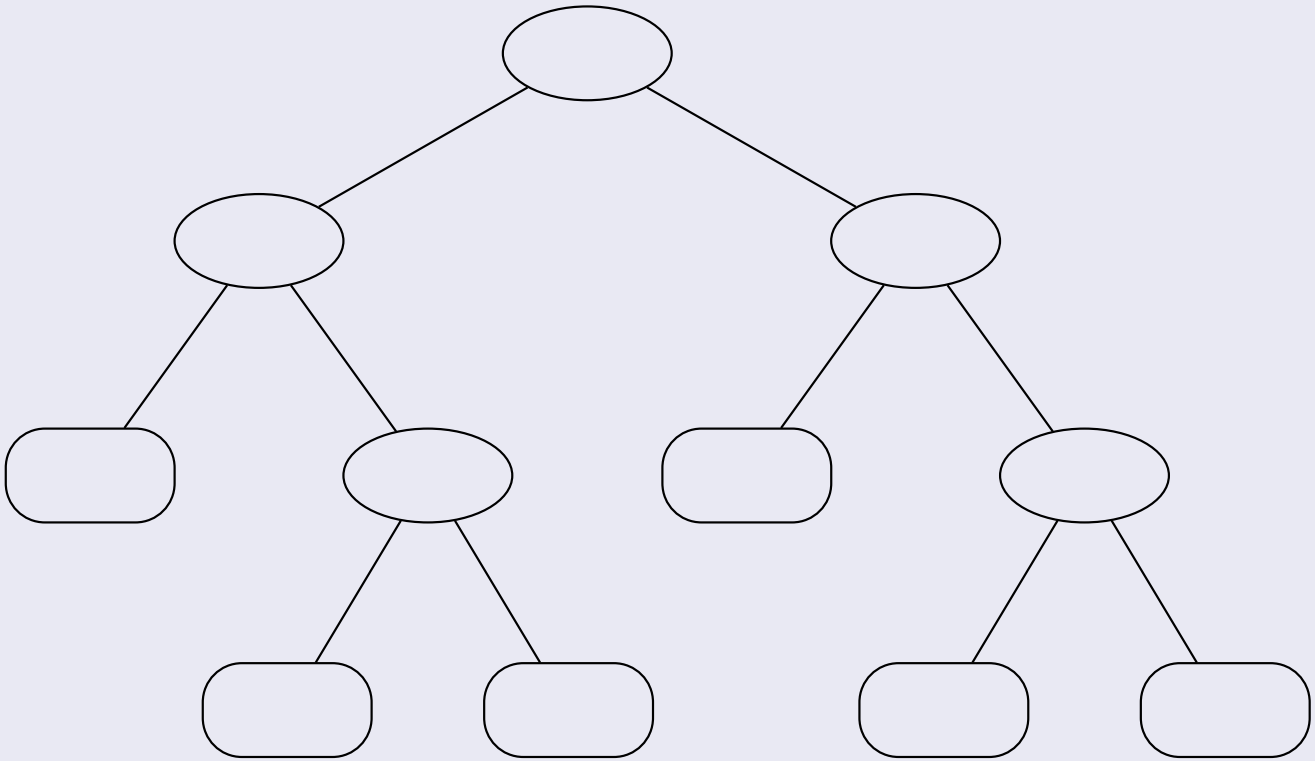
Ordenação: Cota Inferior

Árvores de Decisão

- 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

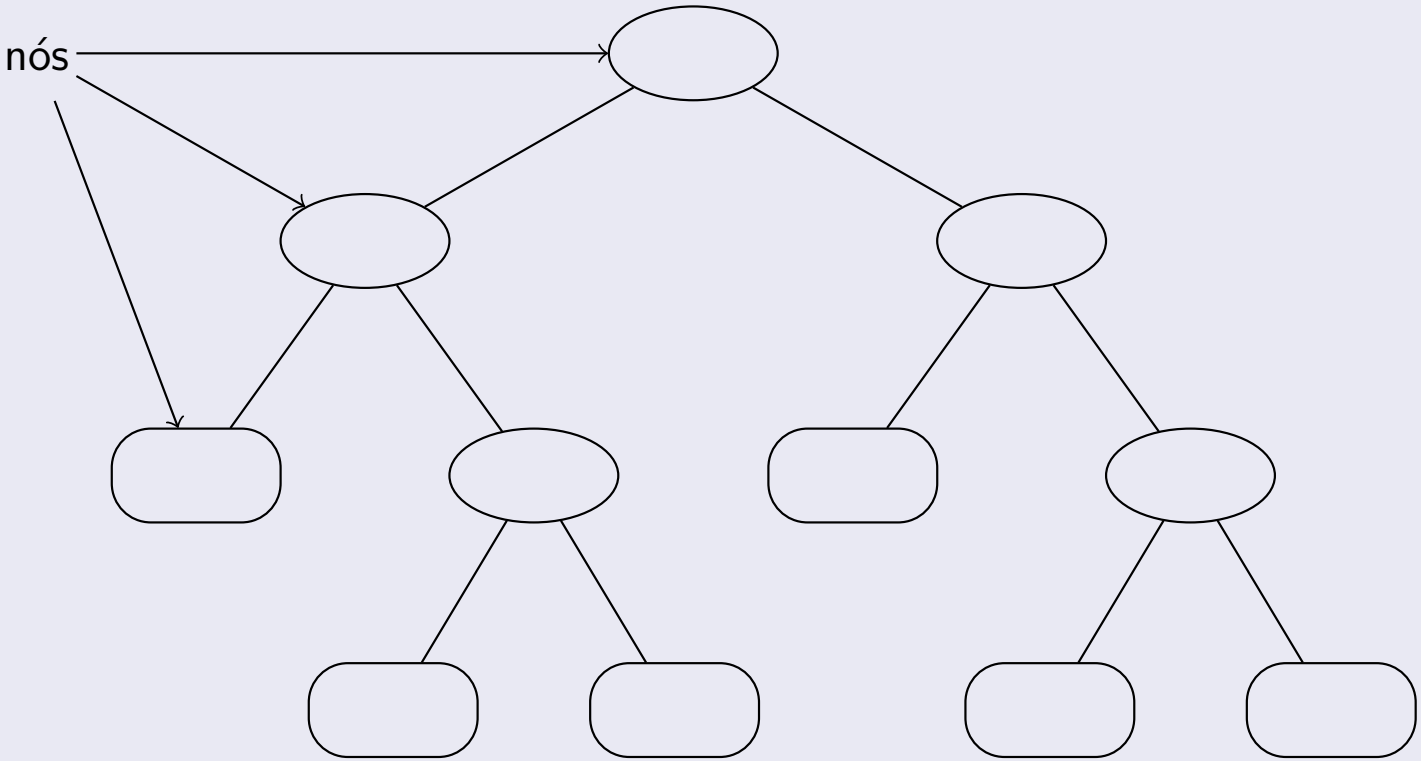
Ordenação: Cota Inferior

Árvores de Decisão



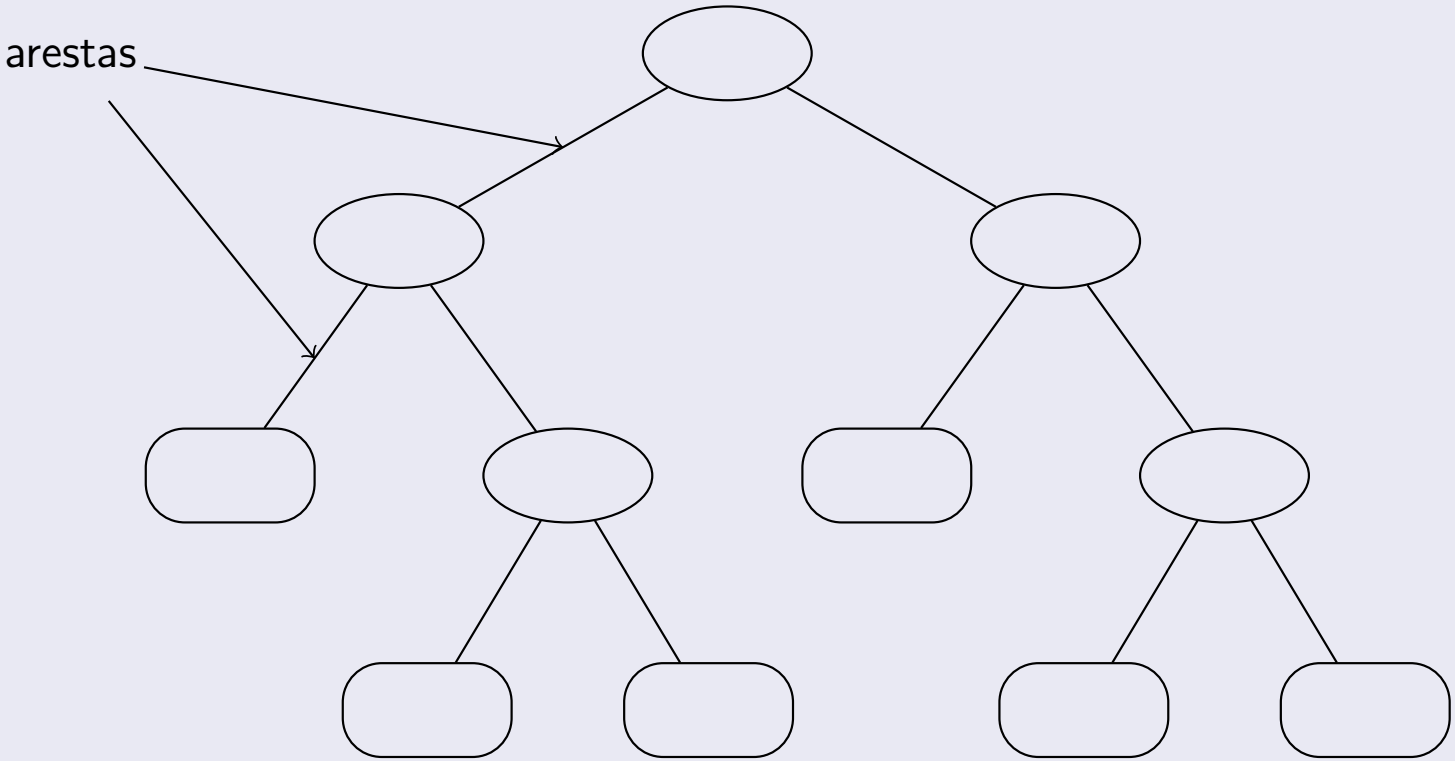
Ordenação: Cota Inferior

Árvores de Decisão



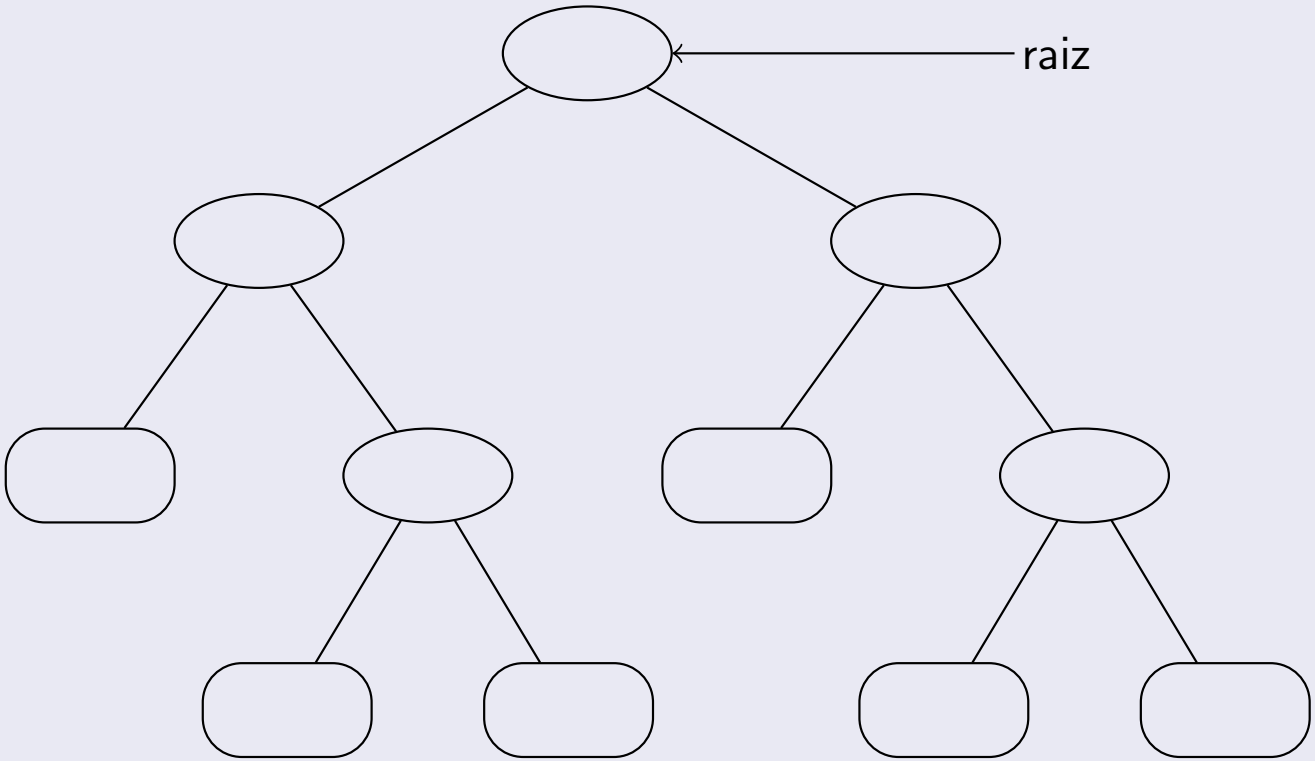
Ordenação: Cota Inferior

Árvores de Decisão



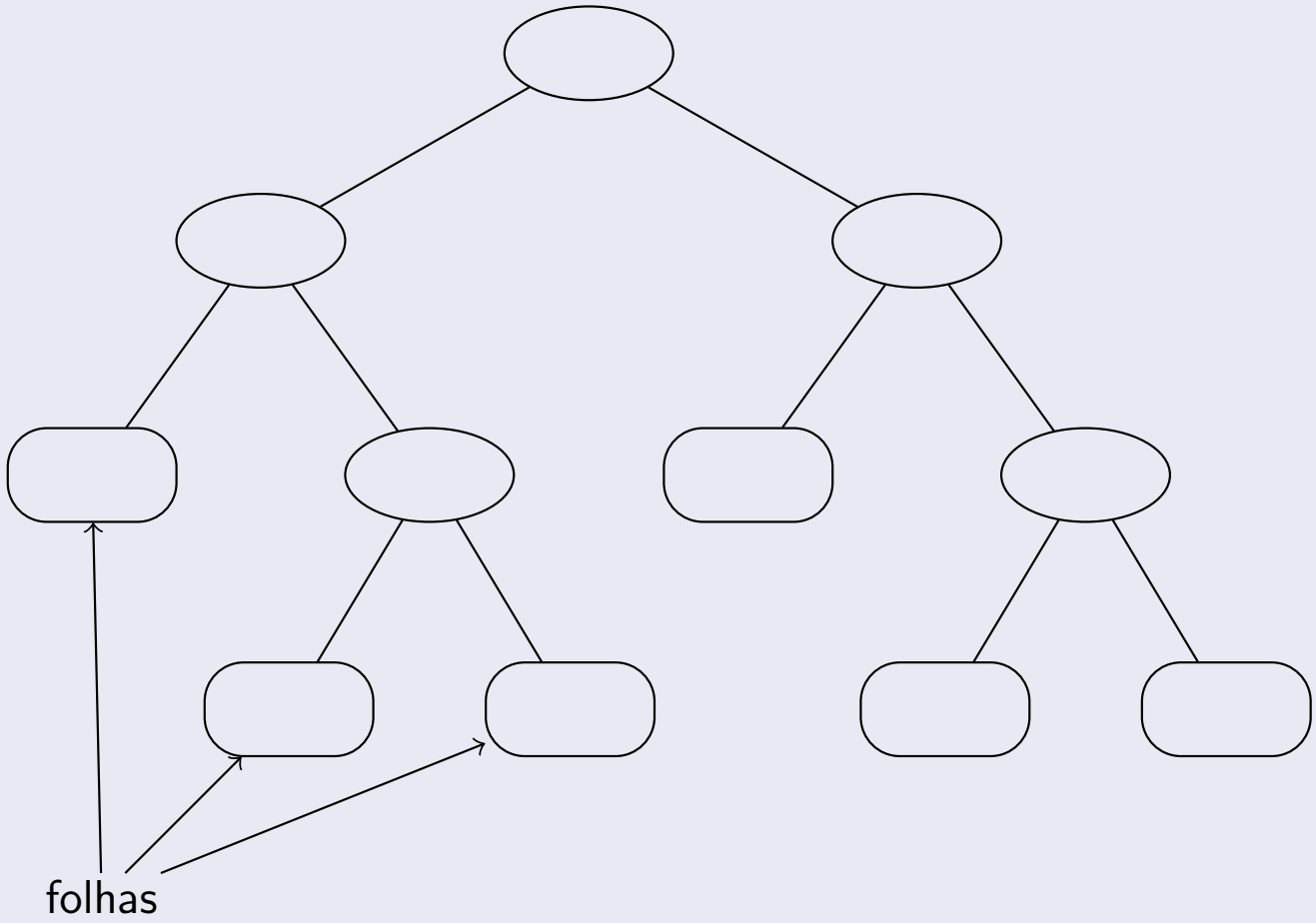
Ordenação: Cota Inferior

Árvores de Decisão



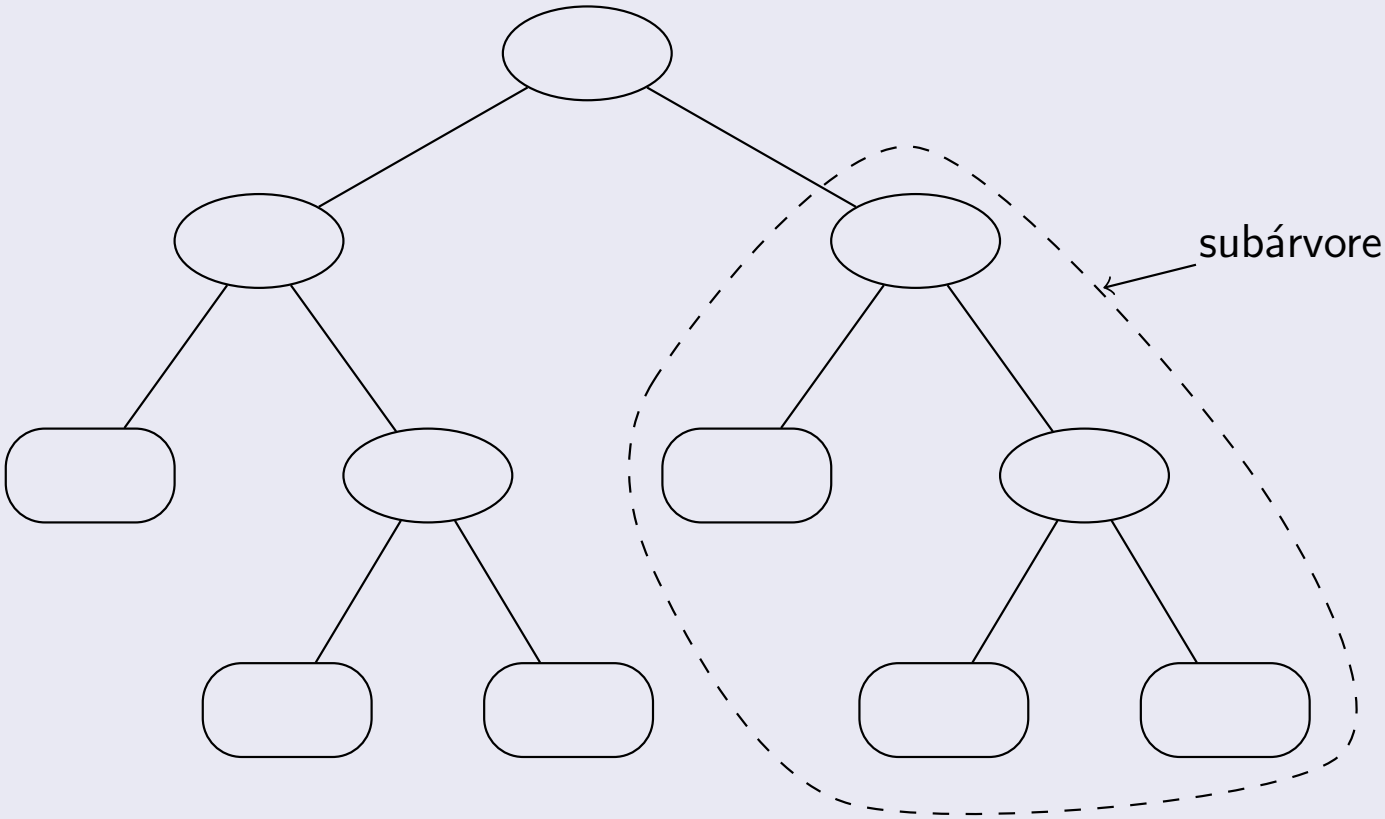
Ordenação: Cota Inferior

Árvores de Decisão



Ordenação: Cota Inferior

Árvores de Decisão



Ordenação: Cota Inferior

Árvores de Decisão – Ordenação

- Considere a seguinte definição alternativa do problema da ordenação:

Ordenação: Cota Inferior

Árvores de Decisão – Ordenação

- Considere a seguinte definição alternativa do problema da ordenação:
- Dado um conjunto de n valores $x_1, x_2, \ldots, x_n$, encontre uma permutação p dos índices $1 \leq i \leq n$ tal que $x_{p(1)} \leq x_{p(2)} \leq \ldots \leq x_{p(n)}$

Ordenação: Cota Inferior

Árvores de Decisão – Ordenação

- Considere a seguinte definição alternativa do problema da ordenação:
- Dado um conjunto de n valores $x_1, x_2, \ldots, x_n$, encontre uma permutação p dos índices $1 \leq i \leq n$ tal que $x_{p(1)} \leq x_{p(2)} \leq \ldots \leq x_{p(n)}$
  - Representaremos uma permutação como $\langle p(1), p(2), \ldots, p(n) \rangle$, significando uma ordem específica dos valores $x_1, x_2, \ldots, x_n$

Ordenação: Cota Inferior

Árvores de Decisão – Ordenação

- Considere a seguinte definição alternativa do problema da ordenação:
- Dado um conjunto de n valores $x_1, x_2, \ldots, x_n$, encontre uma permutação p dos índices $1 \leq i \leq n$ tal que $x_{p(1)} \leq x_{p(2)} \leq \ldots \leq x_{p(n)}$
  - Representaremos uma permutação como $\langle p(1), p(2), \ldots, p(n) \rangle$, significando uma ordem específica dos valores $x_1, x_2, \ldots, x_n$
  - Ex: $\langle 3, 1, 2 \rangle$ significa a permutação $\langle x_3, x_1, x_2 \rangle$ da entrada x_1, x_2, x_3

Ordenação: Cota Inferior

Árvores de Decisão – Ordenação

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

Ordenação: Cota Inferior

Árvores de Decisão – Ordenação

- 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, ex: $x_i \leq x_j$

Ordenação: Cota Inferior

Árvores de Decisão – Ordenação

- 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, ex: $x_i \leq x_j$
  - As ramificações representam os possíveis resultados da comparação: verdadeiro se $x_i \leq x_j$, ou falso se $x_i > x_j$

Ordenação: Cota Inferior

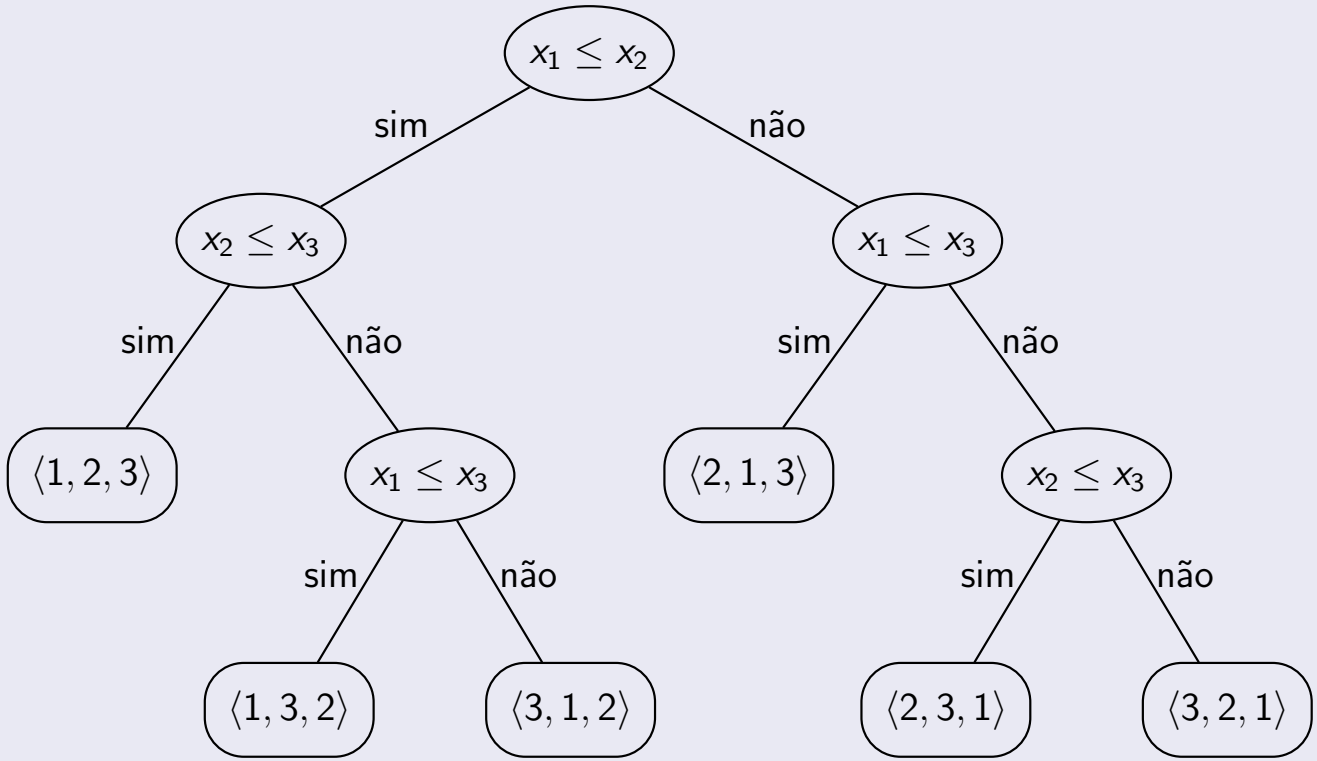
Árvores de Decisão – Ordenação

- 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, ex: $x_i \leq x_j$
  - 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

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

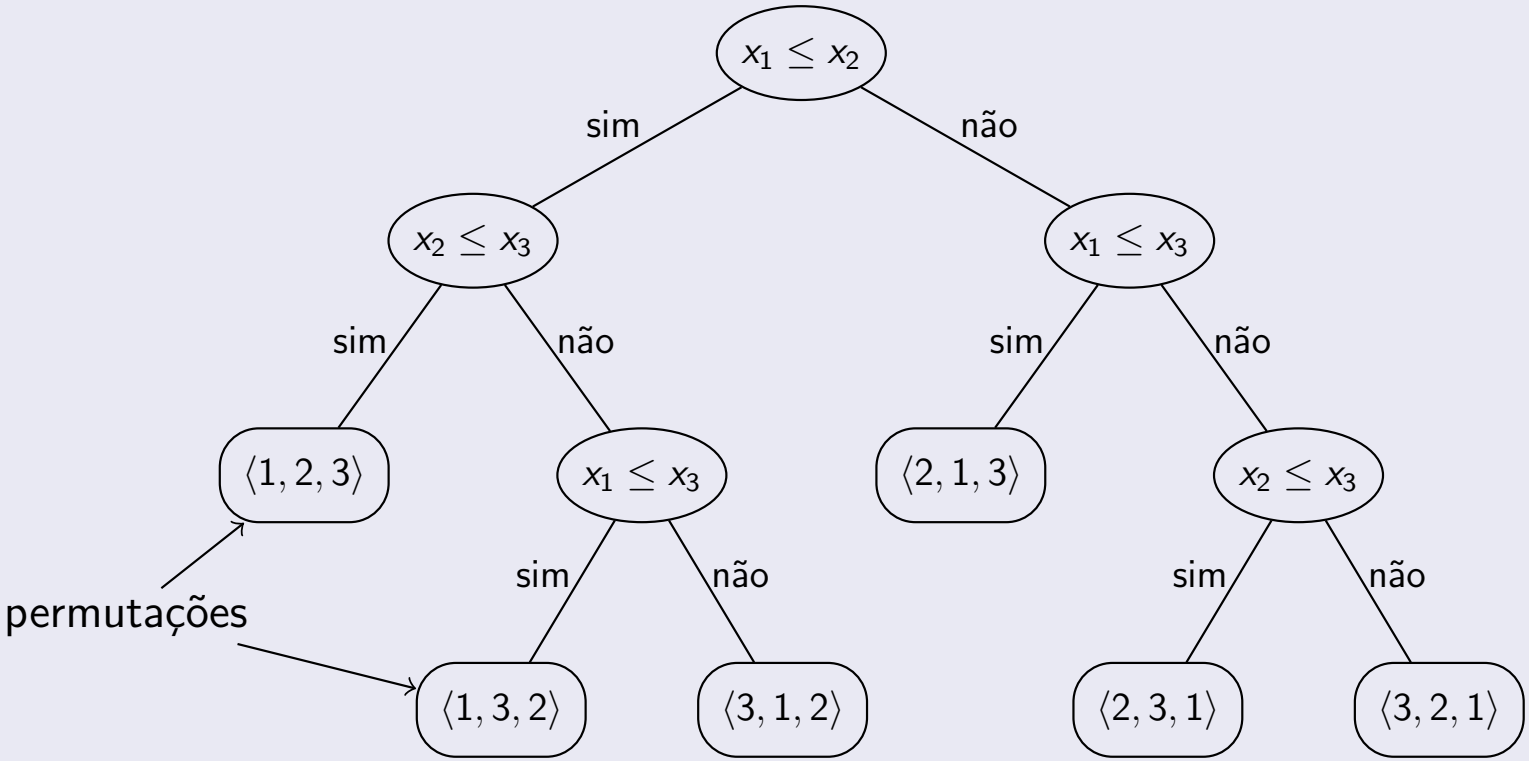
- Ex: Árvore de decisão para o método da Inserção, com arranjo de 3 elementos:



Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

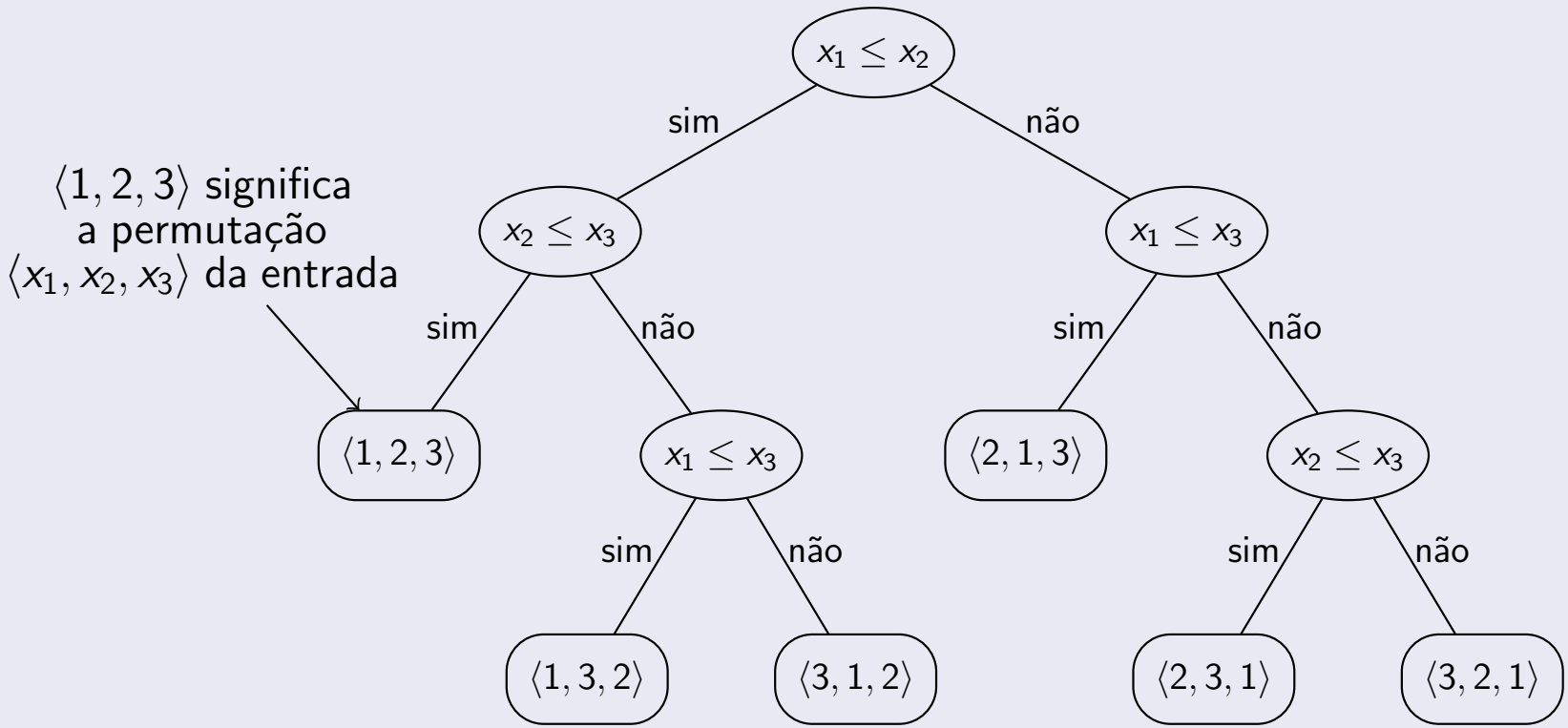
- Ex: Árvore de decisão para o método da Inserção, com arranjo de 3 elementos:



Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

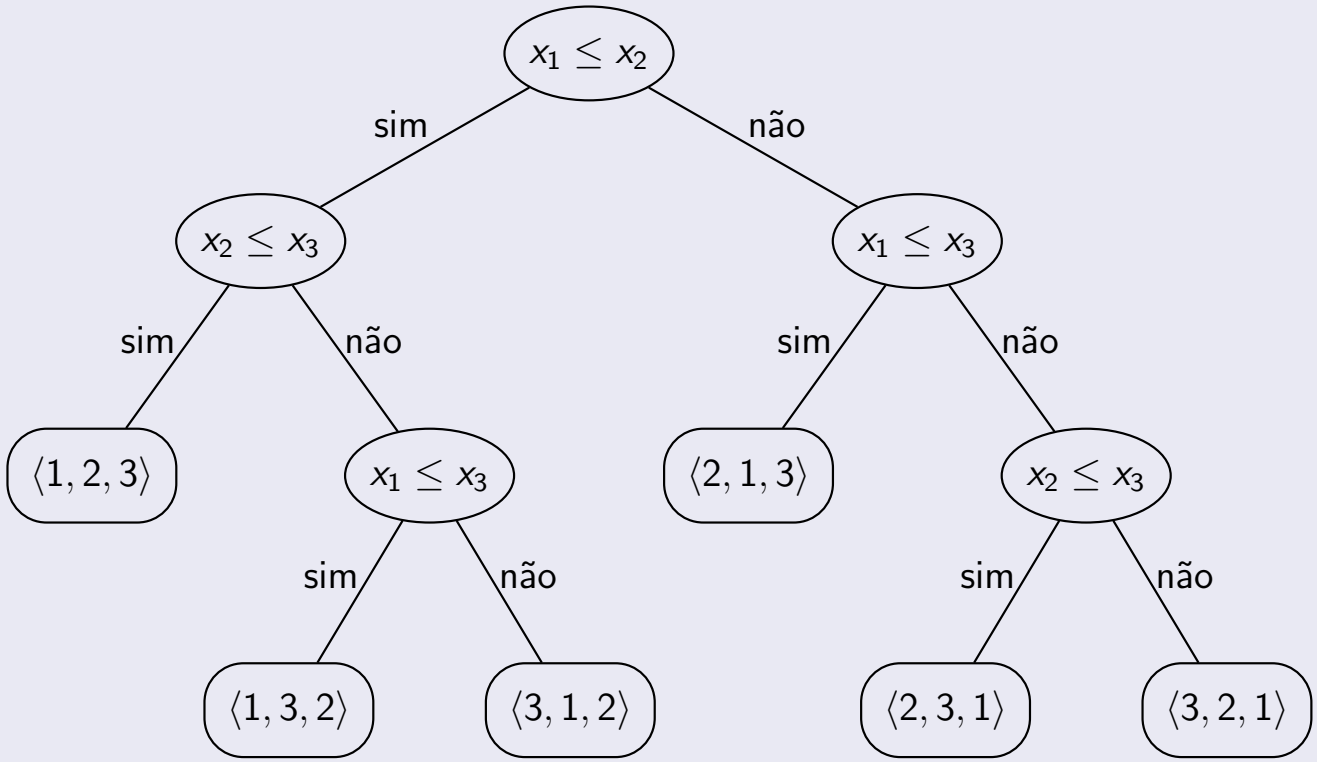
- Ex: Árvore de decisão para o método da Inserção, com arranjo de 3 elementos:



Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

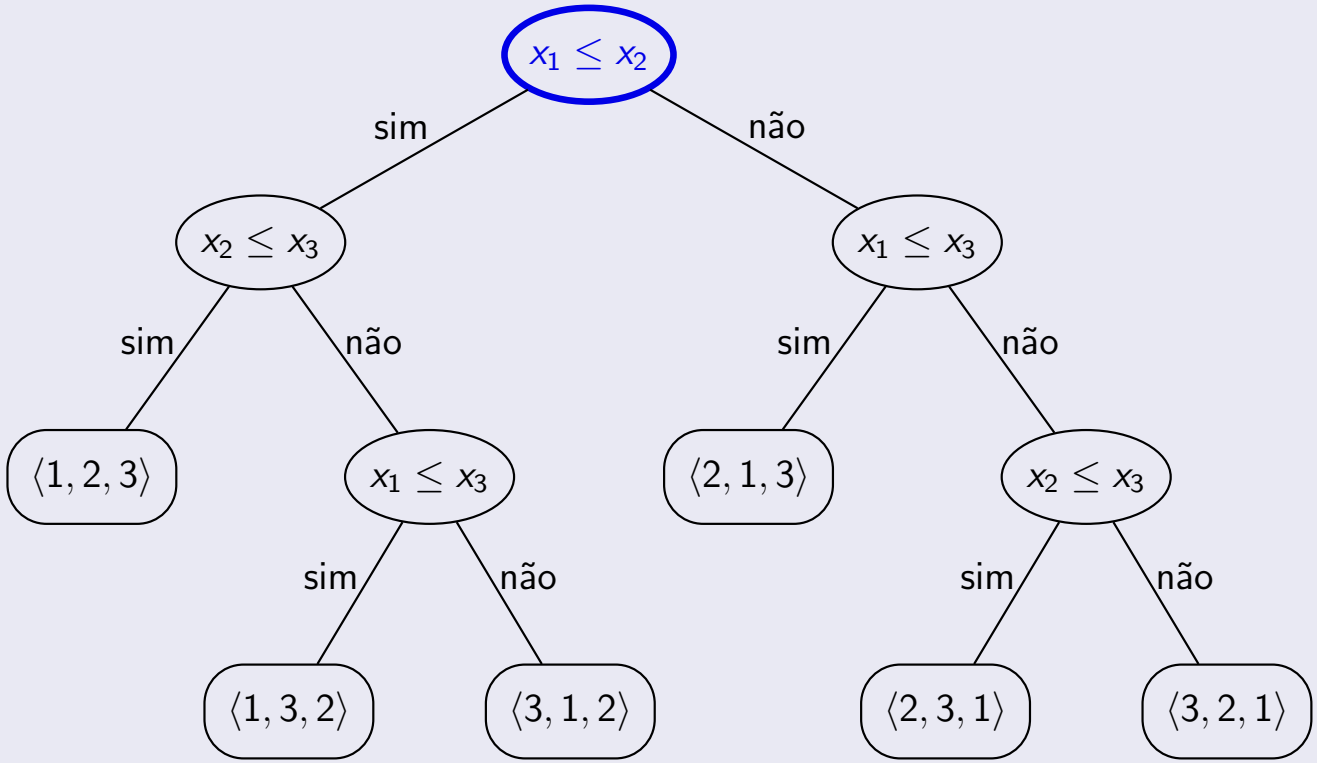
- Ex: Entrada $\langle x_1 = 6, x_2 = 8, x_3 = 5 \rangle$



Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

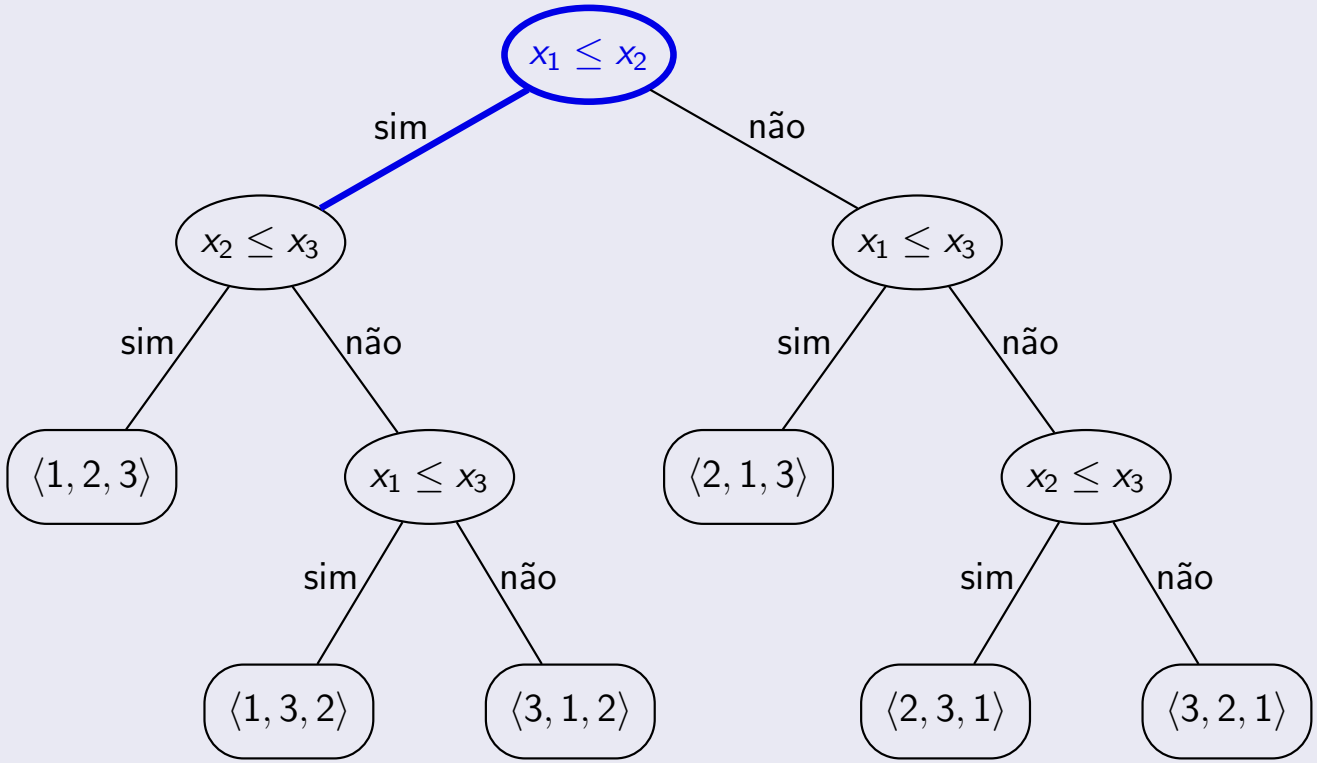
- Ex: Entrada $\langle x_1 = 6, x_2 = 8, x_3 = 5 \rangle$



Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

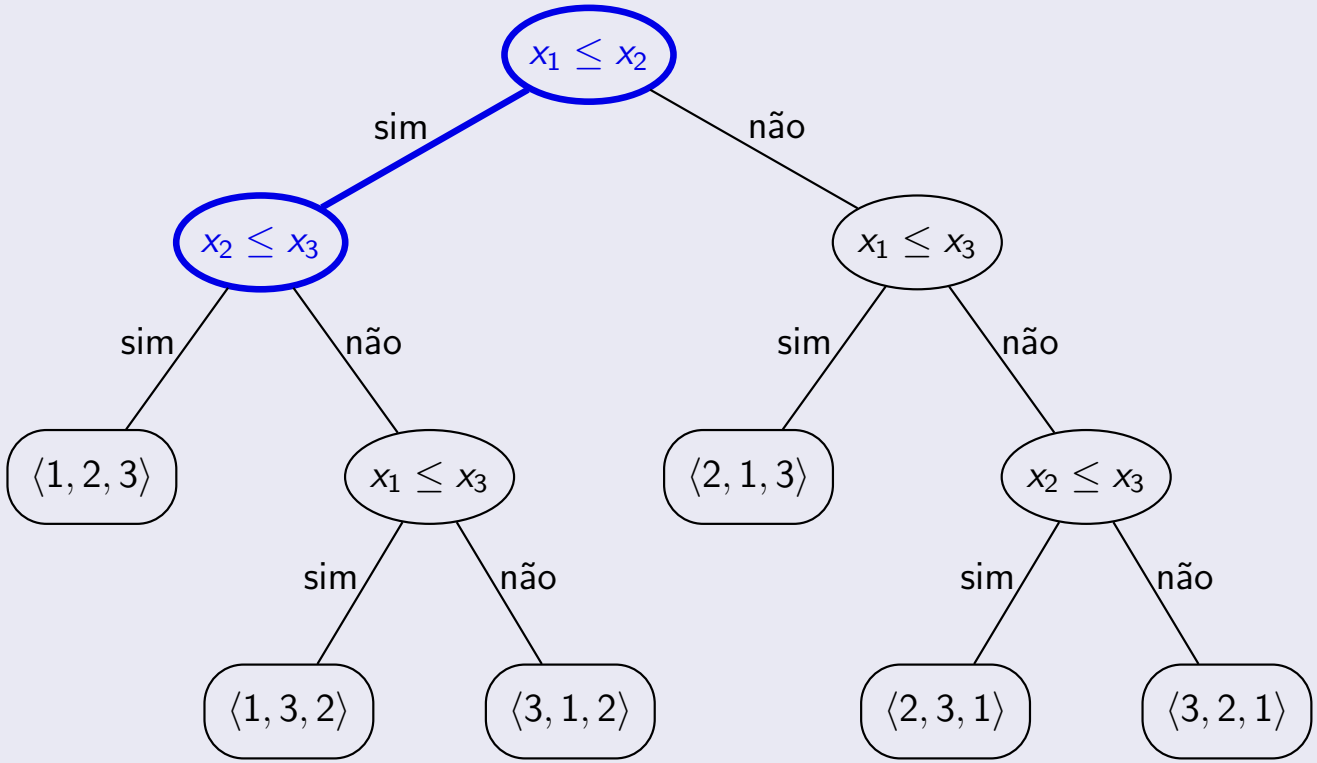
- Ex: Entrada $\langle x_1 = 6, x_2 = 8, x_3 = 5 \rangle$



Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

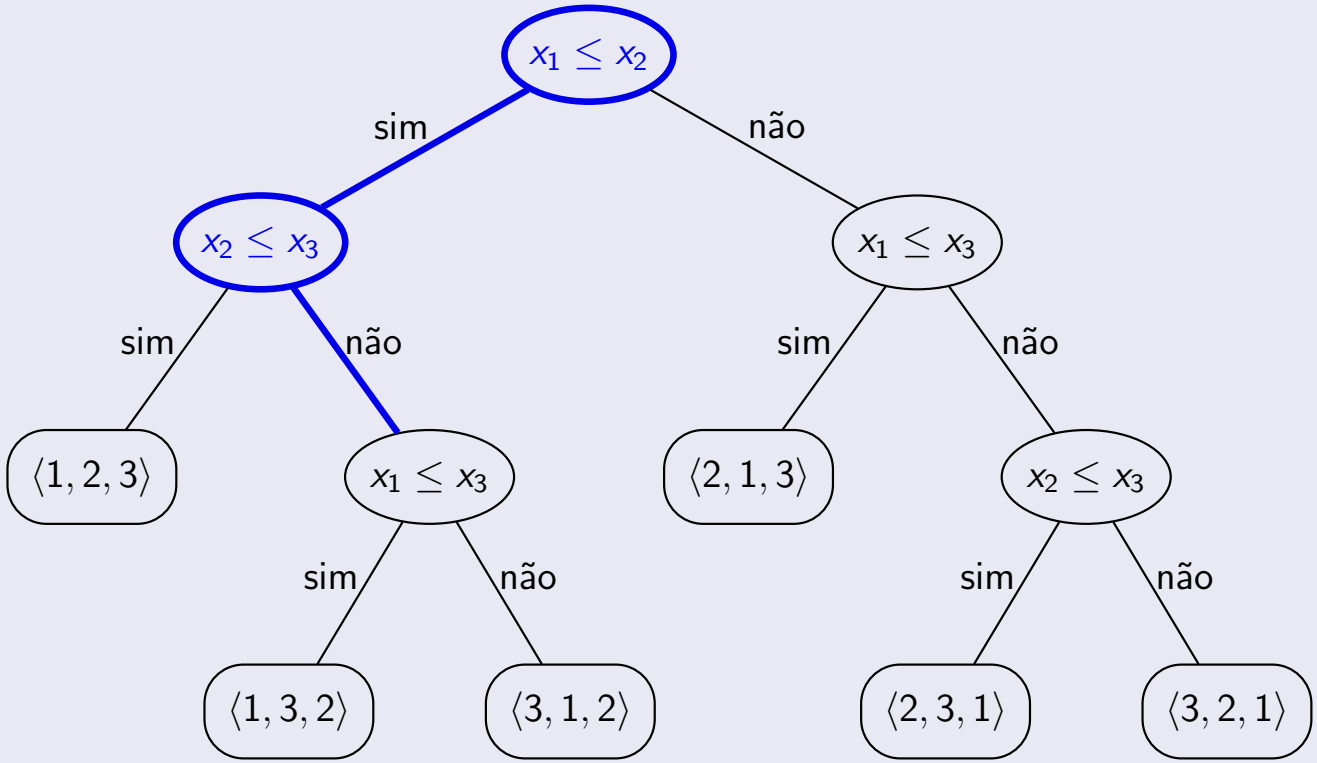
- Ex: Entrada $\langle x_1 = 6, x_2 = 8, x_3 = 5 \rangle$



Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

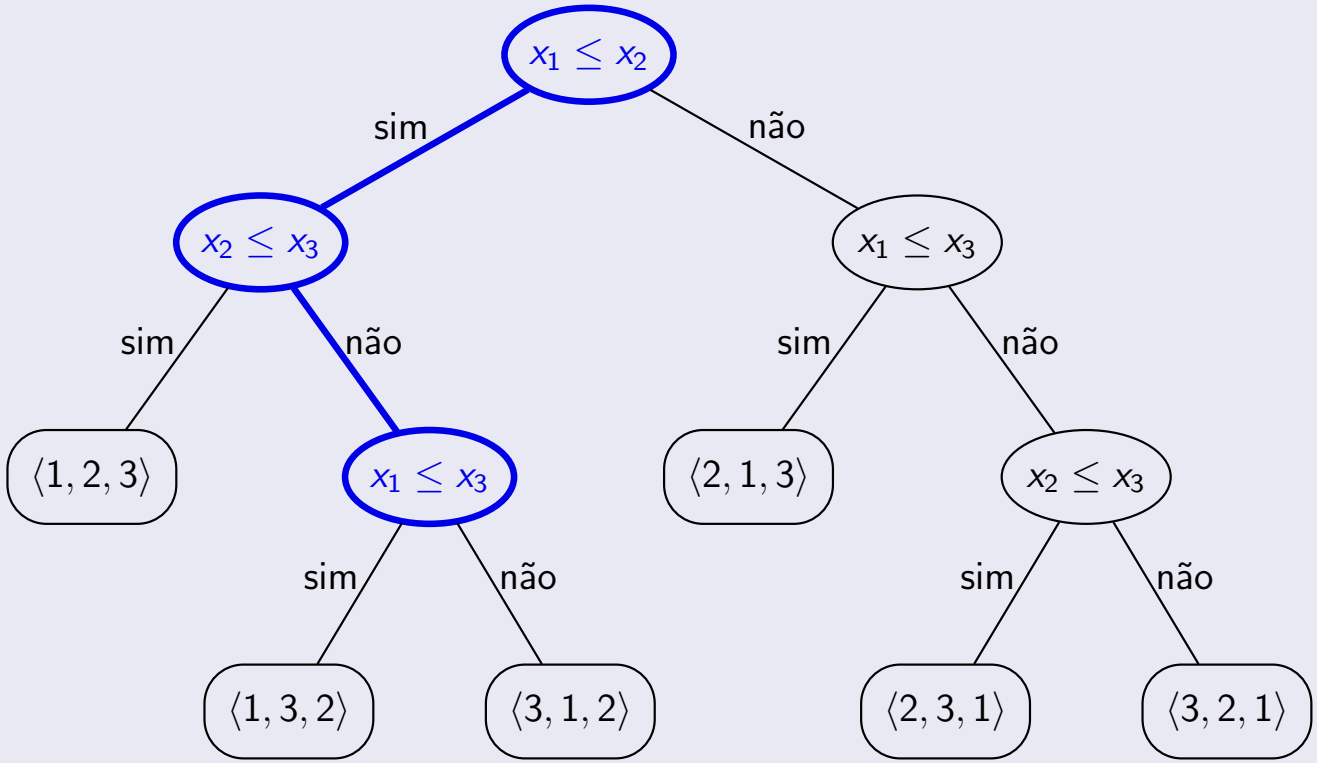
- Ex: Entrada $\langle x_1 = 6, x_2 = 8, x_3 = 5 \rangle$



Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

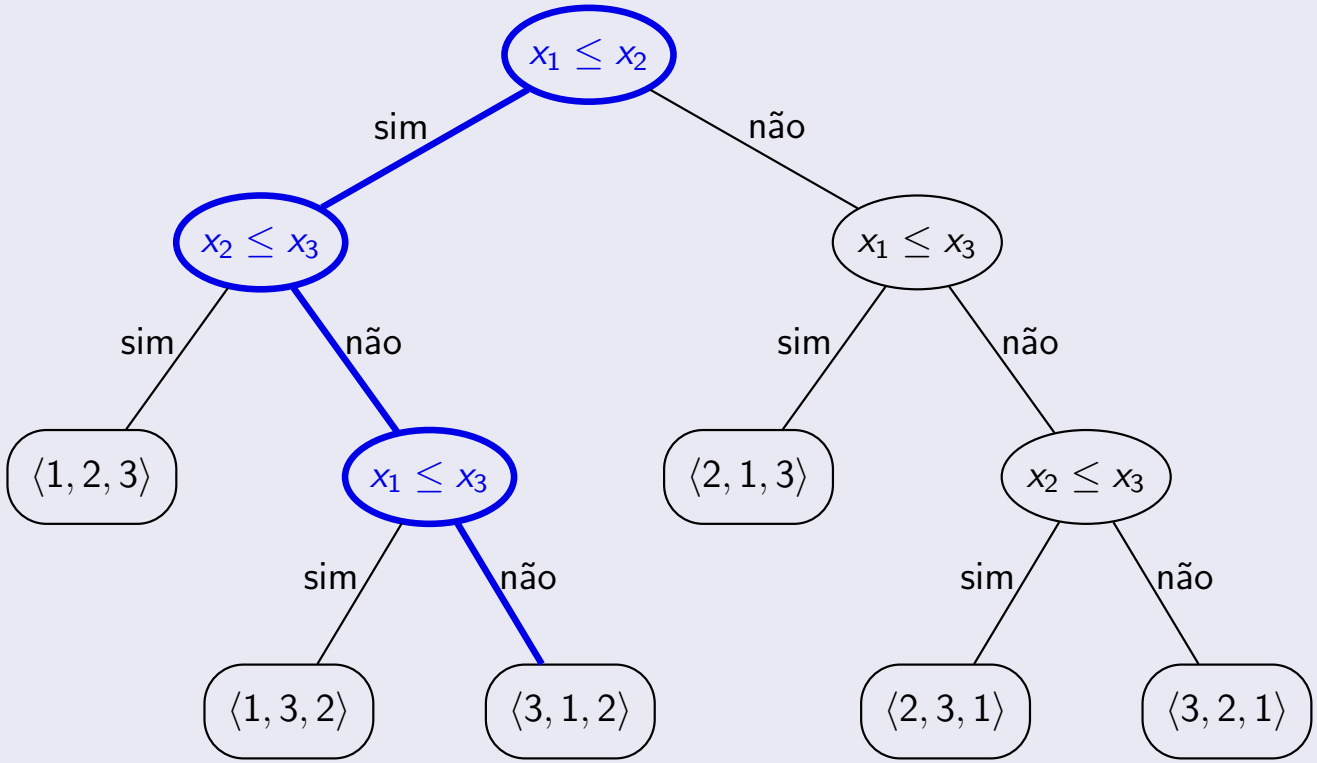
- Ex: Entrada $\langle x_1 = 6, x_2 = 8, x_3 = 5 \rangle$



Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

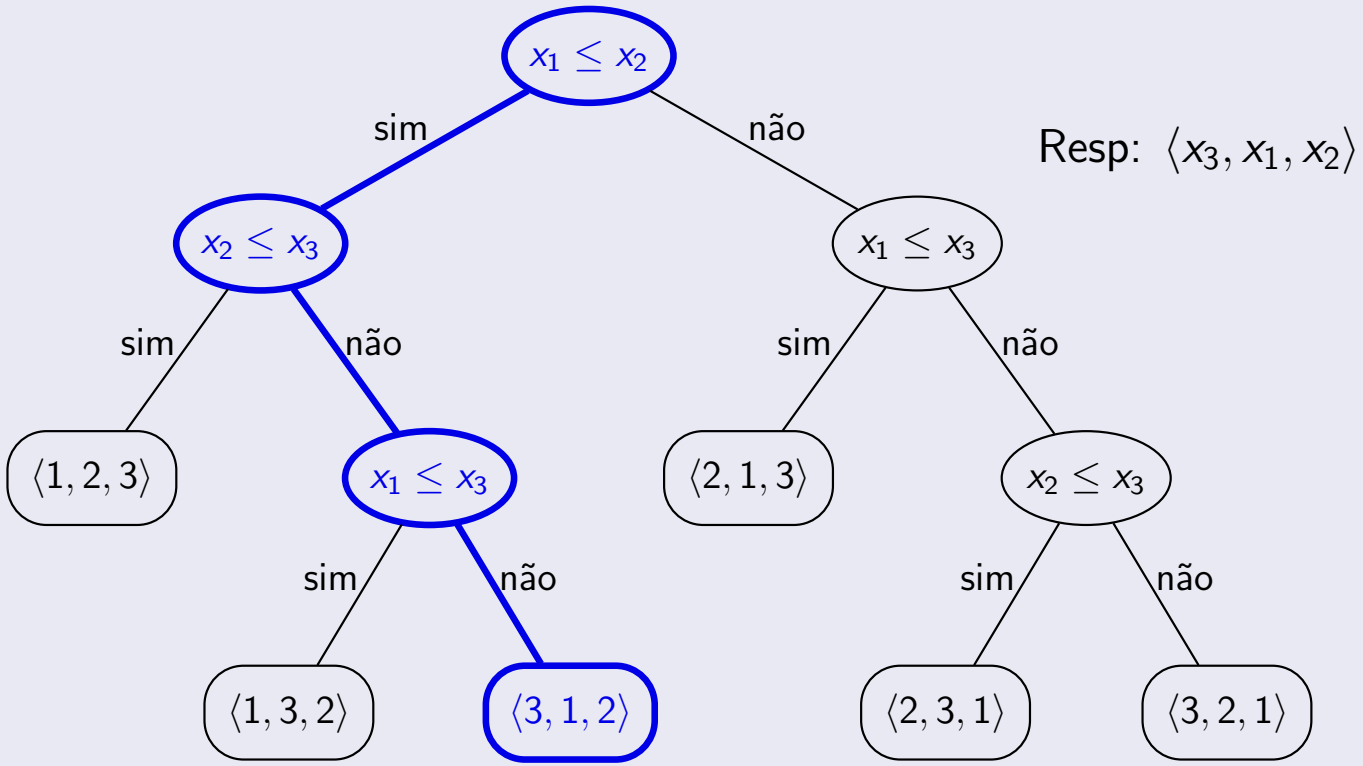
- Ex: Entrada $\langle x_1 = 6, x_2 = 8, x_3 = 5 \rangle$



Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

- Ex: Entrada $\langle x_1 = 6, x_2 = 8, x_3 = 5 \rangle$



Resp: $\langle x_3, x_1, x_2 \rangle$

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

- Ao representarmos um algoritmo de ordenação baseado em comparações por uma árvore binária de decisão, todas as permutações de n elementos devem ser possíveis soluções

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

- Ao representarmos um algoritmo de ordenação 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

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

- Ao representarmos um algoritmo de ordenação 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
  - Pode ter mais, pois nada impede que duas sequências distintas de decisões terminem no mesmo resultado

Ordenação: Cota Inferior

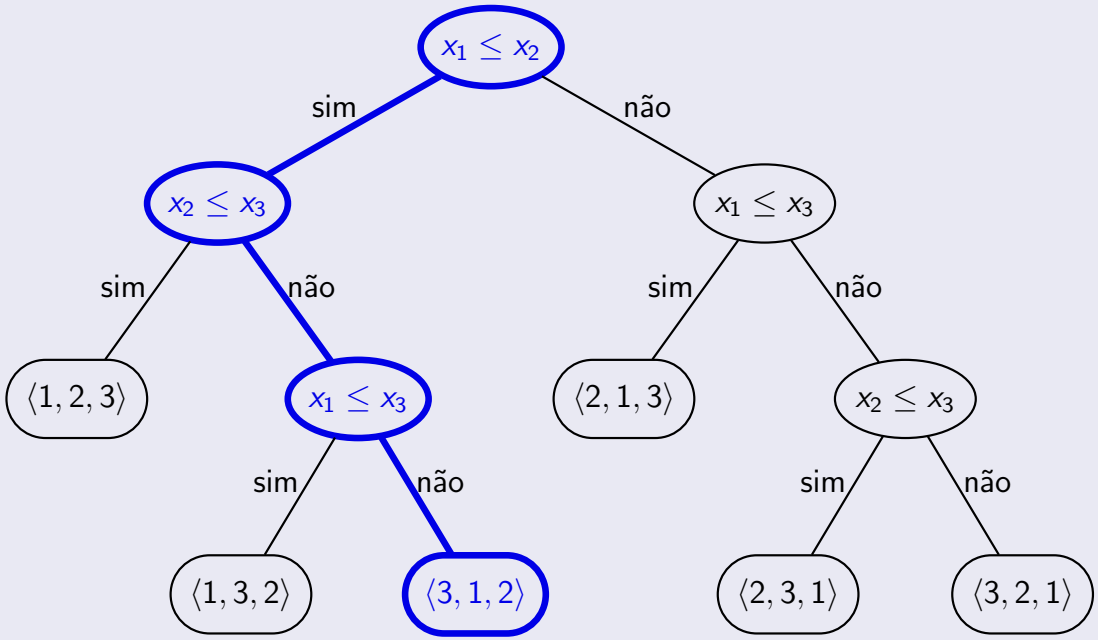
Árvores de Decisão: Ordenação

- A execução do algoritmo de ordenação corresponde a traçar um caminho da raiz a uma folha

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

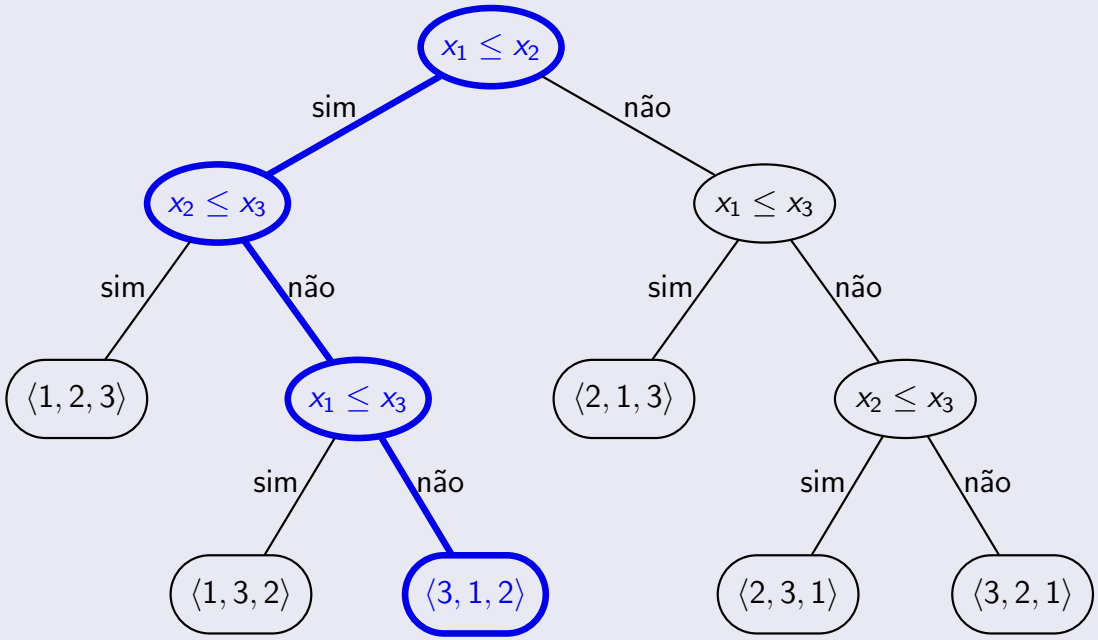
- A execução do algoritmo de ordenação corresponde a traçar um caminho da raiz a uma folha



Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

- A execução do algoritmo de ordenação corresponde a traçar um caminho da raiz a uma folha



  - O caminho mais longo da raiz a uma folha representa o **pior caso** de execução do algoritmo

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

- A altura de uma árvore binária de decisão representa o número de comparações, no pior caso, que o algoritmo por ela representado executa

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

- A altura de uma árvore binária de decisão representa o número de comparações, no pior caso, que o algoritmo por ela representado executa
  - A altura (h) de uma árvore é o comprimento do caminho mais longo da raiz até qualquer uma de suas folhas

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

- A altura de uma árvore binária de decisão representa o número de comparações, no pior caso, que o algoritmo por ela representado executa
  - A altura (h) de uma árvore é o comprimento do caminho mais longo da raiz até qualquer uma de suas folhas
- Então um limite inferior nas alturas de todas as árvores de decisão nas quais cada permutação aparece em uma folha é um limite inferior para o tempo de execução de qualquer algoritmo de ordenação baseado em comparações

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

- E qual é esse limite?

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

- E qual é esse limite?
- Na busca binária, vimos que uma árvore binária de decisão T com altura h tem, no máximo, 2^h folhas

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

- E qual é esse limite?
- Na busca binária, vimos que uma árvore binária de decisão T com altura h tem, no máximo, 2^h folhas
- Portanto, se T tem $n!$ folhas, então $n! \leq 2^h$, ou seja $h \geq log_2(n!)$

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

$$log_2(n!) = log_2(\prod_{i=1}^{n} i)$$

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

$$
\begin{aligned}
log_2(n!) &= log_2(\prod_{i=1}^{n} i) \\
&= \sum_{i=1}^{n} log_2(i)
\end{aligned}
$$

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

$$
\begin{aligned}
log_2(n!) &= log_2(\prod_{i=1}^{n} i) \\
&= \sum_{i=1}^{n} log_2(i) \\
&\geq \sum_{i=n/2}^{n} log_2(i)
\end{aligned}
$$

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

$$
\begin{aligned}
log_2(n!) &= log_2(\prod_{i=1}^{n} i) \\
&= \sum_{i=1}^{n} log_2(i) \\
&\geq \sum_{i=n/2}^{n} log_2(i) \\
&\geq \sum_{i=n/2}^{n} log_2(\frac{n}{2})
\end{aligned}
$$

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

$$
\begin{aligned}
log_2(n!) &= log_2(\prod_{i=1}^{n} i) \\
&= \sum_{i=1}^{n} log_2(i) \\
&\geq \sum_{i=n/2}^{n} log_2(i) \\
&\geq \sum_{i=n/2}^{n} log_2(\frac{n}{2}) \\
&= (n - \tfrac{n}{2}) log_2(\tfrac{n}{2})
\end{aligned}
$$

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

$$log_2(n!) \geq (n - \frac{n}{2})log_2(\frac{n}{2})$$

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

$$\begin{aligned} log_2(n!) &\geq (n - \tfrac{n}{2}) log_2(\tfrac{n}{2}) \\ &= \tfrac{n}{2} log_2(\tfrac{n}{2}) \end{aligned}$$

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

$$
\begin{aligned}
log_2(n!) &\geq (n - \tfrac{n}{2}) log_2(\tfrac{n}{2}) \\
&= \tfrac{n}{2} log_2(\tfrac{n}{2}) \\
&= \tfrac{n}{2}(log_2(n) - log_2 2)
\end{aligned}
$$

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

$$
\begin{aligned}
log_2(n!) &\geq (n - \tfrac{n}{2}) log_2(\tfrac{n}{2}) \\
&= \tfrac{n}{2} log_2(\tfrac{n}{2}) \\
&= \tfrac{n}{2}(log_2(n) - log_2 2) \\
&= \tfrac{n}{2}(log_2(n) - 1)
\end{aligned}
$$

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

$$
\begin{aligned}
log_2(n!) &\geq (n - \tfrac{n}{2}) log_2(\tfrac{n}{2}) \\
&= \tfrac{n}{2} log_2(\tfrac{n}{2}) \\
&= \tfrac{n}{2}(log_2(n) - log_2 2) \\
&= \tfrac{n}{2}(log_2(n) - 1) \\
&= \tfrac{n}{2} log_2(n) - \tfrac{n}{2}
\end{aligned}
$$

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

$$
\begin{aligned}
log_2(n!) &\geq (n - \tfrac{n}{2}) log_2(\tfrac{n}{2}) \\
&= \tfrac{n}{2} log_2(\tfrac{n}{2}) \\
&= \tfrac{n}{2}(log_2(n) - log_2 2) \\
&= \tfrac{n}{2}(log_2(n) - 1) \\
&= \tfrac{n}{2} log_2(n) - \tfrac{n}{2} \\
&= \Omega(n log_2(n)), n \geq 4, c = \tfrac{1}{4}
\end{aligned}
$$

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

- Portanto, $h = \Omega(nlog(n))$

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

- Portanto, $h = \Omega(nlog(n))$
  - Lembrando que a altura h representa o número de comparações, no pior caso, que o algoritmo executa

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

- Portanto, $h = \Omega(nlog(n))$
  - Lembrando que a altura h representa o número de comparações, no pior caso, que o algoritmo executa
- Temos então uma cota inferior para ordenação baseada em comparações

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

- Portanto, $h = \Omega(nlog(n))$
  - Lembrando que a altura h representa o número de comparações, no pior caso, que o algoritmo executa
- Temos então uma cota inferior para ordenação baseada em comparações
  - Mas nossos algoritmos, até agora, são $\Theta(n^2) \Rightarrow \Omega(n^2)$

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

- Portanto, $h = \Omega(nlog(n))$
  - Lembrando que a altura h representa o número de comparações, no pior caso, que o algoritmo executa
- Temos então uma cota inferior para ordenação baseada em comparações
  - Mas nossos algoritmos, até agora, são $\Theta(n^2) \Rightarrow \Omega(n^2)$
- Haveria algum algoritmo de ordenação, baseado em comparações, que seja $\Theta(nlog(n))$?

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

- Portanto, $h = \Omega(nlog(n))$
  - Lembrando que a altura h representa o número de comparações, no pior caso, que o algoritmo executa
- Temos então uma cota inferior para ordenação baseada em comparações
  - Mas nossos algoritmos, até agora, são $\Theta(n^2) \Rightarrow \Omega(n^2)$
- Haveria algum algoritmo de ordenação, baseado em comparações, que seja $\Theta(nlog(n))$?
  - Ou seja, um algoritmo ótimo

Ordenação: Cota Inferior

Árvores de Decisão: Ordenação

- Portanto, $h = \Omega(nlog(n))$
  - Lembrando que a altura h representa o número de comparações, no pior caso, que o algoritmo executa
- Temos então uma cota inferior para ordenação baseada em comparações
  - Mas nossos algoritmos, até agora, são $\Theta(n^2) \Rightarrow \Omega(n^2)$
- Haveria algum algoritmo de ordenação, baseado em comparações, que seja $\Theta(nlog(n))$?
  - Ou seja, um algoritmo ótimo
  - Na próxima aula...

Referências

- Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford. Introduction to Algorithms. 2a ed. MIT Press, 2001.

- Material baseado em slides dos professores Cid de Souza, Cândida da Silva e Delano Beder