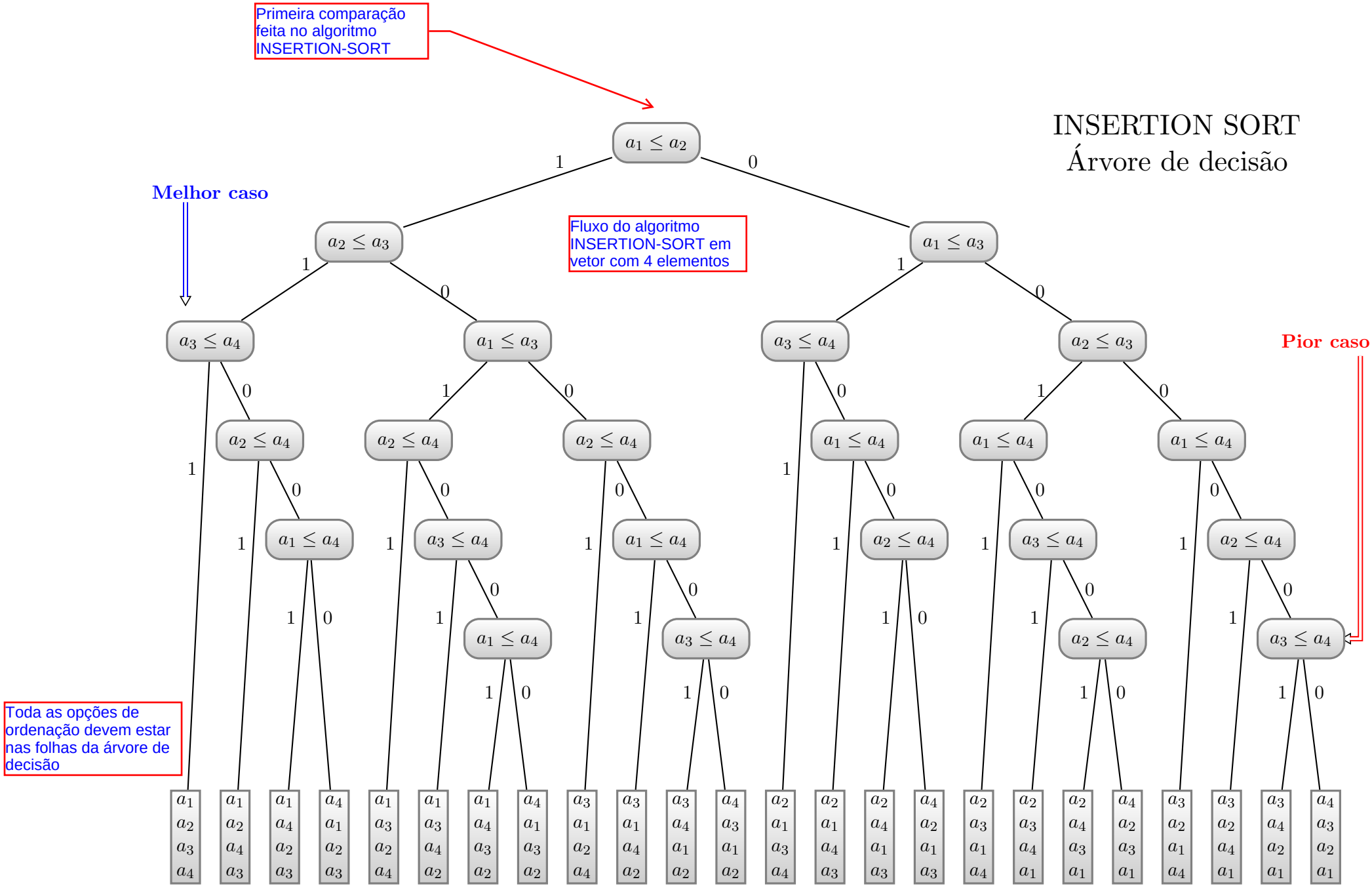
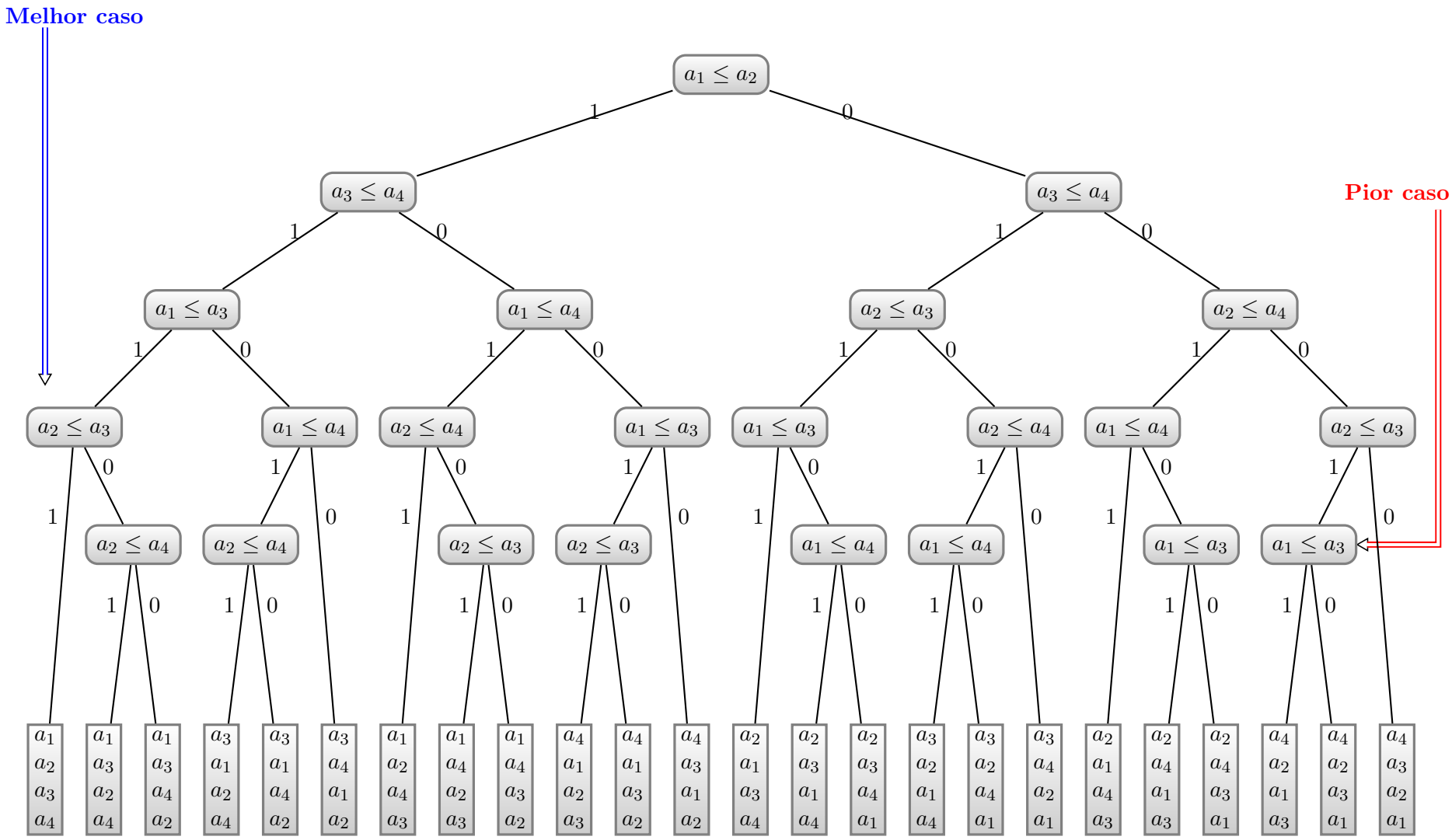

Primeira comparação feita no algoritmo INSERTION-SORT
INSERTION SORT
Árvore de decisão
Melhor caso
Fluxo do algoritmo INSERTION-SORT em vetor com 4 elementos
Pior caso
Toda as opções de ordenação devem estar nas folhas da árvore de decisão
$a_1 \leq a_2$
$a_2 \leq a_3$
$a_1 \leq a_3$
$a_3 \leq a_4$
$a_1 \leq a_3$
$a_3 \leq a_4$
$a_2 \leq a_3$
$a_2 \leq a_4$
$a_2 \leq a_4$
$a_2 \leq a_4$
$a_1 \leq a_4$
$a_1 \leq a_4$
$a_1 \leq a_4$
$a_1 \leq a_4$
$a_1 \leq a_4$
$a_3 \leq a_4$
$a_1 \leq a_4$
$a_3 \leq a_4$
$a_2 \leq a_4$
$a_3 \leq a_4$
1
0

MERGE SORT
Árvore de decisão



Existem muitos algoritmos de ordenação de um vetor. Alguns tem complexidade de pior caso $\Theta(n^2)$ (como o Insertion-Sort) e outros tem complexidade de pior caso $\Theta(n \log n)$ (como o Merge-Sort). Surge então a seguinte questão: "*Existe algum algoritmo de ordenação com complexidade de pior caso melhor que estes?*". A resposta é "NÃO", considerando os algoritmos clássicos, baseados em comparações. Dizemos que um algoritmo de ordenação baseia-se em **comparações** se, para vetores de tamanho n, o fluxo do algoritmo depende apenas do resultado de comparações entre elementos do vetor. Os algoritmos mais conhecidos, como Insertion-Sort, Bubble-Sort, Merge-Sort, Heap-Sort e Quick-Sort, são todos baseados em comparações. Como exemplos de algoritmo de ordenação que NÃO são baseados em comparações, citamos o Counting-Sort e o Radix-Sort, que são baseados em **contagem** e funcionam apenas sobre condições bem restritas. Considere então qualquer algoritmo de ordenação baseado em comparações executado sobre um vetor $(a_1, a_2, \ldots, a_n)$. Para simplificar, considere também que os elementos do vetor são todos diferentes entre si.

Neste caso, podemos assumir, sem perda de generalidade, que todas as comparações feitas pelo algoritmo são do tipo "$a_i \leq a_j$". A árvore de decisão de um tal algoritmo (definida para cada valor de n) é uma árvore binária cujos nós internos estão rotulados por pares de elementos do vetor de entrada, denotados por exemplo por $a_i \leq a_j$. Cada nó interno ($a_i \leq a_j$) tem dois filhos. As arestas de um nó interno para os seus filhos estão rotuladas uma por 1 (verdadeiro) para o filho esquerdo e a outra por 0 (falso) para o filho direito. Cada folha está rotulada por uma permutação de $a_1, a_2, \ldots, a_n$. A árvore de decisão está associada ao algoritmo da seguinte maneira. O rótulo da raiz da árvore corresponde à primeira comparação efetuada pelo algoritmo quando a entrada é um vetor $a_1, a_2, \ldots, a_n$. A subárvore esquerda da raiz descreve as comparações subsequentes, caso o resultado desta primeira comparação, digamos, $a_i \leq a_j$, seja verdadeiro. Já a subárvore direita descreve as comparações subsequentes caso o resultado da comparação $a_i \leq a_j$ seja falso. Com isso, cada vetor $a_1, a_2, \ldots, a_n$ corresponde a um caminho da raiz até uma folha da árvore de decisão. A permutacão que rotula essa folha é exatamente a permutação que deixa o vetor $a_1, a_2, \ldots, a_n$ ordenado.

Veja os exemplos das páginas anteriores contendo as árvores de decisão dos algoritmos Insertion-Sort e Merge-Sort para uma entrada com quatro elementos (a_1, a_2, a_3, a_4). Note que cada uma das $4! = 24$ permutações aparece nas folhas. Isso é de se esperar mesmo para um n arbitrário. Ou seja, cada uma das $n!$ permutações deve aparecer como rótulo de uma das folhas na árvore de decisão. Afinal, para cada permutação específica, há pelo menos uma sequência de entrada que é ordenada apenas por aquela permutação. Se uma permutação não aparecesse, então o algoritmo não ordenaria a correspondente sequência de entrada.

Observe também que o número de comparações que o algoritmo faz no pior caso é exatamente a altura da árvore de decisão (ou seja, a maior distância de uma folha para a raiz). Portanto, se calcularmos um limite inferior para a altura da árvore de decisão do algoritmo, temos uma delimitação inferior para a complexidade de pior caso do algoritmo. Isso é o que vamos fazer.

Existem $n!$ permutações de $(a_1, a_2, \ldots, a_n)$. Cada uma delas deve aparecer em alguma das folhas da árvore de decisão. Portanto, a árvore de decisão deve ter pelo menos $n!$ folhas. Por outro lado, uma árvore binária de altura h tem no máximo 2^h folhas. Assim, se h é a altura da árvore de decisão de um algoritmo de ordenação baseado em comparações, então $2^h \geq n!$. Sabemos, pela fórmula de Stirling, que $n! \geq \left(\frac{n}{e}\right)^n$, onde e é o número de Euler $e = 2,718281828459\ldots$. Portanto,

$$h \geq \log_2(n!) \geq \log_2 \left(\frac{n}{e}\right)^n = n \log_2 \left(\frac{n}{e}\right) = n \log_2 n - n \log_2 e = \Omega(n \log n).$$

Portanto, de fato, qualquer algoritmo de ordenação baseado em comparações tem complexidade de pior caso $\Omega(n \log n)$.

Ordenação: limite inferior

Problema: Rearranjar um vetor $A[1\,..\,n]$ de modo que ele fique em ordem crescente.

Existem algoritmos que consomem tempo $\mathrm{O}(n \lg n)$.

Ordenação: limite inferior

Problema: Rearranjar um vetor $A[1\,..\,n]$ de modo que ele fique em ordem crescente.

Existem algoritmos que consomem tempo $\mathrm{O}(n \lg n)$.

Existe algoritmo assintoticamente melhor?

Ordenação: limite inferior

Problema: Rearranjar um vetor $A[1\,..\,n]$ de modo que ele fique em ordem crescente.

Existem algoritmos que consomem tempo $\mathrm{O}(n \lg n)$.

Existe algoritmo assintoticamente melhor?

NÃO, se o algoritmo é baseado em comparações.

Prova?

Ordenação: limite inferior

Problema: Rearranjar um vetor $A[1\,..\,n]$ de modo que ele fique em ordem crescente.

Existem algoritmos que consomem tempo $\mathrm{O}(n \lg n)$.

Existe algoritmo assintoticamente melhor?

NÃO, se o algoritmo é baseado em comparações.

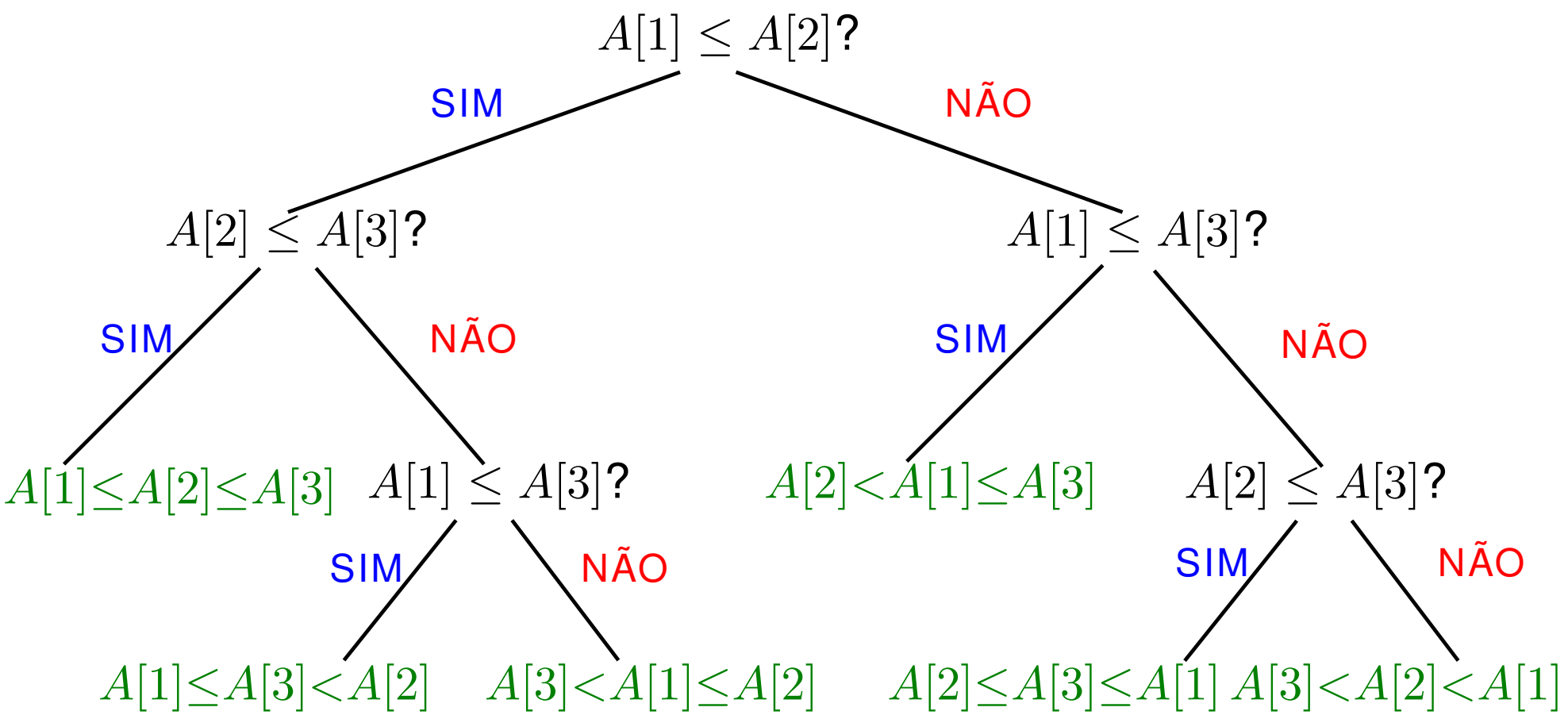
Prova?

Qualquer algoritmo baseado em comparações é uma "árvore de decisão".

Exemplo

INSERTION-SORT

ORDENA-POR-INSERÇÃO $(A[1\,..\,3])$:



Limite inferior

Considere uma árvore de decisão para $A[1\,..\,n]$.

Limite inferior

Considere uma árvore de decisão para $A[1\,..\,n]$.

Número de comparações, no pior caso?

Limite inferior

Considere uma árvore de decisão para $A[1\,..\,n]$.

Número de comparações, no pior caso?
Resposta: altura, h, da árvore de decisão.

Limite inferior

Considere uma árvore de decisão para $A[1\,..\,n]$.

Número de comparações, no pior caso?
Resposta: altura, h, da árvore de decisão.

Todas as $n!$ permutações de $1, \ldots, n$ devem ser folhas.

Limite inferior

Considere uma árvore de decisão para $A[1\,..\,n]$.

Número de comparações, no pior caso?
Resposta: altura, h, da árvore de decisão.

Todas as $n!$ permutações de $1, \ldots, n$ devem ser folhas.

Toda árvore binária de altura h tem no máximo 2^h folhas.

Limite inferior

Considere uma árvore de decisão para $A[1\,..\,n]$.

Número de comparações, no pior caso?
Resposta: altura, h, da árvore de decisão.

Todas as $n!$ permutações de $1, \ldots, n$ devem ser folhas.

Toda árvore binária de altura h tem no máximo 2^h folhas.

Prova: Por indução em h. A afirmação vale para $h = 0$.

Limite inferior

Considere uma árvore de decisão para $A[1\,..\,n]$.

Número de comparações, no pior caso?
Resposta: altura, h, da árvore de decisão.

Todas as $n!$ permutações de $1, \ldots, n$ devem ser folhas.

Toda árvore binária de altura h tem no máximo 2^h folhas.

Prova: Por indução em h. A afirmação vale para $h = 0$.
Suponha que a afirmação vale para toda árvore binária de altura menor que h, para $h \geq 1$.

Limite inferior

Considere uma árvore de decisão para $A[1\,..\,n]$.

Número de comparações, no pior caso?
Resposta: altura, h, da árvore de decisão.

Todas as $n!$ permutações de $1, \ldots, n$ devem ser folhas.

Toda árvore binária de altura h tem no máximo 2^h folhas.

Prova: Por indução em h. A afirmação vale para $h = 0$.
Suponha que a afirmação vale para toda árvore binária de altura menor que h, para $h \geq 1$.
Número de folhas de árvore de altura h é a soma do número de folhas das subárvores, que têm altura $\leq h - 1$.

Limite inferior

Considere uma árvore de decisão para $A[1\,..\,n]$.

Número de comparações, no pior caso?
Resposta: altura, h, da árvore de decisão.

Todas as $n!$ permutações de $1, \ldots, n$ devem ser folhas.

Toda árvore binária de altura h tem no máximo 2^h folhas.

Prova: Por indução em h. A afirmação vale para $h = 0$.
Suponha que a afirmação vale para toda árvore binária de altura menor que h, para $h \geq 1$.
Número de folhas de árvore de altura h é a soma do número de folhas das subárvores, que têm altura $\leq h - 1$.
Logo, o número de folhas de uma árvore de altura h é

$$\leq \; 2 \times 2^{h-1} \; = \; 2^h.$$

Limite inferior

Assim, devemos ter $2^h \geq n!$, donde $h \geq \lg(n!)$.

Limite inferior

Assim, devemos ter $2^h \geq n!$, donde $h \geq \lg(n!)$.

$$(n!)^2 = \prod_{i=0}^{n-1} (n-i)(i+1) \geq \prod_{i=1}^{n} n = n^n$$

Um modo de provar sem usar a fórmula de STIRLING para fatoriais

Limite inferior

Assim, devemos ter $2^h \geq n!$, donde $h \geq \lg(n!)$.

$$(n!)^2 = \prod_{i=0}^{n-1} (n-i)(i+1) \geq \prod_{i=1}^{n} n = n^n$$

Portanto,

$$h \geq \lg(n!) \geq \frac{1}{2} n \lg n.$$

Limite inferior

Assim, devemos ter $2^h \geq n!$, donde $h \geq \lg(n!)$.

$$(n!)^2 \;=\; \prod_{i=0}^{n-1} (n-i)(i+1) \;\geq\; \prod_{i=1}^{n} n \;=\; n^n$$

Portanto,

$$h \;\geq\; \lg(n!) \;\geq\; \frac{1}{2}\, n \lg n.$$

Agora usando a fórmula de STIRLING para fatoriais

Alternativamente, a fórmula de Stirling diz que

$$n! \;=\; \sqrt{2\pi\, n}\left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right).$$

Limite inferior

Assim, devemos ter $2^h \geq n!$, donde $h \geq \lg(n!)$.

$$(n!)^2 \;=\; \prod_{i=0}^{n-1} (n-i)(i+1) \;\geq\; \prod_{i=1}^{n} n \;=\; n^n$$

Portanto,

$$h \;\geq\; \lg(n!) \;\geq\; \frac{1}{2}\, n \lg n.$$

Alternativamente, a fórmula de Stirling diz que

$$n! \;=\; \sqrt{2\pi\, n}\big(\frac{n}{e}\big)^n (1 + \Theta(\frac{1}{n})).$$

Disso, temos que $h \;\geq\; \lg(n!) \;\geq\; \lg\left(\frac{n}{e}\right)^n \;=\; n(\lg n - \lg e)$.

Conclusão

Todo algoritmo de ordenação
baseado em comparações faz

$$\Omega(n \lg n)$$

comparações no pior caso.

Mas existem algoritmos de ordenação em tempo linear O(n), que não são por comparação e que funcionam sob certas condições restritas do vetor da entrada