Limitante Inferior da Ordenação

Limitante Inferior para a Ordenação

- Em um algoritmo de ordenação por comparação, usamos apenas a comparação entre os elementos para ganhar informação sobre a ordem de uma sequência de entrada $\langle a_1, a_2, \ldots, a_n \rangle$
  ↳ vetor
  - Dados dois elementos a_i, a_j, fazemos os testes
    $a_i < a_j$, $a_i \leq a_j$, $a_i = a_j$, $a_i > a_j$ ou $a_i \geq a_j$
    para determinar a ordem relativa entre eles.

Limitante Inferior para a Ordenação

- Nesta aula vamos provar que se $T(n)$ é o tempo de um algoritmo de ordenação por comparação, então

$$T(n) \in \Omega(n \lg n)$$

⇒ Pare e aprecie essa informação

⇒ Qualquer algoritmo

⇒ Inclusive os que ainda não foram inventados

Limitante Inferior para a Ordenação

- Suponha que todos os elementos da entrada são distintos.
  - ⇒ isso implica que comparações da forma $a_i = a_j$ são inúteis.
  - ⇒ Então vamos assumir que nenhuma operação desse tipo ocorre.
  - ⇒ tbm podemos notar que comparações $a_i \leq a_j$, $a_i \geq a_j$, $a_i > a_j$ e $a_i < a_j$ nos dão a mesma informação (são equivalentes)
    - ⇒ Podemos supor que temos apenas operações da forma $a_i < a_j$.

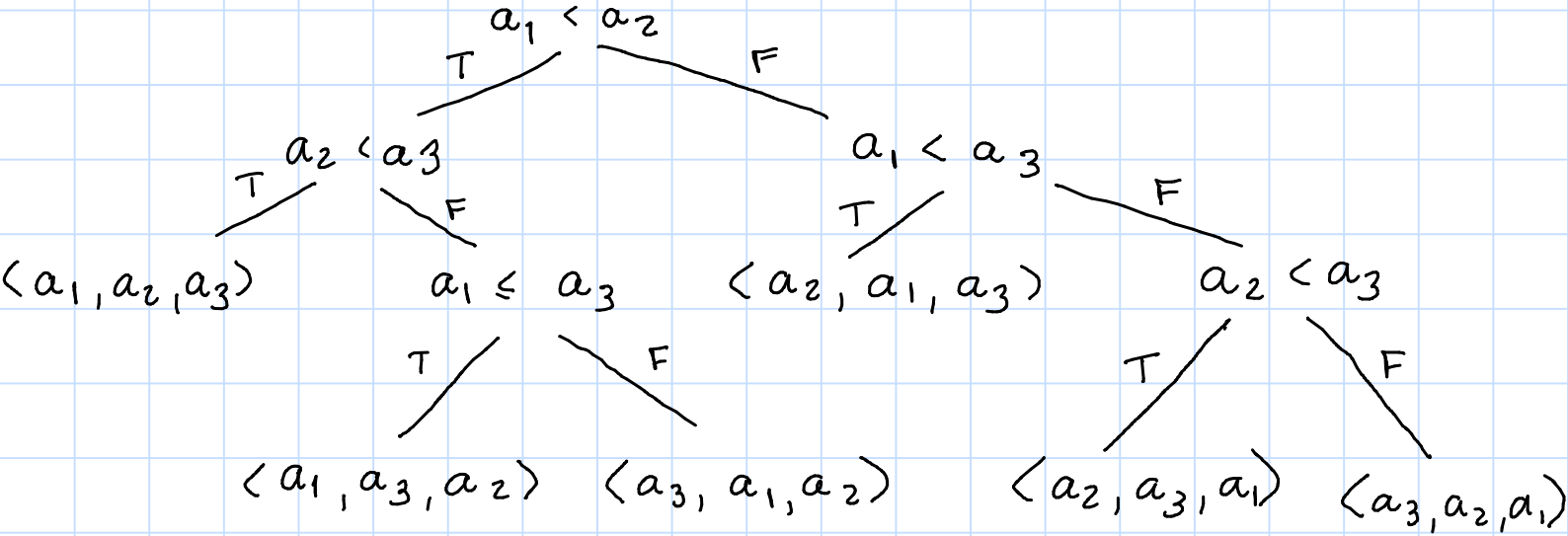
Modelo Árvore de Decisão

- Podemos ver um algoritmo de ordenação por comparação em termos de árvores de decisão.
- Uma <u>árvore de decisão</u> é uma árvore binária que representa a comparação entre os dementos que é executada pelo algoritmo de ordenação.

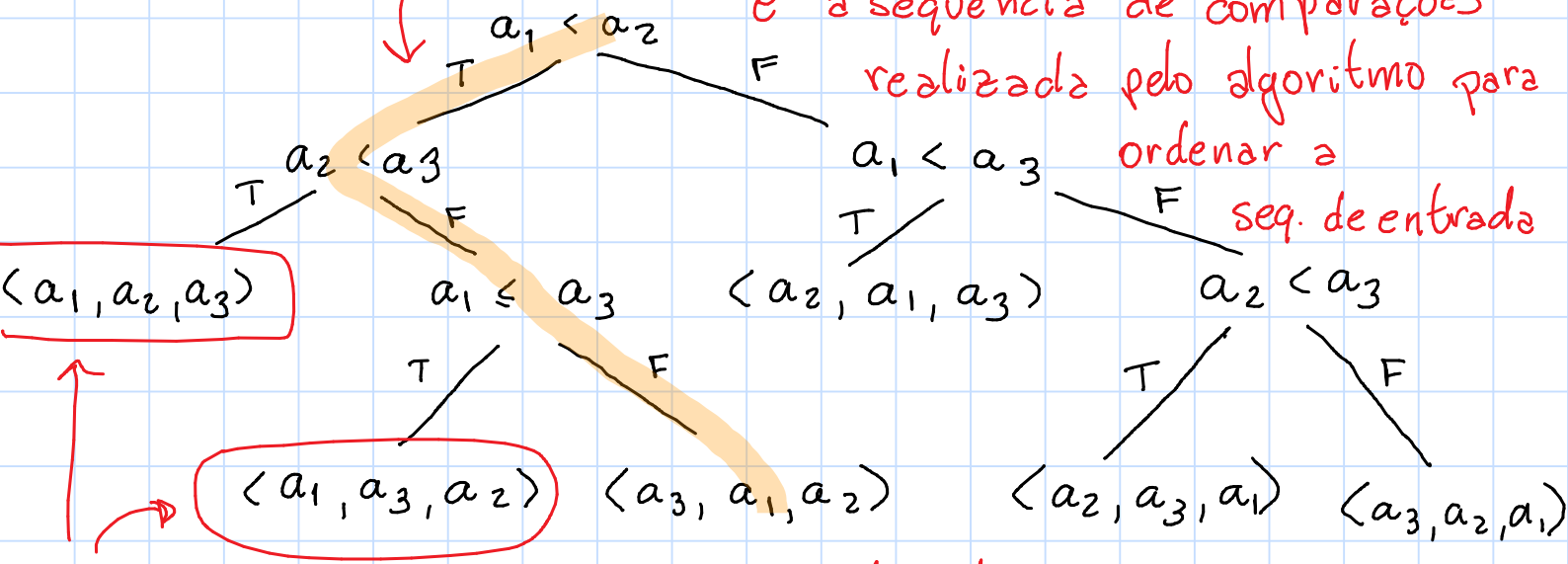
⇒ Controle de fluxo, movimentação de dados e todo o resto são ignorados

⇒ Vamos mostrar que só o custo com comparações já é $\Omega(n \lg n)$.

Modelo Árvore de Decisão



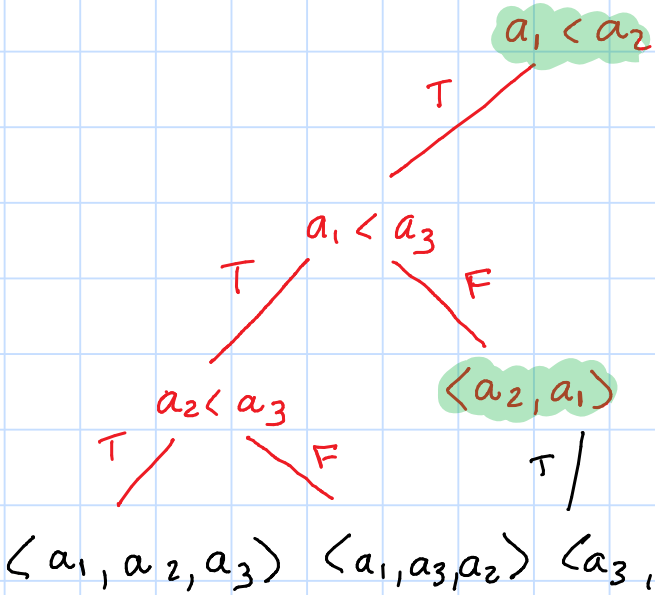
Modelo Árvore de Decisão

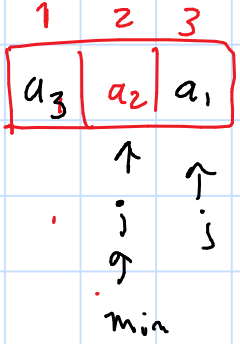


② um caminho da raiz até uma folha é a sequência de comparações realizada pelo algoritmo para ordenar a seq. de entrada

① folhas são as sequências ordenadas

③ Como o algoritmo tem que funcionar para qualquer uma das $n!$ sequências de entrada, temos ao menos essa quantidade de folhas.






```
SelectionSort(A, n)
    Para i ← 1 até n-1
        min ← i
        para j ← i+1 até n
            Se A[min] > A[j]
                min ← j
        A[i] ⇔ A[min]
```

Limitante Inferior para a Ordenação

⇒ Seja T essa árvore de decisão gerada para um algoritmo de ordenação por comparação arbitrário X para uma instância de tamanho n.

⇒ A altura da raiz de T representa o pior caso no número de comparações.

⇒ Para obter o que queremos, mostraremos que a altura de T é $\Omega(n \lg n)$.

Teo Qualquer algoritmo de ordenação por comparação requer

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

comparações no pior caso.

Demo.

- Seja T a árvore binária de decisão de um algoritmo por comparação X para uma instância de tamanho n.
- Como discutido antes, T tem ao menos $n!$ folhas
- Uma árvore binária de altura h tem no máximo 2^h folhas
- Se l é a altura de T, então temos que

$$2^l \geq n!$$

- Logo $l \geq \lg n!$

- Agora vamos provar que $\lg(n!) \in \Omega(n \lg n)$
- Primeiro observe que $n! \geq \lceil \frac{n}{2} \rceil^{\lceil \frac{n}{2} \rceil} \geq (\frac{n}{2})^{\frac{n}{2}}$, pois ao menos $\lceil \frac{n}{2} \rceil$ temos em $n \cdot (n-1)(n-2) \cdots 2 \cdot 1$ são maiores ou iguais a $\lceil \frac{n}{2} \rceil$
- Agora mostramos que $\lg (\frac{n}{2})^{\frac{n}{2}} \in \Omega(n \lg n)$, que implica no resultado desejado pois

$$\lg(n!) \geq \lg (\tfrac{n}{2})^{\frac{n}{2}} \geq C\, n \lg n \quad \forall\, n \geq n_0$$

- Para isso, usaremos limites

$$\lim_{n \to \infty} \frac{\lg (\frac{n}{2})^{\frac{n}{2}}}{n \lg n} = \lim_{n \to \infty} \frac{\frac{n}{2} \lg (\frac{n}{2})}{n \lg n} = \lim_{n \to \infty} \frac{\frac{n}{2} [\lg n - \lg 2]}{n \lg n}$$

$$= \lim_{n \to \infty} \left[\frac{\cancel{n \lg n}}{2 \cancel{n \lg n}} - \frac{\cancel{n}}{2 \cancel{n} \lg n} \right] = \lim_{n \to \infty} \left[\frac{1}{2} - \frac{1}{2 \lg n} \right] = \frac{1}{2}$$

- Logo, pelo teorema dos limites, temos que $\lg\left(\frac{n}{2}\right)^{\frac{n}{2}} \in \Theta(n \lg n)$ e, por conseguinte, $\lg\left(\frac{n}{2}\right)^{\frac{n}{2}} \in \Omega(n \lg n)$.

- Assim, concluímos que a altura ℓ da árvore de decisão é $\Omega(n \lg n)$. □

Cor. Os algoritmos HeapSort e MergeSort são assintóticamente ótimos.