Kapitel ADS:III

III. Sortieren

- Sortieralgorithmen
- Insertion Sort
- Heapsort
- Merge Sort
- Quicksort
- Counting Sort
- Radix Sort
- Bucket Sort
- Minimales vergleichsbasiertes Sortieren

Minimales vergleichsbasiertes Sortieren

Entscheidungsbaummodell

Was ist die kleinstmögliche Maximalzahl an Vergleichen, um n Zahlen zu sortieren?

Beispiel $n = 3$:

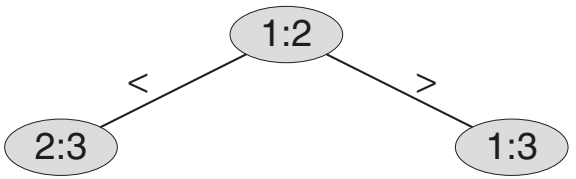
- Darstellung des Vorgehens eines hypothetischen Algorithmus.
- $i : j$ steht für den Vergleich von $A[i] = a_i$ mit $A[j] = a_j$.

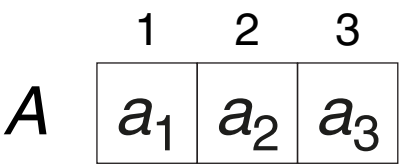
Minimales vergleichsbasiertes Sortieren

Entscheidungsbaummodell

Was ist die kleinstmögliche Maximalzahl an Vergleichen, um n Zahlen zu sortieren?

Beispiel $n = 3$:





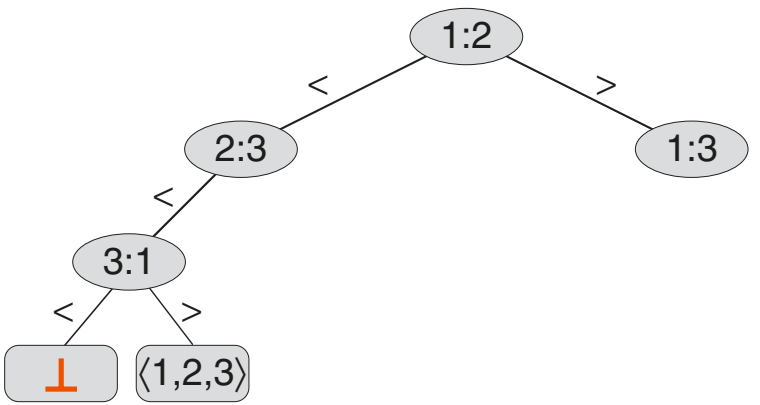
- Darstellung des Vorgehens eines hypothetischen Algorithmus.
- $i : j$ steht für den Vergleich von $A[i] = a_i$ mit $A[j] = a_j$.

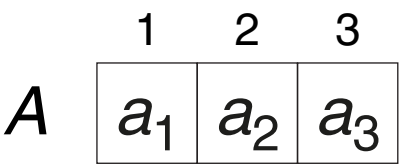
Minimales vergleichsbasiertes Sortieren

Entscheidungsbaummodell

Was ist die kleinstmögliche Maximalzahl an Vergleichen, um n Zahlen zu sortieren?

Beispiel $n = 3$:





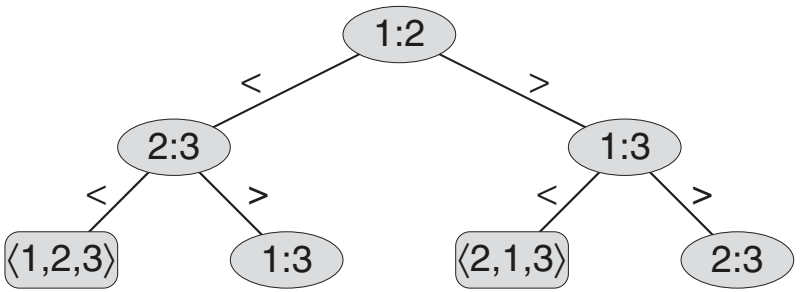
- Darstellung des Vorgehens eines hypothetischen Algorithmus.
- $i : j$ steht für den Vergleich von $A[i] = a_i$ mit $A[j] = a_j$.
- Unmögliche Situation: $a_1 < a_2 < a_3 < a_1$.
- Redundanter Vergleich (Transitivität): Aus $a_1 < a_2$ und $a_2 < a_3$ folgt $a_1 < a_3$.
- Blattknoten $\langle i, j, k \rangle$ stehen für Permutationen der Probleminstanz $[a_i, a_j, a_k]$.

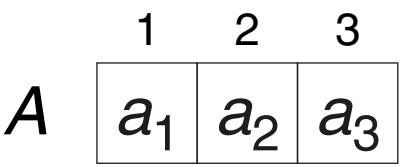
Minimales vergleichsbasiertes Sortieren

Entscheidungsbaummodell

Was ist die kleinstmögliche Maximalzahl an Vergleichen, um n Zahlen zu sortieren?

Beispiel $n = 3$:





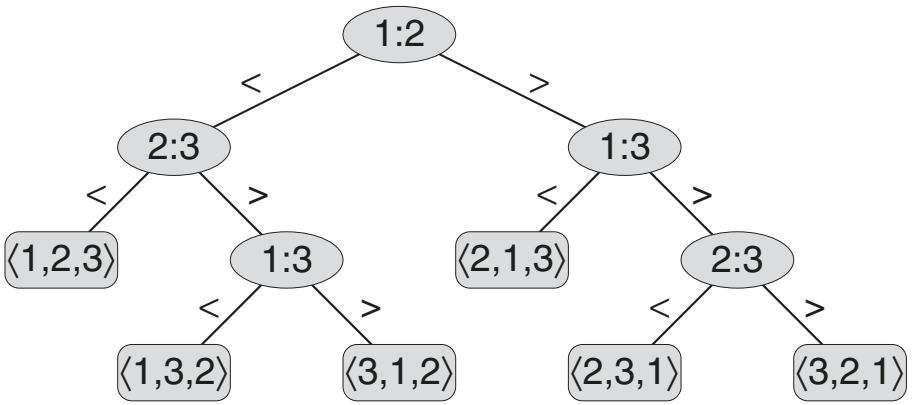
- Darstellung des Vorgehens eines hypothetischen Algorithmus.
- $i : j$ steht für den Vergleich von $A[i] = a_i$ mit $A[j] = a_j$.
- Blattknoten $\langle i, j, k \rangle$ stehen für Permutationen der Probleminstanz $[a_i, a_j, a_k]$.

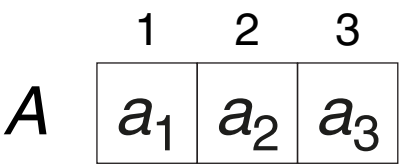
Minimales vergleichsbasiertes Sortieren

Entscheidungsbaummodell

Was ist die kleinstmögliche Maximalzahl an Vergleichen, um n Zahlen zu sortieren?

Beispiel $n = 3$:





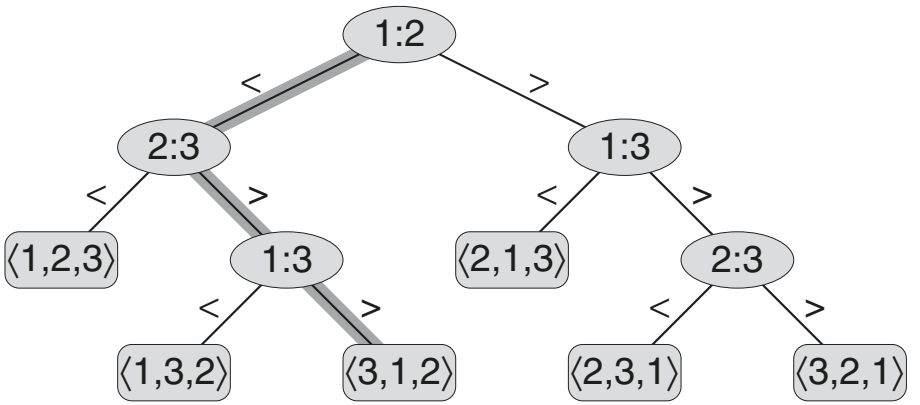
- Darstellung des Vorgehens eines hypothetischen Algorithmus.
- $i : j$ steht für den Vergleich von $A[i] = a_i$ mit $A[j] = a_j$.
- Blattknoten $\langle i, j, k \rangle$ stehen für Permutationen der Probleminstanz $[a_i, a_j, a_k]$.
- Es gibt mindestens $n!$ Blattknoten; mind. einen für jede mögliche Permutation.

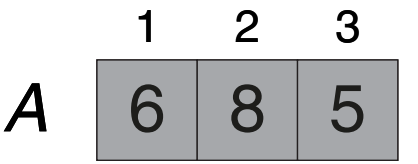
Minimales vergleichsbasiertes Sortieren

Entscheidungsbaummodell

Was ist die kleinstmögliche Maximalzahl an Vergleichen, um n Zahlen zu sortieren?

Beispiel $n = 3$:





- Darstellung des Vorgehens eines hypothetischen Algorithmus.
- $i : j$ steht für den Vergleich von $A[i] = a_i$ mit $A[j] = a_j$.
- Blattknoten $\langle i, j, k \rangle$ stehen für Permutationen der Probleminstanz $[a_i, a_j, a_k]$.
- Es gibt mindestens $n!$ Blattknoten; mind. einen für jede mögliche Permutation.
- Sortieren bedeutet, den Pfad zur richtigen Permutation zu ermitteln.

Minimales vergleichsbasiertes Sortieren

Satz 1 (Untere Schranke für den Worst Case für vergleichsbasiertes Sortieren)

Jeder vergleichsbasierte Sortieralgorithmus benötigt im Worst-Case $\Omega(n \lg n)$ Vergleiche.

Minimales vergleichsbasiertes Sortieren

Satz 1 (Untere Schranke für den Worst Case für vergleichsbasiertes Sortieren)

Jeder vergleichsbasierte Sortieralgorithmus benötigt im Worst-Case $\Omega(n \lg n)$ Vergleiche.

Beweis:

Für das Sortieren von n Elementen gibt es einen Entscheidungsbaum der Höhe h mit l Blattknoten. Da alle $n!$ möglichen Permutationen vorkommen müssen, gilt:

$$n! \leq l \leq 2^h$$

Minimales vergleichsbasiertes Sortieren

Satz 1 (Untere Schranke für den Worst Case für vergleichsbasiertes Sortieren)

Jeder vergleichsbasierte Sortieralgorithmus benötigt im Worst-Case $\Omega(n \lg n)$ Vergleiche.

Beweis:

Für das Sortieren von n Elementen gibt es einen Entscheidungsbaum der Höhe h mit l Blattknoten. Da alle $n!$ möglichen Permutationen vorkommen müssen, gilt:

$$
\begin{aligned}
& n! \leq l \leq 2^h \\
\Leftrightarrow \quad & 2^h \geq n! \\
\Leftrightarrow \quad & h \geq \lg(n!)
\end{aligned}
$$

Minimales vergleichsbasiertes Sortieren

Satz 1 (Untere Schranke für den Worst Case für vergleichsbasiertes Sortieren)

Jeder vergleichsbasierte Sortieralgorithmus benötigt im Worst-Case $\Omega(n \lg n)$ Vergleiche.

Beweis:

Für das Sortieren von n Elementen gibt es einen Entscheidungsbaum der Höhe h mit l Blattknoten. Da alle $n!$ möglichen Permutationen vorkommen müssen, gilt:

$$
\begin{aligned}
& n! \leq l \leq 2^h \\
\Leftrightarrow \quad & 2^h \geq n! \\
\Leftrightarrow \quad & h \geq \lg(n!) \\
& = \lg((n/e)^n) \qquad \text{(Stirling-Formel)} \\
& = n \lg((n/e)) \\
& = n \lg n - n \lg e
\end{aligned}
$$

Minimales vergleichsbasiertes Sortieren

Satz 1 (Untere Schranke für den Worst Case für vergleichsbasiertes Sortieren)

Jeder vergleichsbasierte Sortieralgorithmus benötigt im Worst-Case $\Omega(n \lg n)$ Vergleiche.

Beweis:

Für das Sortieren von n Elementen gibt es einen Entscheidungsbaum der Höhe h mit l Blattknoten. Da alle $n!$ möglichen Permutationen vorkommen müssen, gilt:

$$
\begin{aligned}
& n! \leq l \leq 2^h \\
\Leftrightarrow \quad & 2^h \geq n! \\
\Leftrightarrow \quad & h \geq \lg(n!) \\
& = \lg((n/e)^n) \qquad \text{(Stirling-Formel)} \\
& = n \lg((n/e)) \\
& = n \lg n - n \lg e \\
& = \Omega(n \lg n) \qquad \square
\end{aligned}
$$

Bemerkungen:

- Es werden nur Algorithmen betrachtet, die vergleichsbasiert sortieren.
- Es werden nur Probleminstanzen betrachtet, in denen n verschiedene Elemente zu sortieren sind. Das Modell ist verallgemeinerbar auf Probleminstanzen mit mehrfach vorkommenden Elementen.
- Es werden nur die Vergleichsoperationen $<$ und $>$ betrachtet. Alle übrigen Anweisungen, zum Beispiel für das Austauschen von Array-Elementen oder zum Kontrollfluss, werden ignoriert.
- Die Zahlen in Knoten beziehen sich auf die Array-Indexe der initialen Probleminstanz. Zwischenzeitliches Verschieben von Array-Elementen wird nicht betrachtet.
- Jeder direkte Pfad von der Wurzel zu einem Blattknoten im Entscheidungsbaum beschreibt eine Folge von Vergleichen, die ein Algorithmus durchführt, um zu der Entscheidung zu gelangen, dass die durch den Blattknoten beschriebene Permutation das Problem löst.