8. Untere Schranken für Sortieren

- Alle bislang betrachteten Sortieralgorithmen hatten (worst-case) Laufzeit $\Omega(n\log(n))$.
- Werden nun gemeinsame Eigenschaften dieser Algorithmen untersuchen.
- Fassen gemeinsame Eigenschaften in Modell des Vergleichssortierers zusammen.
- Zeigen dann, dass jeder Vergleichssortierer Laufzeit $\Omega(n\log(n))$ besitzt.

Laufzeit von Sortieralgorithmen

		Laufzeit	
		worst-case	average-case
Algorithmus	Insertion-Sort	$\Theta(n^2)$	$\Theta(n^2)$
	Merge-Sort	$\Theta(n\,log(n))$	$\Theta(n\,log(n))$
	Quick-sort	$\Theta(n^2)$	$\Theta(n\,log(n))$
	Heap-sort	$\Theta(n\,log(n))$	-

Vergleichssortierer

Definition 8.1: Ein Vergleichssortierer ist ein Algorithmus, der zu jeder beliebigen Eingabefolge $(a_1, a_2, \ldots, a_n)$ von Zahlen eine Permutation π berechnet, so dass $a_{\pi(1)} \leq a_{\pi(2)} \leq \cdots \leq a_{\pi(n)}$. Dabei benutzt ein Vergleichssortierer außer den durch den Pseudocode definierten Kontrolloperationen nur die Vergleichsoperationen $=, \neq, \leq, \geq, <, >$.

Bemerkungen:

1. Wir nehmen an, dass Eingabezahlen immer paarweise verschieden sind. Benötigen daher = nicht.
2. Können uns auf den Vergleich ≤ einschränken. Andere Vergleichen sind hierzu äquivalent.

Entscheidungsbäume

Definition 8.2: Ein Entscheidungsbaum über n Zahlen ist ein binärer Baum, bei dem

1. Jeder innere Knoten mit $i : j, 1 \leq i, j \leq n$ gelabelt ist.
2. Jedes Blatt mit einer Permutation π auf $\{1,\ldots,n\}$ gelabelt ist.

Entscheidungsbäume und Sortieren

- Mit Entscheidungsbäumen können Vergleichssortierer modelliert werden. Hierzu

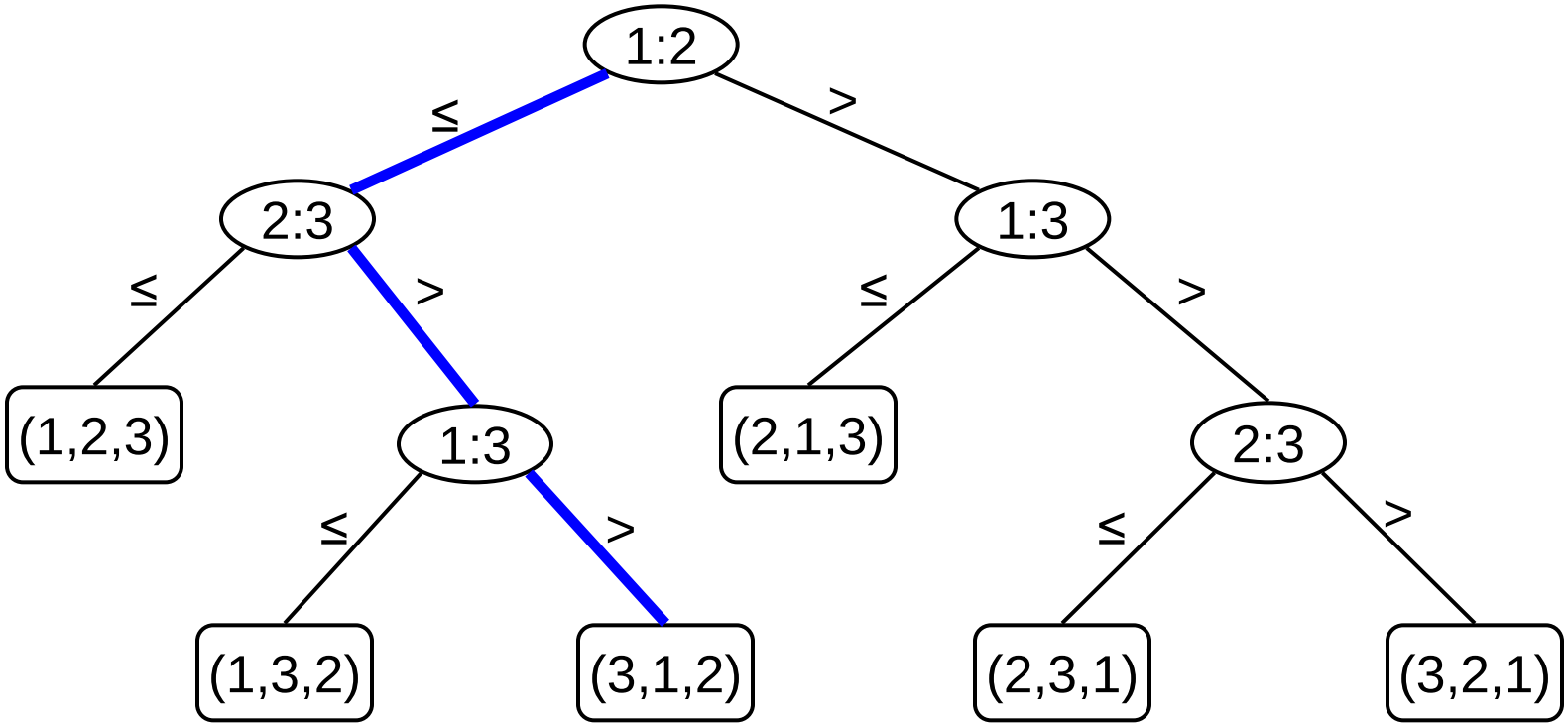
  1. wird bei Eingabe $(a_1,\ldots,a_n)$ ein Pfad von der Wurzel des Baums zu einem Blatt des Baums durchlaufen.
  2. wird an einem inneren Knoten gelabelt mit $i : j$ die Kante zum linken Kind genommen, falls $a_i \leq a_j$, sonst wird die Kante zum rechten Kind genommen.
  3. wird die Permutation π des Blatts am Ende des Pfades ausgegeben.

- Zu einem Vergleichssortierer gibt es für jede Eingabegröße n einen Entscheidungsbaum.

Entscheidungsbaum für Insertion-Sort

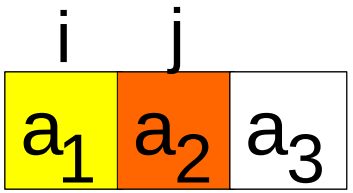
```
InsertionSort(Array A)
1.    for j ← 2 to length(A) do
2.        key ← A[j]
3.        i ← j-1
4.        while i>0 and A[i]>key do
5.            A[i+1] ← A[i]
6.            i ← i-1
7.        A[i+1] ← key
```

Entscheidungsbaum für Insertion-Sort

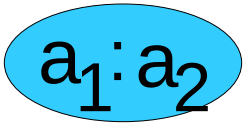


Eingabe : $a_1 = 6, a_2 = 8, a_3 = 5$

Entscheidungsbaum für Insertion-Sort

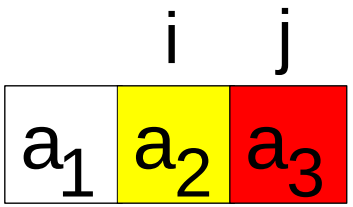


key = a_2

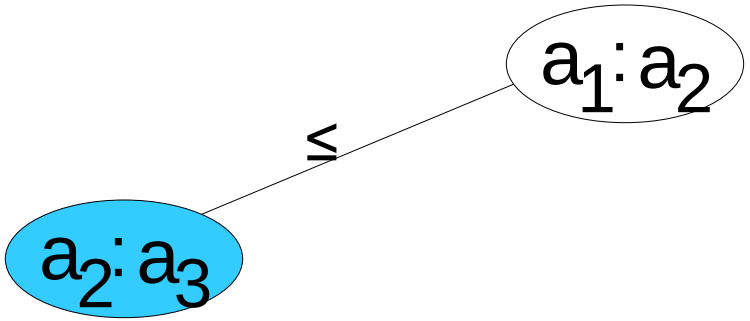



```
InsertionSort(Array A)
for j ← 2 to length(A) do
    key ← A[j]
    i ← j – 1
    while i > 0 and A[i] > key do
        A[i+1] ← A[i]
        i ← i – 1
    A[i+1] ← key
```

Entscheidungsbaum für Insertion-Sort

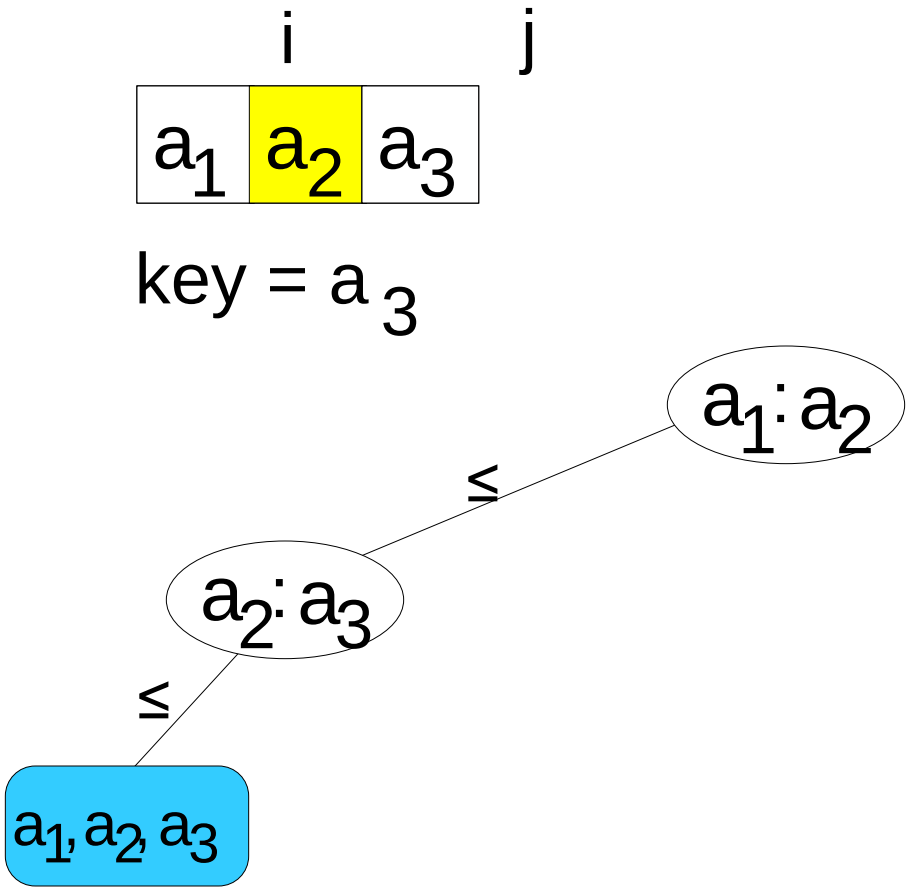


key = a_3



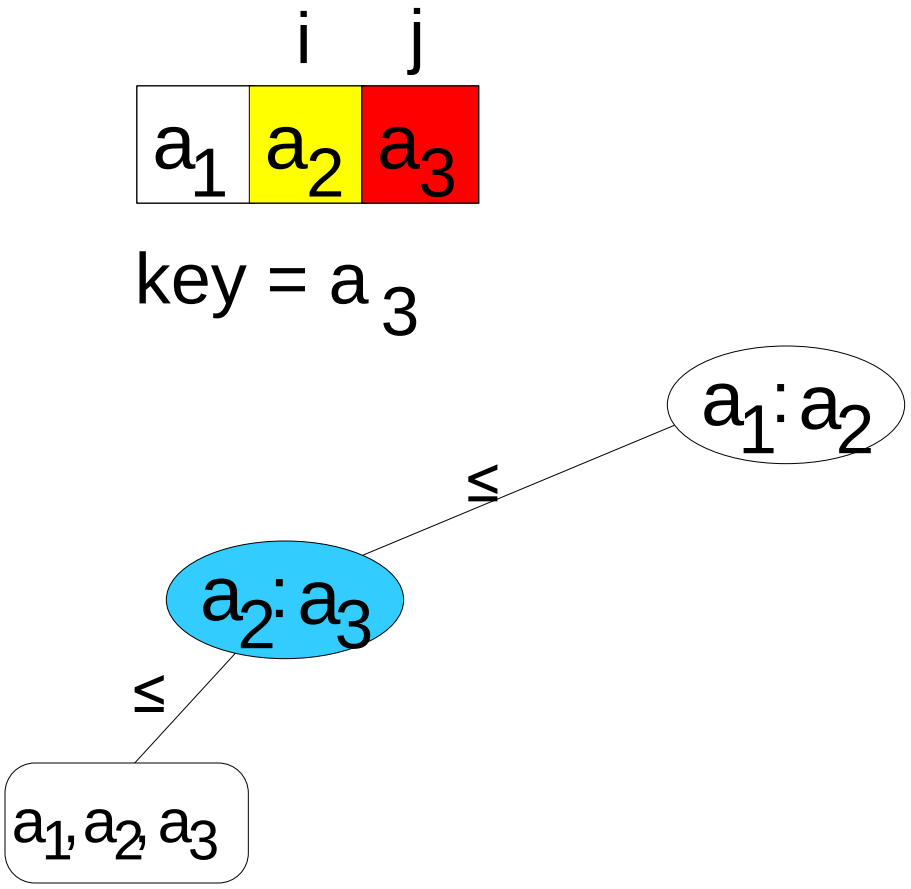

```
InsertionSort(Array A)
for j ← 2 to length(A) do
    key ← A[j]
    i ← j – 1
    while i > 0 and A[i] > key do
        A[i+1] ← A[i]
        i ← i – 1
    A[i+1] ← key
```

Entscheidungsbaum für Insertion-Sort



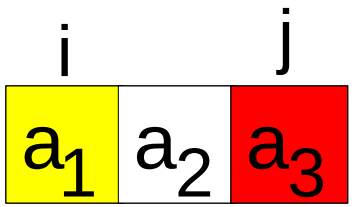

```
InsertionSort(Array A)
for j ← 2 to length(A) do
    key ← A[j]
    i ← j – 1
    while i > 0 and A[i] > key do
        A[i+1] ← A[i]
        i ← i – 1
    A[i+1] ← key
```

Entscheidungsbaum für Insertion-Sort

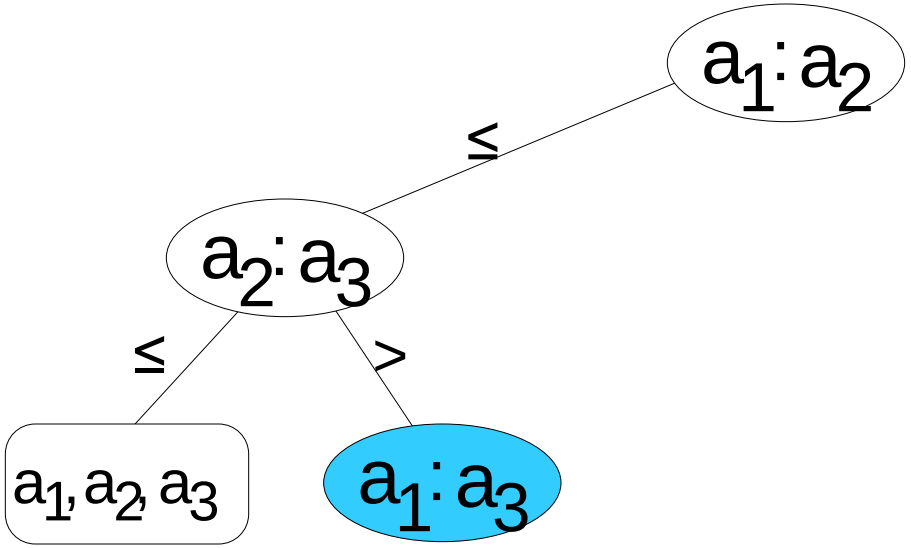



```
InsertionSort(Array A)
for j ← 2 to length(A) do
    key ← A[j]
    i ← j – 1
    while i > 0 and A[i] > key do
        A[i+1] ← A[i]
        i ← i – 1
    A[i+1] ← key
```

Entscheidungsbaum für Insertion-Sort

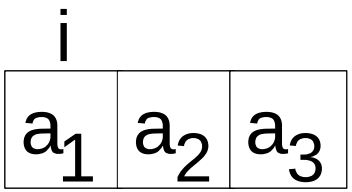


key = a_3



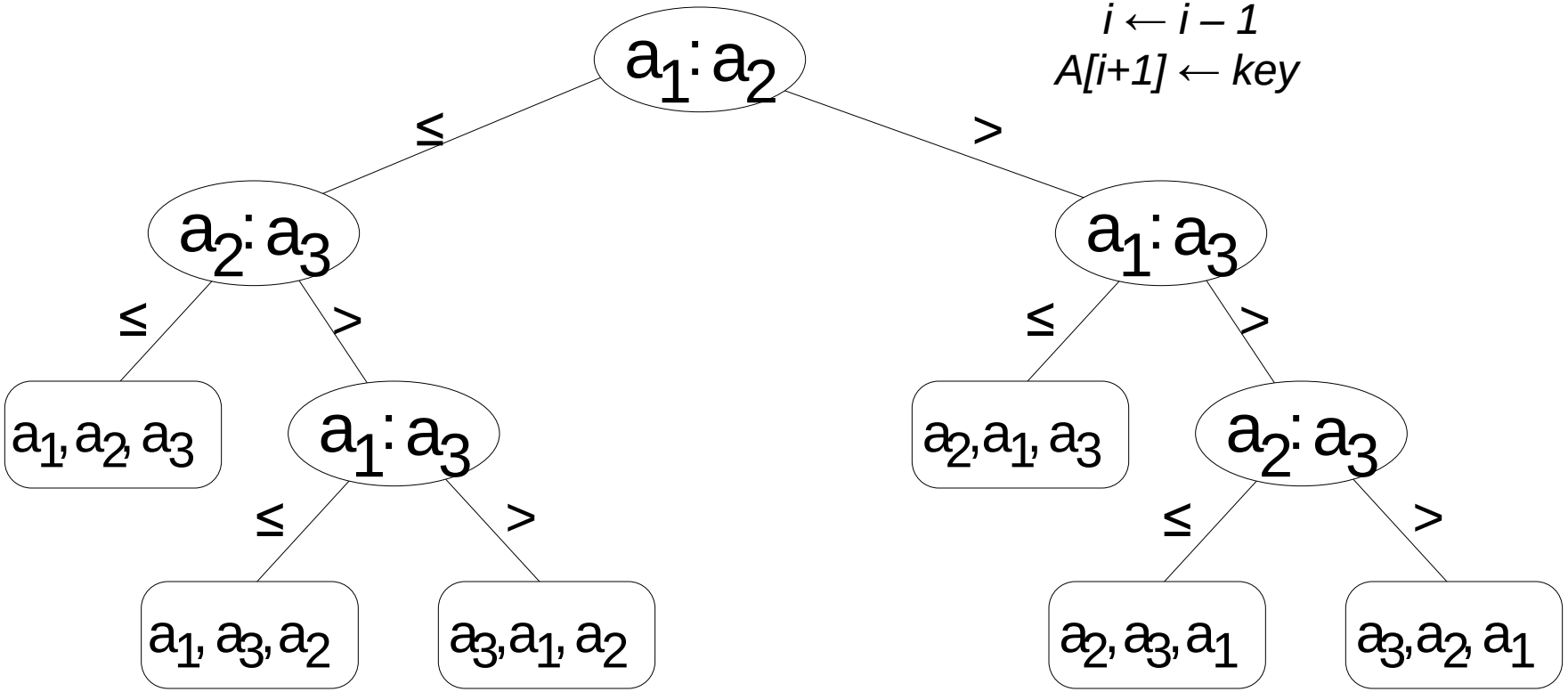

```
InsertionSort(Array A)
for j ← 2 to length(A) do
    key ← A[j]
    i ← j – 1
    while i > 0 and A[i] > key do
        A[i+1] ← A[i]
        i ← i – 1
    A[i+1] ← key
```

Entscheidungsbaum für Insertion-Sort



key = a_3

```
InsertionSort(Array A)
for j ← 2 to length(A) do
    key ← A[j]
    i ← j – 1
    while i > 0 and A[i] > key do
        A[i+1] ← A[i]
        i ← i – 1
    A[i+1] ← key
```



Untere Schranke für Vergleichssortierer

Lemma 8.3: Für Eingaben der Größe n hat ein Entscheidungsbaum für einen Vergleichssortierer mindestens $n!$ Blätter.

Beweis: Wir zeigen, jede Permutation muss in einem der Blätter vorkommen.

- Annahme: eine bestimmte Permutation $(\pi(1), \ldots, \pi(n))$ kommt nicht vor.
- Wir betrachten die Eingabe $(a_1, \ldots, a_n)$ mit $a_{\pi(1)} < \ldots < a_{\pi(n)}$.
- Sei $\pi' \neq \pi$ eine beliebige Permutation. Dann gibt es ein j, so dass $\pi'(j) \neq \pi(j)$ (sei o.B.d.A. $a_{\pi'(j)} > a_{\pi(j)}$).
- In der Ausgabe (nach π') wäre $a_{\pi'(j)}$ nicht an der richtigen Position; somit wäre die Folge nicht sortiert.

Untere Schranke für Vergleichssortierer

Lemma 8.4: $\log(n!) = \Theta(n\log(n))$.

Beweis:

- $n! = 1 \cdot 2 \cdot \ldots \cdot n > (n/2+1) \cdot \ldots \cdot n > (n/2)^{n/2}$
- $n! = 1 \cdot 2 \cdot \ldots \cdot n < n^n$
- $(n/2) \cdot \log(n/2) = \log((n/2)^{n/2}) < \log(n!) < \log(n^n) = n \cdot \log(n)$
- $(n/2) \cdot \log(n/2) = (n/2) \cdot (\log(n)-1) > (n/4) \cdot \log(n)$ für alle $n>4$

Untere Schranke für Vergleichssortierer

Satz 8.5: Die von einem Vergleichssortierer bei Eingabegröße n benötigte Anzahl von Vergleichen ist $\Omega(n\log(n))$.

Beweis: Ein binärer Baum mit N Blättern hat Höhe mindestens $\log(N)$.

- Annahme: Es gibt einen Baum der Höhe $\log(N)-1$ (Anzahl der Ebenen ist $\log(N)$).
- Ebene i hat höchstens 2^i Elemente
- Der Baum hat höchstens $2^{\log(N)-1} < N$ Blätter.

Korollar 8.6: Die von Merge-Sort und Heapsort benötigte Laufzeit von $\Theta(n\log(n))$ ist asymptotisch optimal.