Architektur und Programmierung von Grafik- und Koprozessoren

Sortieren auf der GPU

Stefan Zellmann

Lehrstuhl für Informatik, Universität zu Köln

SS2019

Lernziele

1. **Konstruktion paralleler Sortieralgorithmen** - die Studierenden lernen, wie man mit Hilfe von Sortiernetzwerken parallele Sortieralgorithmen konstruieren kann.
2. **Bitonic Sort** - die Studierenden lernen den Bitonic Sort Algorithmus kennen, der, wenn man ihn parallel formuliert, eine günstige Zeitkomplexität aufweist.
3. **GPU Implementierung** - die Studierenden verstehen, wie man Bitonic Sort auf der GPU implementiert.

Parallele Sortieralgorithmen

Sortierproblem

Gegeben: *(Eingabesequenz)*
Menge $A = \{a_0, a_1, a_{n-1}\}$ mit Ordnung $\leq$.

Gesucht: *(Ausgabesequenz)*
Permutation $A' = \{a'_0, a'_1, a'_{n-1}\}$ sodass $\forall a'_i, a'_j, i < j$ gilt: $a'_i \leq a'_j$.

Vergleichsbasierte Sortieralgorithmen

- Iteriere in festgelegter Reihenfolge über die Eingabesequenz und vergleiche Elemente paarweise. Vertausche Elemente a_i und a_j basierend auf der Ordnungsrelation.
- Diverse vergleichsbasierte Sortieralgorithmen aus Grundstudium bekannt
  - Insertion Sort, Selection Sort, Merge Sort, Quick Sort etc.

Insertion Sort

Informell:
Teilfolge A_l ist bereits sortiert. Iteriere über alle $a' = a_{l+1}, .., a_{n-1}$ und sortiere diese in die Teilfolge A_l ein.

Die einelementige Teilfolge A_0 erfüllt (trivial) die Eigenschaft, bereits sortiert zu sein. Starte daher Iteration mit $a' = a_{l+1} = a_1$.

Vergleiche (von rechts nach links) alle a' mit allen Elementen aus A_l. Sobald Element $a'' \in A_l$ gefunden, sodass $a'' > a'$, bilde neue Teilfolge $A'_l = \{a_0, .., a', a'', .., a_{l+1}\}$ durch Einfügen von a' an der richtigen Stelle und Verschieben der Elemente $\{a'', .., a_l\}$ nach rechts.

Insertion Sort

Beispiel

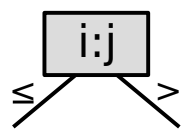
6	***4***	2	8	7	1	$A_I = \{6\}$
4	6	***2***	8	7	1	$A_I = \{4, 6\}$
2	4	6	***8***	7	1	$A_I = \{2, 4, 6\}$
2	4	6	8	***7***	1	$A_I = \{2, 4, 6, 8\}$
2	4	6	7	8	***1***	$A_I = \{2, 4, 6, 7, 8\}$
1	2	4	6	7	8	$A_I = \{1, 2, 4, 6, 7, 8\}$

Entscheidungsbäume

Mit jedem vergleichsbasierten Sortieralgorithmus ist für jede Länge n der Eingabesequenz ein *Entscheidungsbaum* assoziiert, aus dem sich die *Sequenz der Vergleichsoperationen* für jede beliebige Eingabesequenz A ablesen lässt.

Blätter für Permutationen A'. ($\Rightarrow$ Entscheidungsbaum für $|A| = n$ hat $n!$ Blätter.)

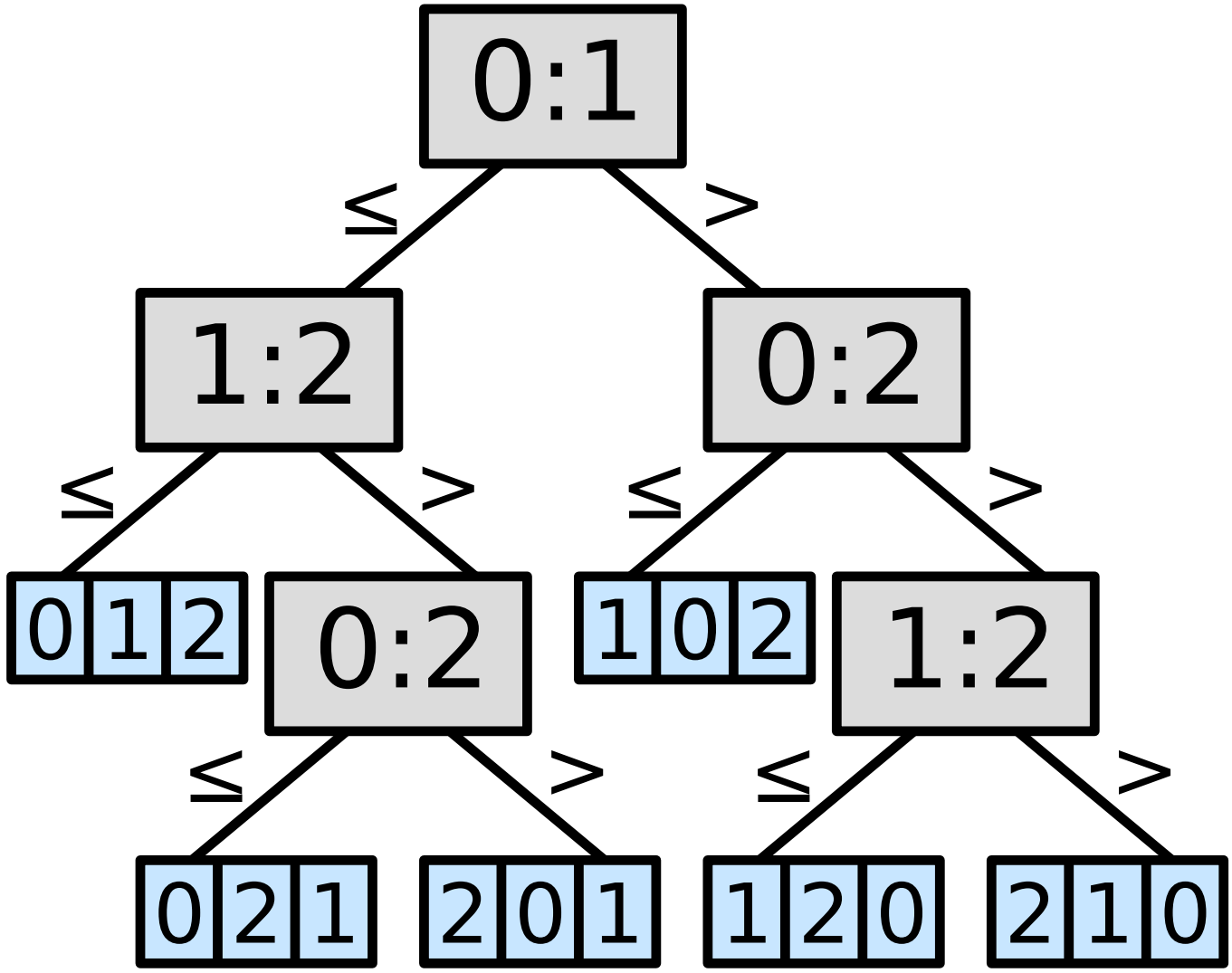
Innere Knoten des Entscheidungsbaums:



vergleicht die Eingabelemente a_i und a_j. Beim Traversieren des Baums entscheiden wir je nach Ergebnis, ob wir nach rechts oder links traversieren.

Entscheidungsbäume

Beispiel: Insertion Sort, $n = 3$

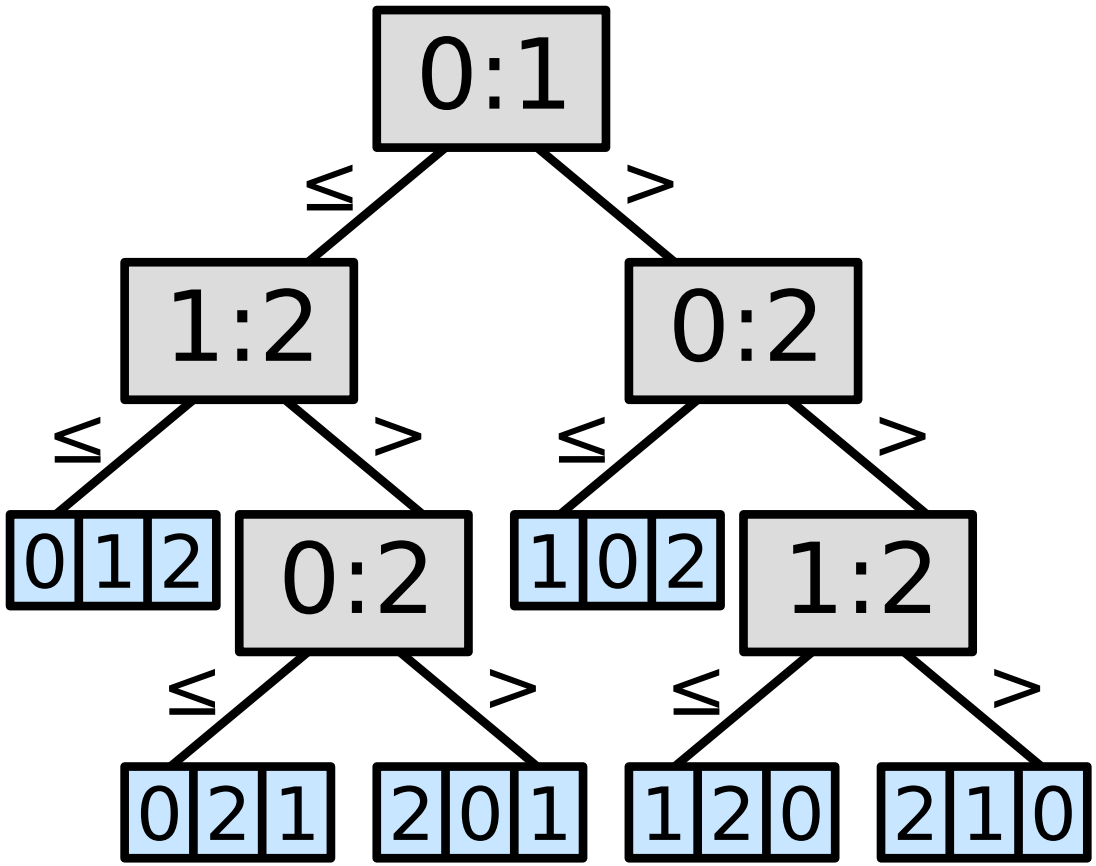


Entscheidungsbäume

Beispiel: Insertion Sort, $A = \{8, 4, 2\}$

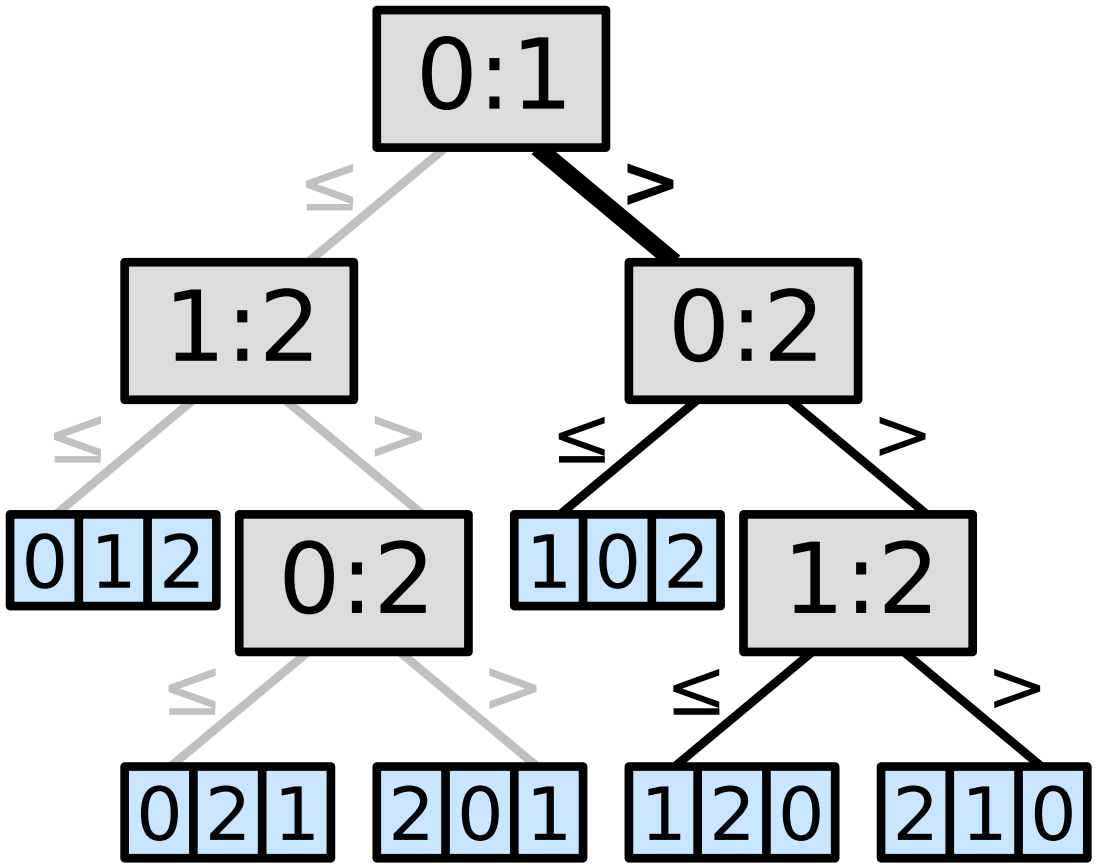
$A = \{8, 4, 2\}$

$A_I = \{a_0\}$



Entscheidungsbäume

Beispiel: Insertion Sort, $A = \{8, 4, 2\}$

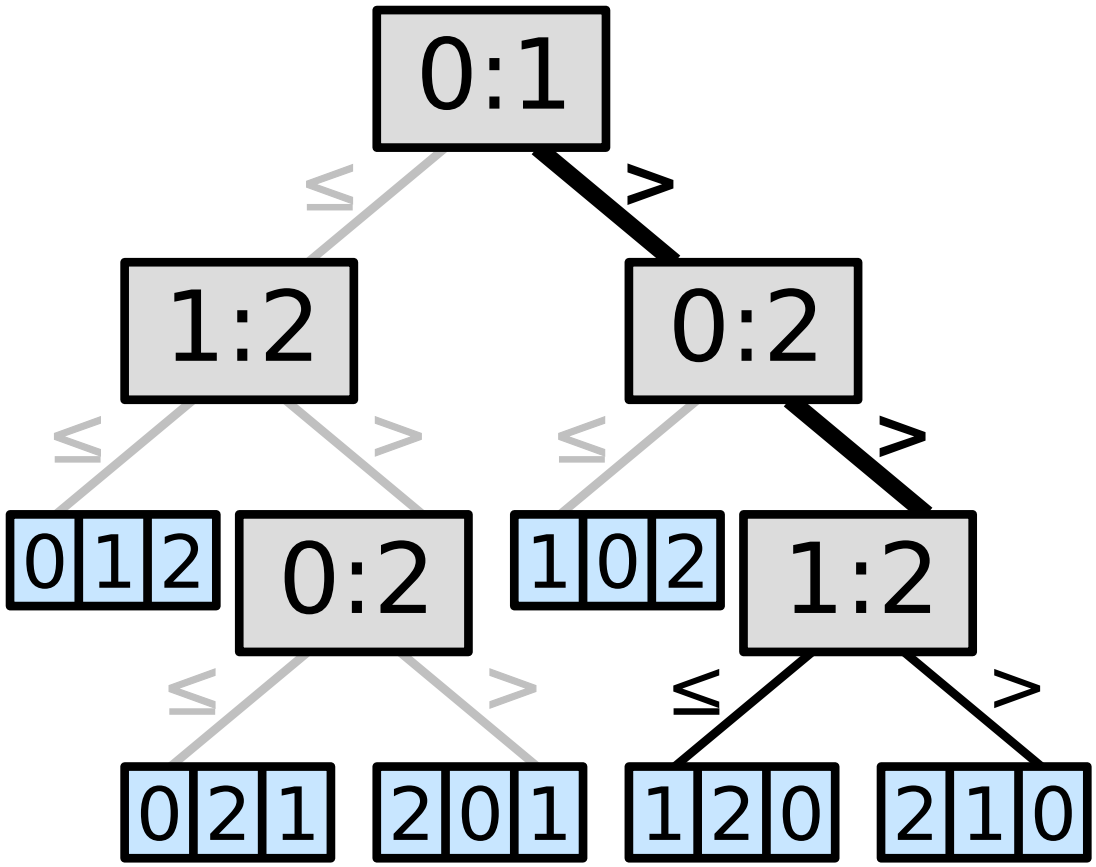


$A = \{8, 4, 2\}$

$a_0 = 8 > a_1 = 4$
$A_l = \{a_1, a_0\}$

Entscheidungsbäume

Beispiel: Insertion Sort, $A = \{8, 4, 2\}$



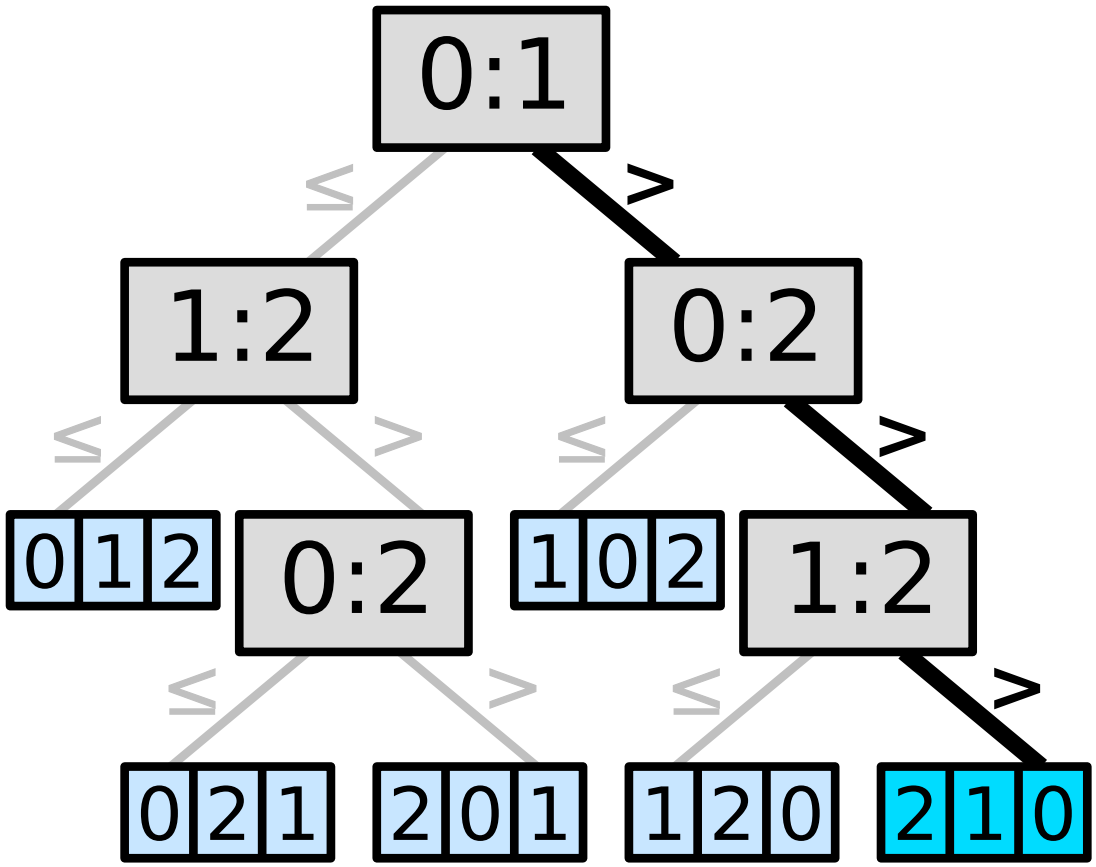
$A = \{8, 4, 2\}$

$a_0 = 8 > a_1 = 4$
$A_l = \{a_1, a_0\}$

$a_0 = 8 > a_2 = 2$

Entscheidungsbäume

Beispiel: Insertion Sort, $A = \{8, 4, 2\}$



$A = \{8, 4, 2\}$

$a_0 = 8 > a_1 = 4$
$A_l = \{a_1, a_0\}$

$a_0 = 8 > a_2 = 2$
$a_1 = 4 > a_2 = 2$

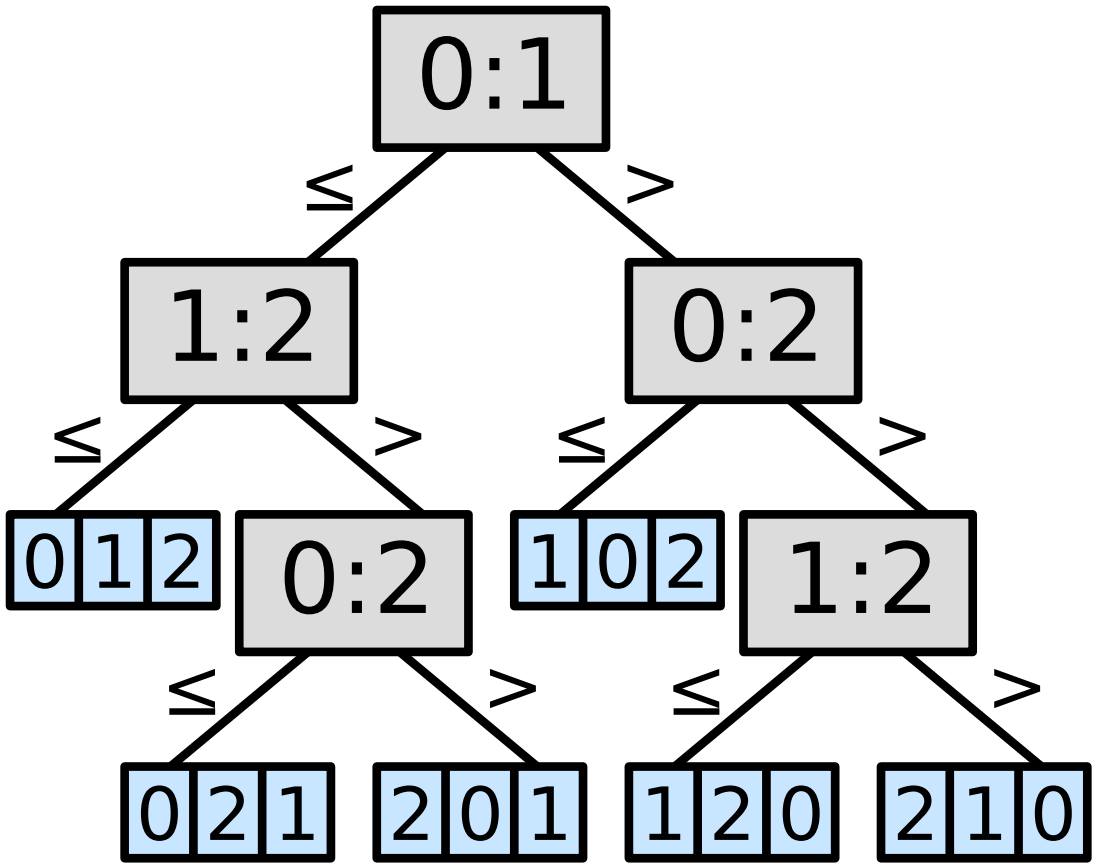
$A' = \{a_2, a_1, a_0\}$

Entscheidungsbäume

Beispiel: Insertion Sort, $A = \{2, 4, 3\}$

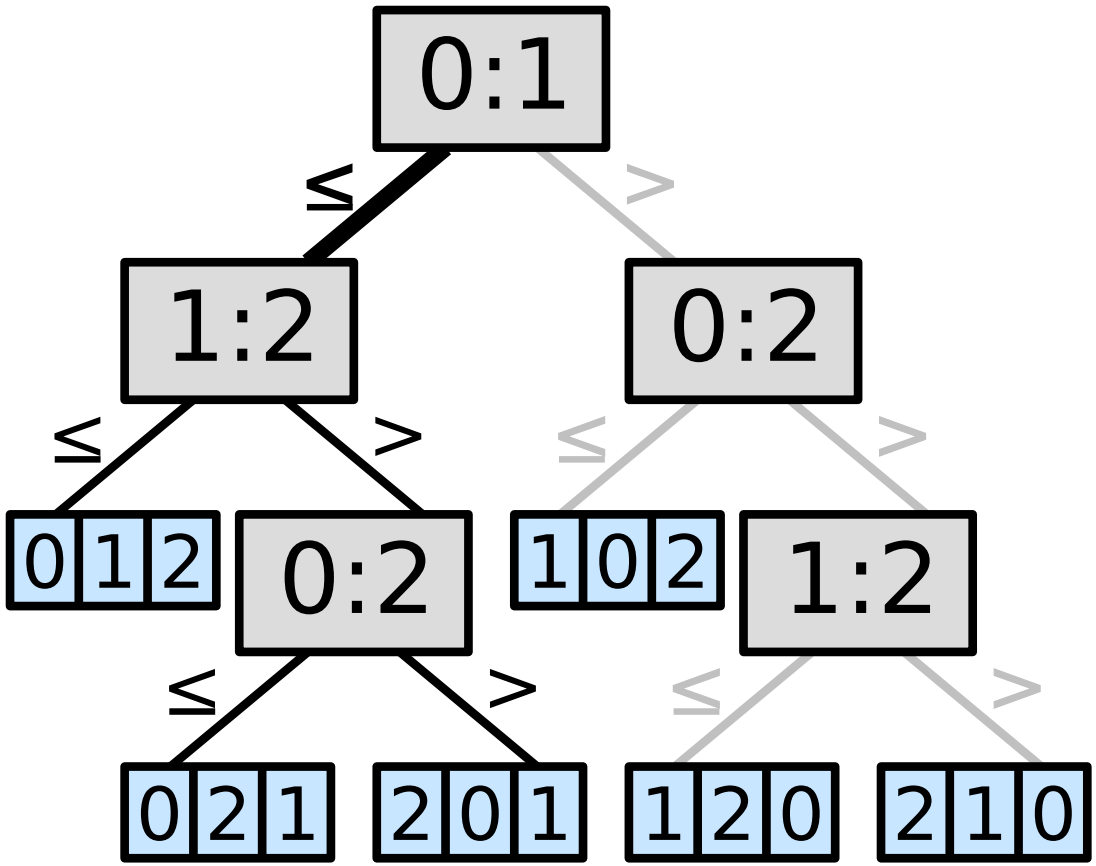
$A = \{2, 4, 3\}$

$A_I = \{a_0\}$



Entscheidungsbäume

Beispiel: Insertion Sort, $A = \{2, 4, 3\}$

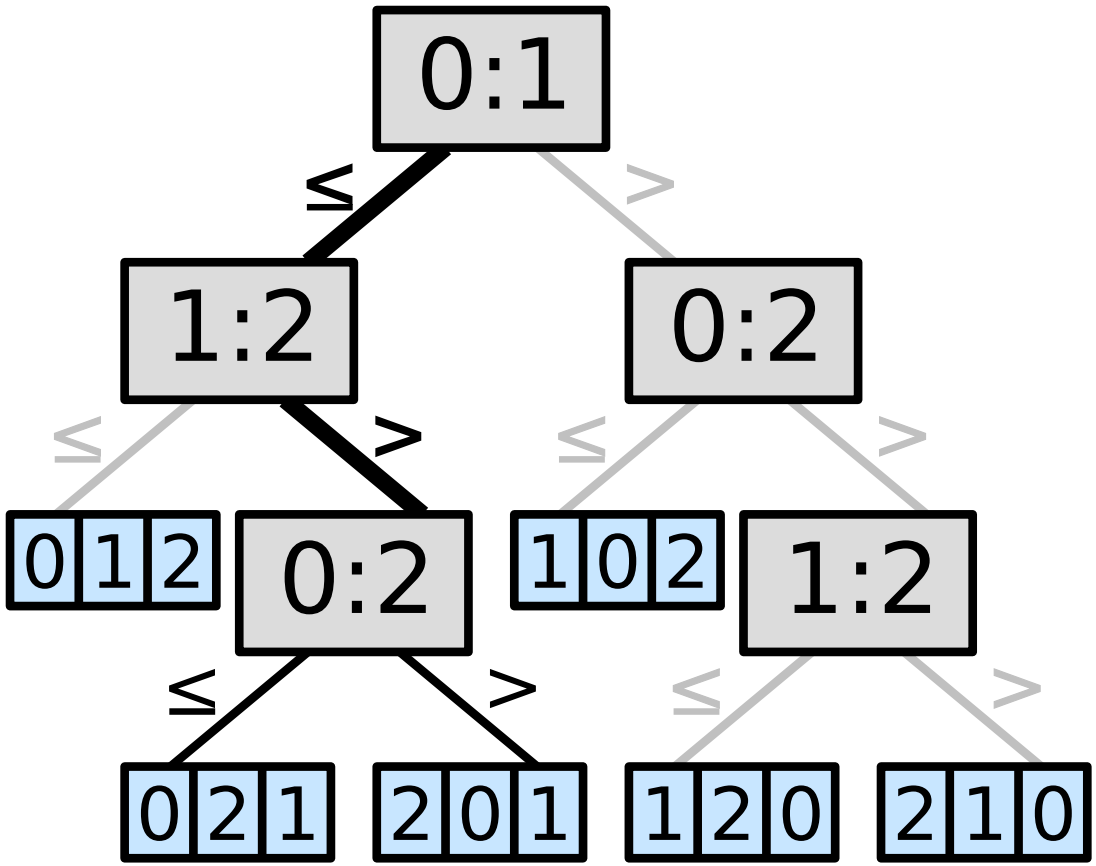


$A = \{2, 4, 3\}$

$a_0 = 2 < a_1 = 4$
$A_l = \{a_0, a_1\}$

Entscheidungsbäume

Beispiel: Insertion Sort, $A = \{2, 4, 3\}$



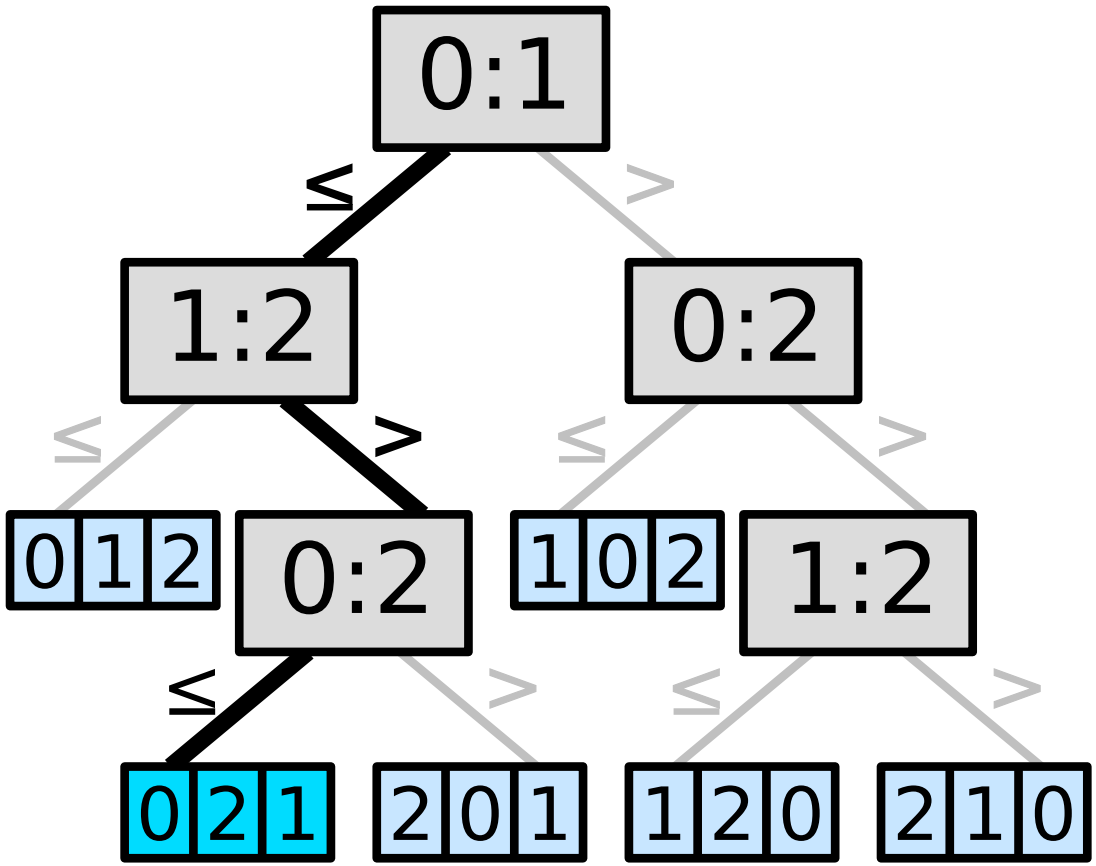
$A = \{2, 4, 3\}$

$a_0 = 2 < a_1 = 4$
$A_l = \{a_0, a_1\}$

$a_1 = 4 > a_2 = 3$

Entscheidungsbäume

Beispiel: Insertion Sort, $A = \{2, 4, 3\}$



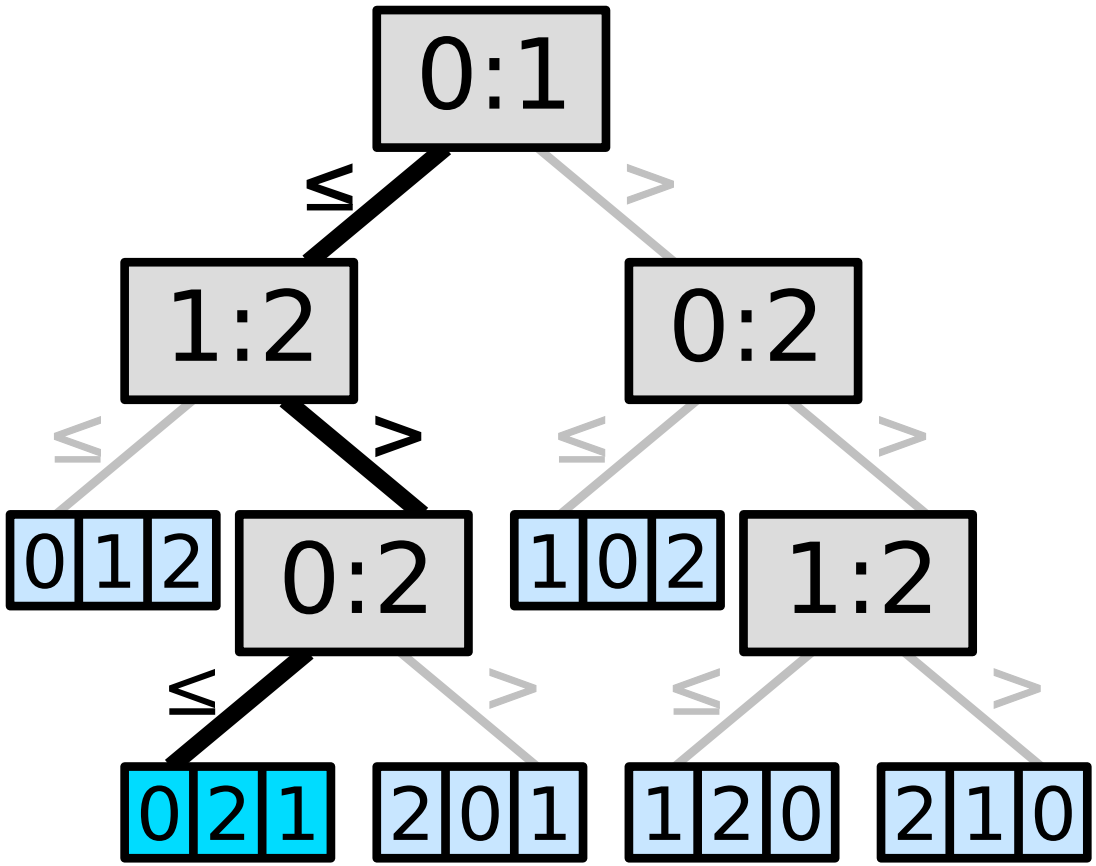
$A = \{2, 4, 3\}$

$a_0 = 2 < a_1 = 4$
$A_l = \{a_0, a_1\}$

$a_0 = 2 < a_2 = 4$
$A' = \{a_0, a_2, a_1\}$

Entscheidungsbäume

Beispiel: Insertion Sort, $A = \{2, 4, 3\}$



$A = \{2, 4, 3\}$

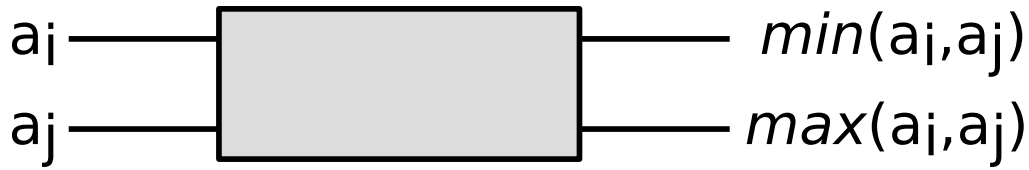
$a_0 = 2 < a_1 = 4$
$A_l = \{a_0, a_1\}$

$a_0 = 2 < a_2 = 3$
$A_l = \{a_0, a_1\}$

$a_1 = 4 > a_2 = 3$
$A' = \{a_0, a_2, a_1\}$

Sortiernetzwerke

Sortiernetzwerke setzen sich aus *Vergleichsmodulen* zusammen:

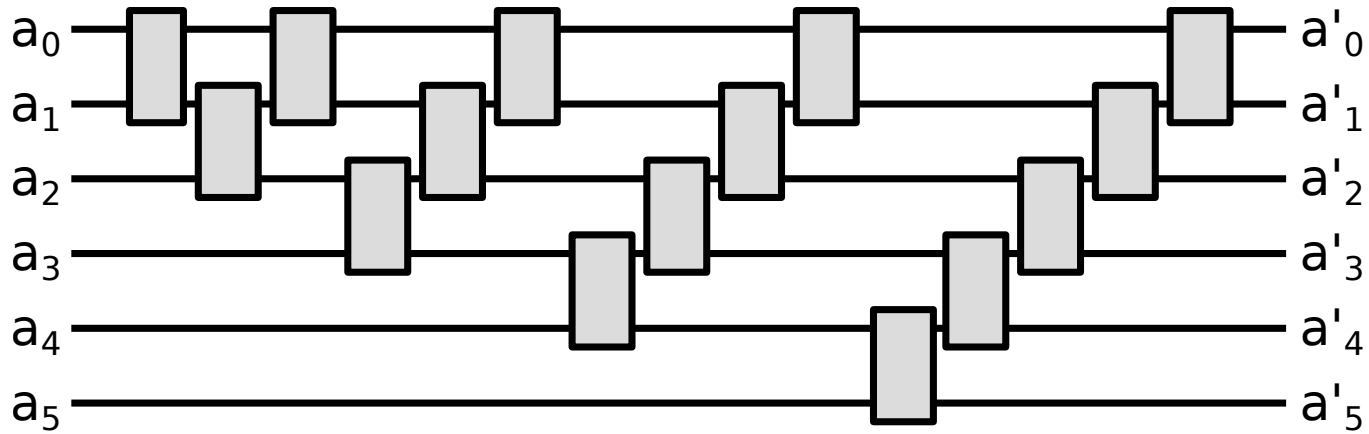


Diese haben zwei Eingabeports (links). Das Vergleichsmodul vergleicht die beiden Eingaben a_i, a_j und vertauscht sie basierend auf der Ordnung. Dies ist eine $O(1)$ Operation. Das Ergebnis liegt auf den Ausgabeports (rechts) an.

Vergleichsmodule sind über *Leitungen* miteinander verbunden.

Sortiernetzwerke

Sortiernetzwerk angelehnt an Insertion Sort, das Eingabefolge $A = \{a_0, ..a_5\}$ in sortierte Ausgabefolge $A' = \{a'_0, ..a'_5\}$ überführt.



Sortiernetzwerke

Dieses Netzwerk kann man auch etwas anders darstellen, dann sieht man, dass manche Vergleichsmodule parallel ausgeführt werden können.