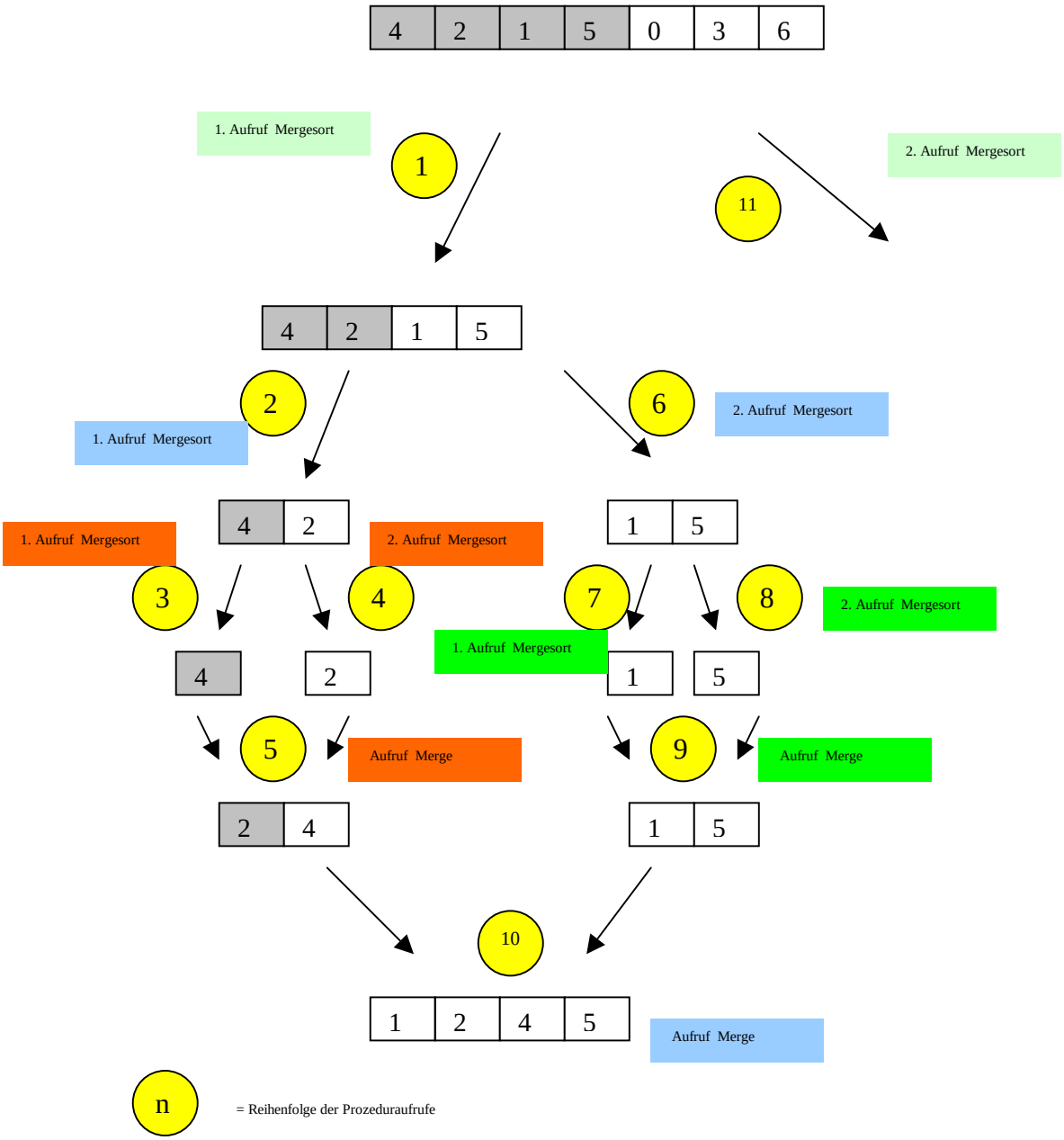
MergeSort (Sortieren durch Mischen)

Mergesort verfährt nach dem Prinzip des Divide and Conquer (Teile und Herrsche Prinzip) Die ursprüngliche Version von Mergesort teilt eine Folge F rekursiv in gleich große Teilfolgen auf, um dann die Teilfolgen zu sortieren und dann wieder zusammen zu mischen. Hierfür benötigt Mergesort zusätzlichen externen Speicherplatz, abhängig von der Eingabelänge N. Daher handelt es sich bei Mergesort um ein externes Sortierverfahren.

Die Methode von MergeSort an einem Beispiel.
parallele Sichtweise :

5	4	1	7	3	2	8	6
[5	4	1	7]	[3	2	8	6]
[5	4]	[1	7]	[3	2]	[8	6]
[5]	[4]	[1]	[7]	[3]	[2]	[8]	[6]
[4	5]	[1	7]	[2	3]	[6	8]
[1	4	5	7]	[2	3	6	8]
1	2	3	4	5	6	7	8

Rekursive Sichtweise:



Wie wird gemischt?

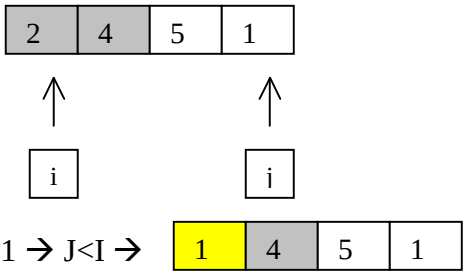
Beispiel: 2 4 1 5

Zum Mischen benötigt man ein Hilfsband. Auf dieses Hilfsband werden nun die Teilfolgen übertragen, so dass die erste Teilfolge links, die zweite Teilfolge von rechts in absteigender Reihenfolge übertragen werden:

2 4 5 1

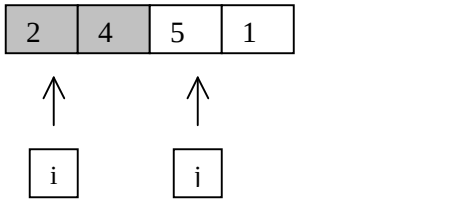
Nun benötigt man nur noch zwei Positionszeiger, die jeweils auf die Anfangselemente der Teilfolgen zeigen. Dann muss man nur noch die jeweiligen Elemente auf die die Zeiger zeigen vergleichen und dann das kleinere übertragen:

1.

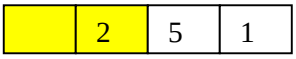


I=2, J=1 → J<I →

2.



I=2, J=5 → J>I →



…

Analyse :

Für das Verschmelzen (merge) sind im ungünstigsten Fall höchstens n Vergleiche und 2n Datenbewegungen (Transport in den Zwischenspeicher und zurück) erforderlich, also höchstens 3n Dominante Operationen.

Jeder Aufruf von mergesort führt im Rekursionsfall zu zwei weiteren Aufrufen von mergesort mit halber Problemgröße.

Da also bei jedem Aufruf von Mergesort die Reihung halbiert wird, finden insgesamt log(n) +1 Aufrufe statt. Es verdoppeln sich auf jeder Stufe die Anzahl derAbschnitte, dafür werden die Abschnitte aber auch nur halb so groß → Auf jeder Stufe sind nur halb so viele dominante Operationen nötig. → Die Anzahl der Operationen bleibt damit konstant.

→ Mergesort ist in O(3n * (log(n)+1)) → O(n log(n))

ABER Mergesort benötigt doppelt so viel Speicherplatz wie ein In-Place Algorithmus!!!!

FAZIT

- Worstcase O(n log n)
- Es baut sich im Gegensatz zu Quicksort (immer) ein Baum mit logarithmischer Tiefe auf
- Averagecase O(n log n)
- Aber zusätzlicher Speicher von Nöten → externes Sortieren
- Eignet sich gut für das Sortieren von Daten auf Sekundärspeichern
- Eigentliche Variante ist rekursiv.
- Es existieren auch iterative Varianten. → gleiche Laufzeiten.
- Iterativ ist Effizienter, da rekursive Aufrufe auch Speicher belegen