VL09, Lösung 1

a) Insertion Sort ist ein stabiles Sortierverfahren. Dies liegt an der Prüfung auf „größer“ im Kopf der `while`-Schleife (`while j >= 1 and A[j] > x do`). Die Schleife wird beendet, wenn man den sortierten Zielteil komplett überprüft hat (`j == 0`), oder wenn man vorher ein Element `A[j]` gefunden hat, das kleiner oder gleich dem einzusortierenden Element `x` ist. (`A[j] <= x`). `x` wird dann rechts von diesem Element eingefügt. Hat auch `A[j]` den Wert `x`, bleibt nach dem Einsortieren die ursprüngliche relative Reihenfolge dieser beiden gleichen Schlüssel erhalten. Beispiel:

2	**4**[1)]	5	4[2)]	3
2	4[1)]	**5**	4[2)]	3
2	4[1)]	5	**4**[2)]	3
2	4[1)]	4[2)]	5	**3**
2	3	4[1)]	4[2)]	5

Die ursprüngliche Reihenfolge der beiden Elemente mit dem Wert 4 wurde beim Sortieren beibehalten. Um die beiden Elemente mit dem Wert 4 auseinanderhalten zu können, wurden diese mit ihrer relativen Position 1) und 2) versehen. Bei einer Prüfung auf „größer gleich“ geht die Stabilität verloren.

b) Vergleiche zwischen Elementen des Arrays treten nur in der Bedingung der `while`-Schleife auf. Dabei kann im günstigsten Fall nur ein Vergleich stattfinden, so dass die Schleife erst gar nicht betreten wird. Dies ist dann der Fall, wenn das Array bereits richtig sortiert ist, so dass der Schlüssel `x` des einzusortierenden Elements größer gleich dem der bereits im Zielteil befindlichen Elemente ist. Die Gesamtzahl der Vergleiche ergibt sich dann aus der Anzahl der Durchläufe der Zählschleife, wobei für jeden Durchlauf ein Vergleich stattfindet. Da die Schleife $n-1$ Mal durchlaufen wird, ergibt sich $n-1$ als Minimalanzahl der Schlüsselvergleiche.

c) Im ungünstigsten Fall ist das einzusortierende Element kleiner als alle bereits im Zielteil befindlichen Elemente. Es wird dann mit allen bereits im Zielteil befindlichen Elementen verglichen, also im k-ten Durchlauf (`i == k+1`) mit allen Elementen der Indizes i-1 (`== k`) bis 1. Das sind genau k Vergleiche im k-ten Sortierschritt. Da es $n-1$ Durchläufe/Sortierschritte gibt (die erste Zahl ist bereits korrekt einsortiert), ergibt sich als Gesamtzahl die Summe aller Zahlen von 1 bis $n-1$. Dies ist bekanntlich $½n*(n-1)$ (Gaußsche Summe). Dieser Fall tritt ein, wenn das Array zu Anfang in umgekehrter Reihenfolge sortiert war.

Sortierschritt `k`	**Position `i` des Elementes**	**Anzahl der Vergleiche**
1	2	1
2	3	2
3	4	3
...	...	...
n-1	n	n-1
		Summe: $\sum_{k=1}^{n-1} k = \frac{n(n-1)}{2}$

VL09, Lösung 2

a) Selection Sort ist nicht stabil. Bei Sortierung der Zahlenfolge 2, 4, 5, 4, 3 ändert sich die relative Reihenfolge der beiden Elemente mit dem Wert 4:

2	4[1)]	5	4[2)]	3
2	4[1)]	5	4[2)]	**3**
2	3	5	**4[2)]**	4[1)]
2	3	4[2)]	5	**4[1)]**
2	3	4[2)]	4[1)]	5

Einen stabilen Sortieralgorithmus erhält man, wenn das Vertauschen der Elemente `A[i]` und `A[min]` durch ein Verschieben (Aufrücken) der entsprechenden Elemente nach rechts ersetzt wird.

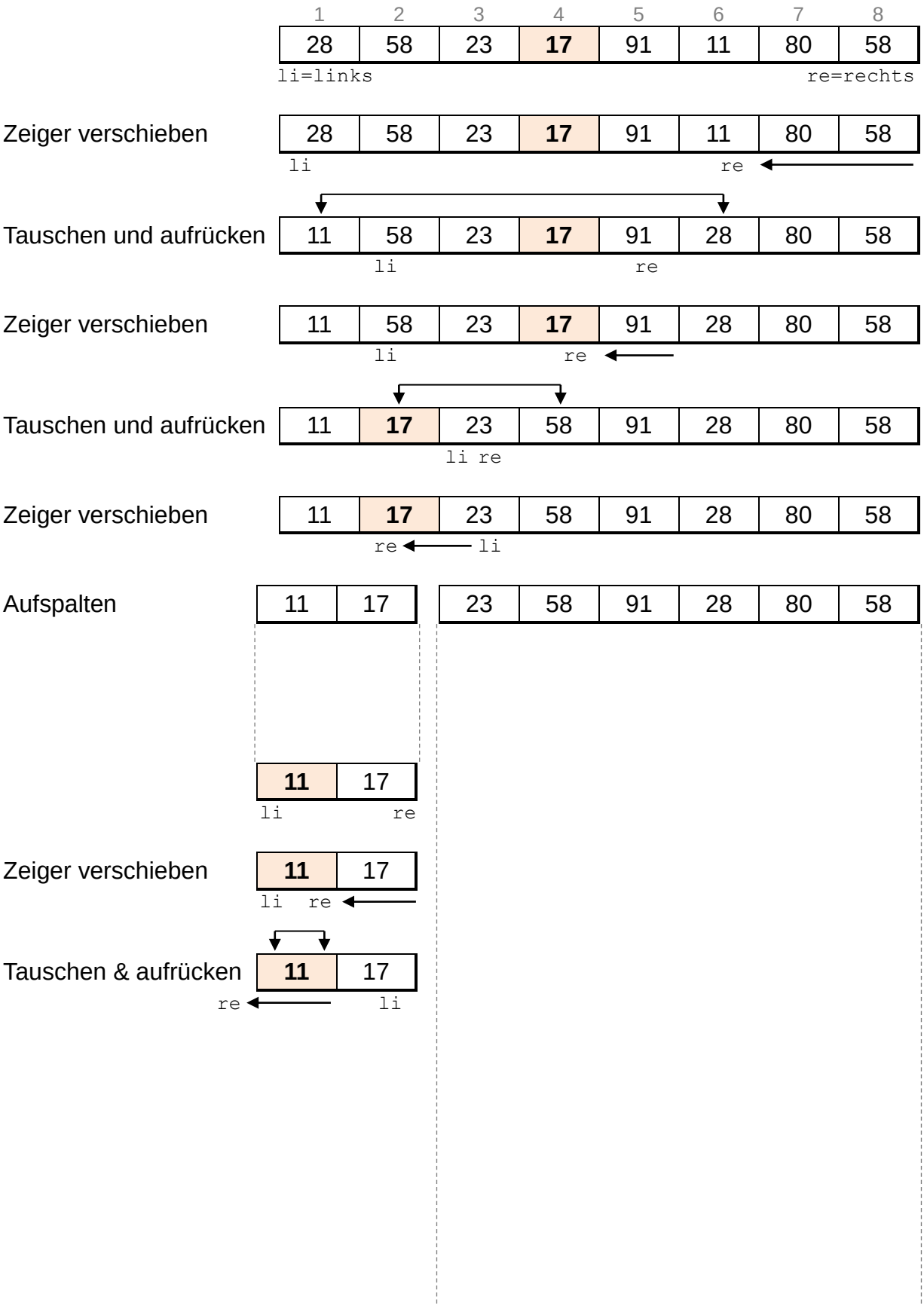
b) Damit die Zahlen in absteigender Reihenfolge sortiert werden, muss im sortierten Teil jeweils das Maximum gesucht werden. Die Prüfung auf „kleiner“ muss durch eine Prüfung auf „größer“ ersetzt werden (`A[j] > A[max]`).

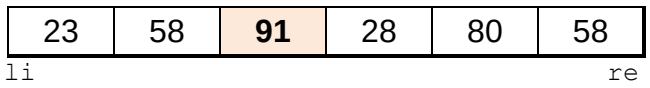
c) In der äußeren `for`-Schleife läuft `i` von 1 bis n-1. `i` ist der linke Rand des noch nicht sortierten Quellteils. In diesem wird nach dem Minimum gesucht, was durch die innere `for`-Schleife realisiert wird. Die innere `for`-Schleife wird n-i Mal durchlaufen. Pro Durchlauf findet ein Vergleich statt. Insgesamt werden bei der Abarbeitung der inneren `for`-Schleife also n-i Vergleiche durchgeführt.

Beginn des Quellbereichs an Position `i`	**Anzahl der Vergleiche**
1	n-1
2	n-2
3	n-3
...	...
n-1	1
	Summe: $\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$

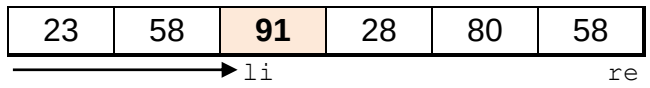
Die Anzahl der Vergleiche insgesamt ist also die Summe aller Zahlen von 1 bis n-1, was bekanntlich ½n∗(n-1) ist.

VL09, Lösung 3

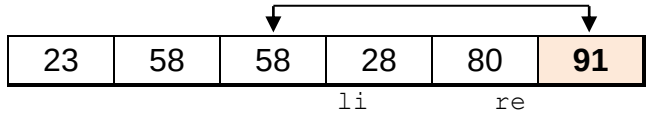




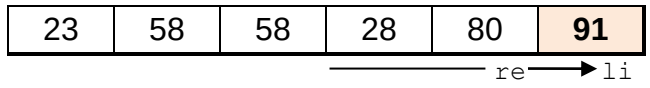
Zeiger verschieben



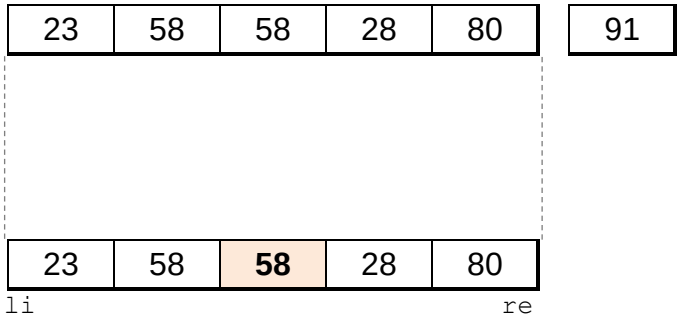
Tauschen und Zeiger aufrücken



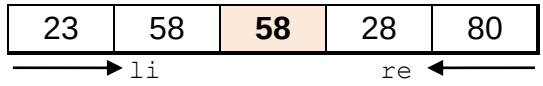
Zeiger verschieben



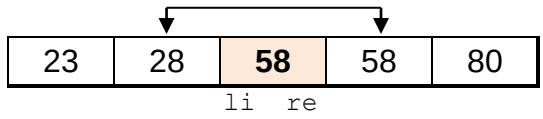
Aufspalten



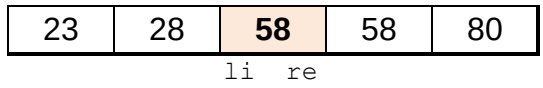
Zeiger verschieben



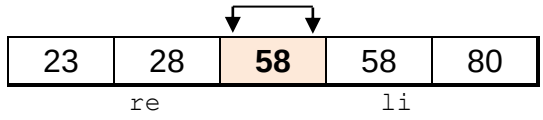
Tauschen und Zeiger aufrücken



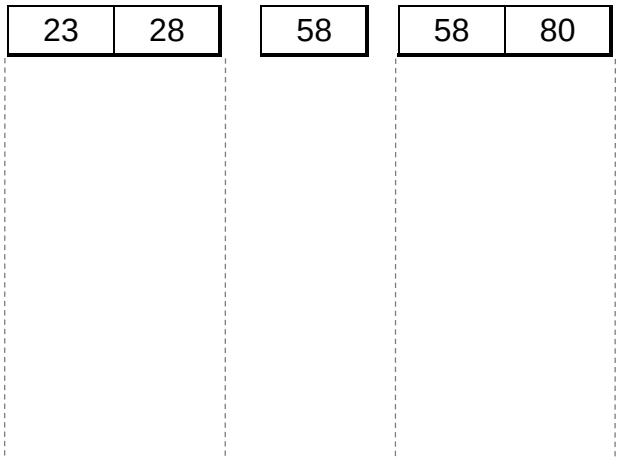
Zeiger verschieben

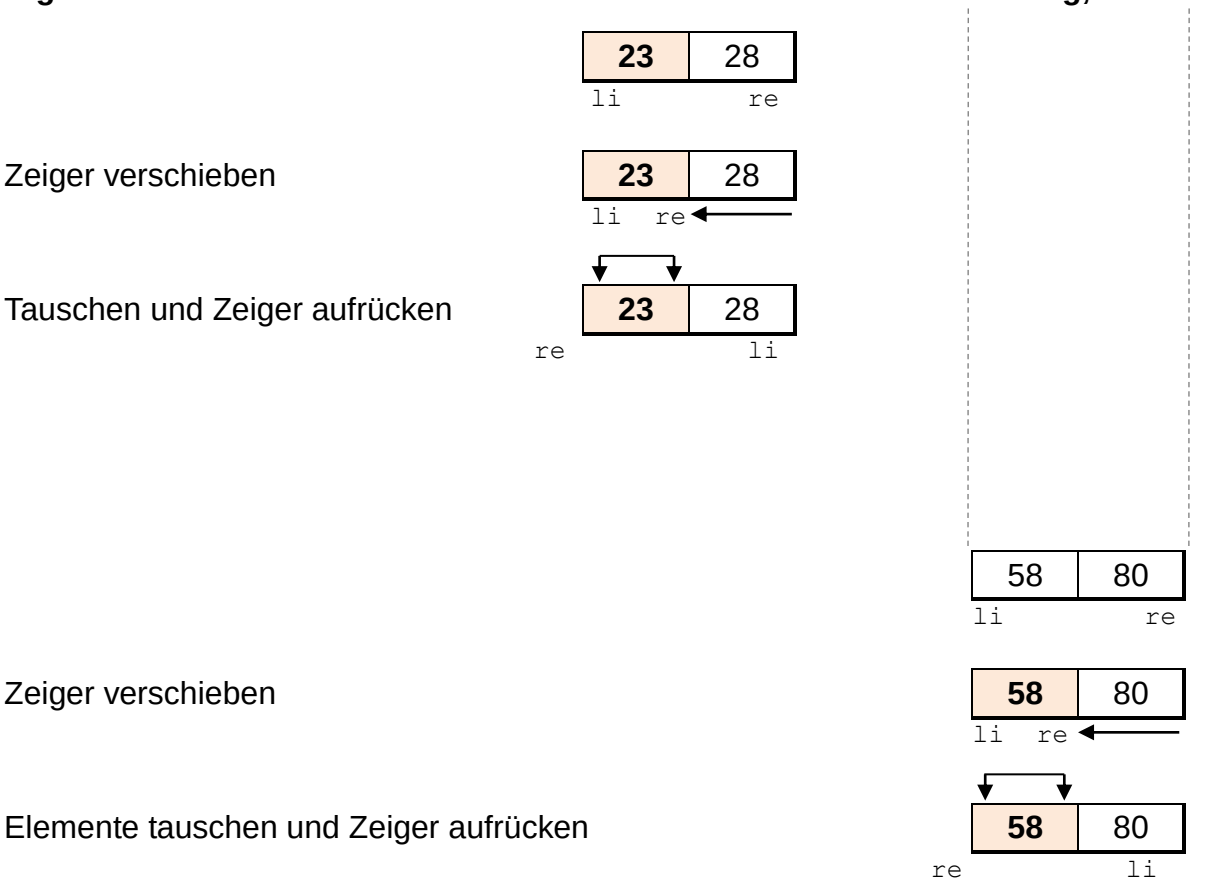


Tauschen und Zeiger aufrücken



Aufspalten





VL09, Lösung 4 und 5

Die Lösungen zu diesen Aufgaben werden gemeinsam mit den Lösungen zu VL10 veröffentlicht.