Quicksort - Algorithmus

Thomas Maul

6. Dezember 2025

1 Quicksort

Es wird oft gewünscht, dass Daten sortiert vorliegen. Dabei ist es in der Regel gleichwertig, ob sie aufsteigend alphabetisch, absteigend oder nach anderen Kriterien sortiert werden.

Sortieralgorithmen sind daher ein häufiges Thema in der Informatik. Bubblesort ist einfach zu erlernen, aber relativ langsam. Bei großen Datenmengen wird Bubblesort eher nicht verwendet.

Quicksort wird häufiger verwendet. Der Algorithmus ist relativ schnell in der Ausführung. Quicksort kann in verschiedenen Varianten implementiert werden. Ich verwende die sogenannte in-Place-Variante, bei der zwei Zeiger die Elemente absuchen und direkt vertauschen, falls nötig.

Im Beispiel möchte ich das folgende Array mit Zahlen sortieren lassen.

1.1 Quicksort, Version in-place

Im ersten Schritt wird ein Pivotelement festgelegt. Das Element wird später benötigt, um andere Einträge mit dem Pivotelement zu vergleichen. Im ersten Schritt ist es gleichgültig, ob ich das erste oder letzte Element wähle. Ich wähle das letzte Element des Arrays (siehe Bild 2).

Jetzt werden zwei Hilfszeiger benötigt (klein und gross). Gross (g) beginnt am linken Ende des Arrays, klein (k) am rechten neben dem Pivotelement (siehe Bild 3). In jedem Arbeitsschritt prüfe ich, ob die Stelle bei klein kleiner ist als das Pivotelement und gross größer oder gleich groß gegenüber dem Pivotelement ist. Falls klein kleiner oder gleich ist im Vergleich zum Pivotelement lasse ich es an der Position stehen und setze den Suchzeiger (k) um eine Position weiter nach rechts. Falls ich ein Element finde, das

5	3	9	4	1	8	6	10	2	7

Abbildung 1: Array, Ausgangssituation

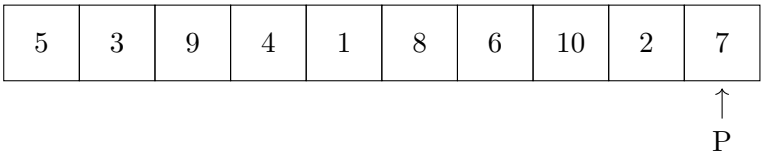

Abbildung 2: Array, Ausgangssituation

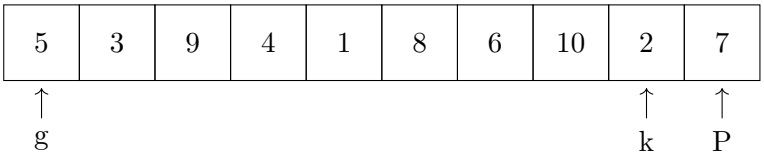

Abbildung 3: Array, Zeiger auf kleineres und größeres Element (relativ zu Pivotelement)

größer als der Pivotelement ist, lasse ich den Suchzeiger k dort stehen und suche mit gross (g) nach einem Element, das kleiner oder gleich gegenüber dem Pivotelement ist.

Im Beispiel sucht der Zeiger ‚g' zuerst von links kommend das erste Element, das größer als 7 ist. Anschließend sucht k das nächste Element, das kleiner oder gleich 7 ist (Bild 4). Hier entsprechen die Werte 5 und 2 an den ersten Positionen von g und k den Kriterien der Suche.

Im folgenden Schritt werden die Einträge, auf die g und k zeigen, vertauscht. Anschließend sucht der Algorithmus wieder, solange, bis die Hilfszeiger wieder passende Elemente gefunden haben. Falls g und k bei der Suche aneinander vorbeilaufen, bricht der Such- und Vertauschungs-Algorithmus ab. In Bild 5 stehen 8 und 6 zum Tauscht an.

Zum Abschluss des Arbeitsschritts wird geprüft, ob die Inhalte an den Positionen P und g getauscht werden müssen. Wenn das Element an der Position g größer ist, als das an P (Bild 6), werden die Inhalte ausgetauscht. Damit ist die erste Runde des Algorithmus beendet.

1.1.1 Runde 2

In der nächsten runde werden die Abschnitte links und rechts von P_1 zu eigenen Teilen, die einzeln betrachtet werden. Im linken Abschnitt wird 6 zum neuen Pivotelement, 8 ist im rechten Abschnitt das neue Pivotelement (Bild 8).

In Runde zwei der Sortierung betrachte ich zuerst den linken Teil und suche hier

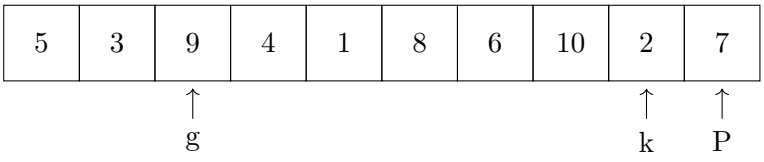

Abbildung 4: Array, Zeiger auf Elemente zum Tausch

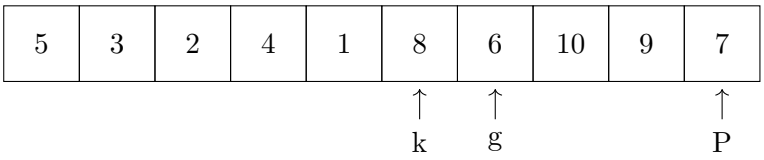

Abbildung 5: Zweites Paar zum Tauschen

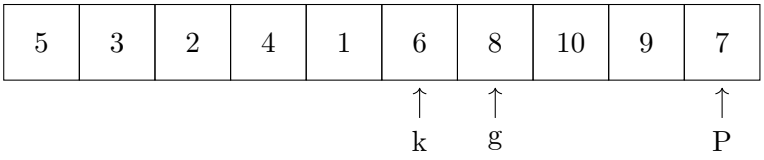

Abbildung 6: Array, Hilfszeiger haben die Position gewechselt

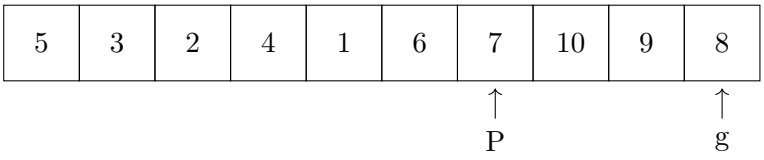

Abbildung 7: Tausch ‚gross‘ mit Pivotelement

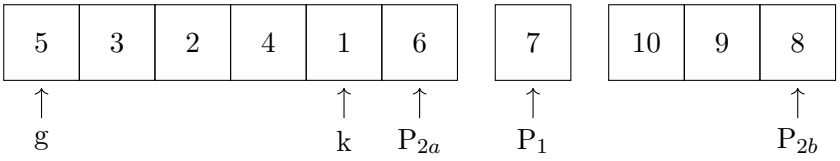

Abbildung 8: Aufteilung, P_1 ist alleine, links und rechts neue Bereiche

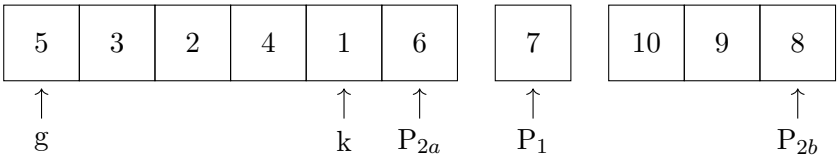

Abbildung 9: Suche links ist abgeschlossen ($5 < 7$)

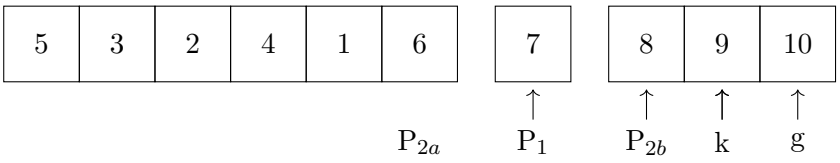

Abbildung 10: Tausch rechts (10 und 8)

wieder die Elemente, die kleiner und größer als das Pivotelement sind (Bild 8). Da kein Element aus der linken Teilmenge größer als 7 ist, wird hier nicht getauscht.

Im rechten Teil werden die Elemente 10 und 9 nicht getauscht, da 9 größer als 8 ist. Die zweite Prüfung ($10 > 8$) fällt positiv aus. Daher werden 10 (k) und P_{2b} getauscht.

Somit sind 6 und 10 sortiert.

1.1.2 Rund 3

Da keine Zahl kleiner als 1 ist (Bild 11), werden hier keine Werte von g und k vertauscht. Lediglich 1 und 5 werden vertauscht. In der rechten Teilmenge ist nur noch 9 als Element vorhanden, hier ist der Vergleich $9 > 10$ negativ, daher bleiben die Elemente an der Position stehen. Somit ist auch Runde 3 angeschlossen.

1.1.3 Runde 4

In Runde 4 wird kein Element vertauscht (Bild 12).

1.1.4 Runde 5

In Runde 5 (Bild 13) werden wieder die Elemente 2 und 3 vertauscht. Bei der Prüfung in Runde 6 (1 und 2) gibt es keine weiteren Vertauschungen, damit ist der Sortieralgorithmus abgeschlossen.

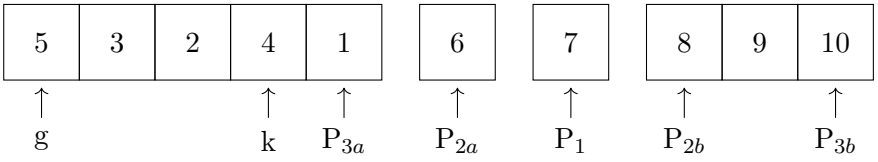

Abbildung 11: Runde 3, suche links (g, k, P_{3a})

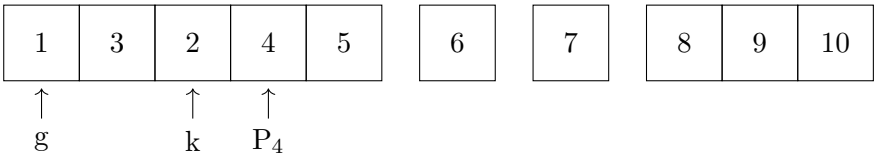

Abbildung 12: Runde 4, suche links (g, k, P_4)

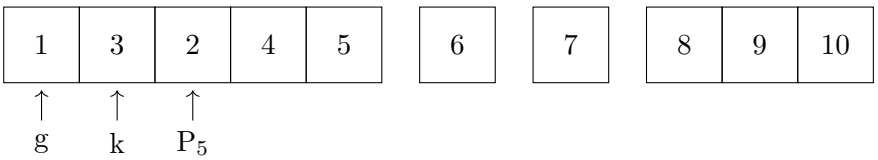

Abbildung 13: Runde 5, suche links (g, k, P_5)

1.2 Quicksort, zweite Implementierung

Bei der zweiten Variante wird ebenfalls ein Pivotelement festgelegt und dann alle andere Elemente danach ausgerichtet. Elemente, die kleiner sind kommen auf die linke Seite, Elemente, die größer sind auf die rechte Seite. Anschließend wird der Bereich am Pivotelement getrennt und jeweils der linke und der rechte Bereich als neue Gesamtbereiche betrachtet.