Programmierung in der gymnasialen Oberstufe
für Lehramtsstudierende im Fach Informatik

**Aktivitäten zu Sortierverfahren im Informatikunterricht**

Autor:innen: Maximilian Wadenpohl, Paula Bobach

In diesem Dokument werden verschiedene analoge Aktivitäten zur Visualisierung der Arbeitsweise von Sortieralgorithmen aufgeführt. Diese Aktivitäten können im Seminar mit den Studierenden gemeinsam erarbeitet werden und als Diskussionsgrundlage dienen.

**Aktivität 1 – Bubble Sort Algorithmus:**

Benötigte Materialien:

- 6 Zettel mit den Zahlen 0-4 (eine Zahl doppelt)

Wir orientieren uns an der einfachsten möglichen BubbleSort Programmierung (hier in Java):

void BubbleSort(int[] F) {

 **for** (int n= F.length; n >1; n=n-1) {

 **for** (int i =0; i < F.length-1; i++) {

 **if** (F[i] > F[i+1]){

 swap(F, i, i+1);

 }

 }

 }

}

7 Schüler werden gebeten nach vorne zu kommen. 6 davon sollen sich in einer beliebigen Reihenfolge in einer Reihe aufstellen. Anschließend zieht jeder der 6 Schüler verdeckt einen der vorbereiteten Zettel und hält ihn gut sichtbar vor sich. Sie stellen das Array dar, dem verschiedene integer Werte in Form von Zetteln übergeben werden. Zusätzlich wird n=0 an der Tafel festgehalten. Der siebte Schüler simuliert das Durchlaufen des Algorithmus. Was er dabei genau zu tun hat kann gemeinsam mit der Klasse erörtert oder durch die Lehrperson vorgegeben werden. Grundsätzlich solle der Ablauf folgendermaßen aussehen: Der siebte Schüler tritt von links nach rechts gehend jeweils an die in einer Reihe stehenden Schüler heran, nimmt zwei der Zettel, vergleicht sie, vertauscht sie gegebenenfalls und gibt sie an die Schüler zurück. Wenn er die Reihe einmal durchlaufen hat wird das n an der Tafel um eins erhöht und ein weiterer Durchlauf wird gestartet. Dies wird solange gemacht bis an der Tafel n=5 geschrieben wird (also nach insgesamt 5 Durchgängen). Nun bricht die äußere For-Schleife ab und das Array sollte spätestens nach dem letzten Durchlauf vollständig sortiert sein.

Aufbauend darauf bieten sich mehrere Schwerpunkte und Vertiefungsmöglichkeiten an:

* Was passiert, wenn das Array bereits sortiert ist? Oder alle Zahlen darin identisch sind?
* Wovon hängt die Laufzeit ab? Welche Laufzeit hat BubbleSort also?
* Wie könnte man das Programm optimieren?

Aktivität 2 – MergeSort:

Als Vorbereitung sollte mit den Schülern die arbeitsweise der Funktion merge (siehe Anhang im Dokument „Sortierverfahren“) besprochen worden sein.

Materialien:

* 6 Zettel mit jeweils einer Zahl darauf (eine Zahl doppelt, der Rest verschieden zueinander)
* Kreide (3 Farben)

Wir orientieren uns an folgender MergeSort Programmierung (in Python):

# Sortiert ein Teilstueck des Arrays a

# beginnend beim Index lo bis zum Index hi (ausschliesslich)

def mergesort(a, lo, hi):

 if hi-lo>1:

 m=(lo+hi)//2 # m = Index in der Mitte

 mergesort(a, lo, m) # vordere Haelfte sortieren

 mergesort(a, m, hi) # hintere Haelfte sortieren

 merge(a, lo, m , hi) # Haelften verschmelzen

Für eine übersichtliche Durchführung der Aktivität wird viel Platz benötigt. Es empfiehlt sich daher mit den Schülern gemeinsam den Schulhof aufzusuchen. Fünf Schüler sollen sich in einer beliebigen Reihenfolge in einer Reihe aufstellen. Anschließend wird mit Kreide ein Kreis um die Schüler gemalt. Der Kreis stellt den ersten Aufruf mit MergeSort da. Anschließend werden die Zettel an die Schüler verteilt.

Nun wird die Reihe der Schüler geteilt. Die zwei am weitesten links treten aus dem Kreis heraus nach vorne (sie behalten ihre Reihenfolge bei). Um sie wird ein weiterer Kreis auf den Boden gezeichnet. Zusätzlich wird ein Pfeil vom ersten, größeren Kreis zum zweiten nun neu entstandenen Kreis gezeichnet. So wird der erste rekursive Aufruf von MergeSort simuliert. Parallel dazu machen die restlichen drei Schüler auf der rechten Seite das Gleiche. Dieser Vorgang wird wiederholt bis jeder Schüler in seinem eigenen Kreis steht (s. Abb. 1). Anschließend wird rückläufig zu den Pfeilen gemerged. Die Schüler vergleichen Ihre Zahlen, der Schüler mit der kleinsten Zahl tritt zuerst in den nächstgrößeren Kreis, im folgt der Schüler mit der nächsthöheren Zahl. Jeder Wechsel von Kreis zu Kreis wird mit einem entsprechenden Kreidepfeil versehen. Am Ende stehen die Schüler nach ihren Zetteln geordnet im großen Kreis.

Abbildung 1 – MergeSort: Gelb ist der Startzustand, Blau der zustand nach den rekursiven MergeSort Aufrufen

Als Abwandlung der Aktivität oder im Anschluss daran kann beim durchlaufen des Algorithmus an den entsprechenden Pfeilen ein zeitlicher Ablauf in einer weiteren Kreidefarbe notiert werden (s. Abb.2).

Abbildung 2 - mögliche Kreidezeichnung am Ende der Aktivität

Aufbauend darauf bieten sich mehrere Schwerpunkte und Vertiefungs-möglichkeiten an:

* Was passiert, wenn das Array bereits sortiert ist? Oder alle Zahlen darin identisch sind?
* Wovon hängt die Laufzeit ab? Welche Laufzeit hat MergeSort? (Im Vergleich zu BubbleSort?)
* Wie könnte man das Programm optimieren?