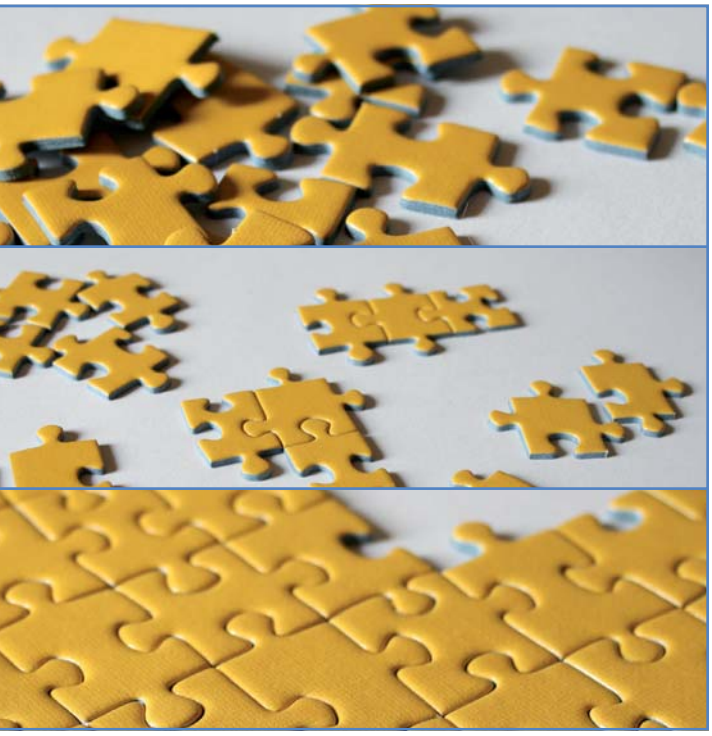
Lehreinheit E-So1 *Schuften oder Denken?*



Nach welchem System geht man beim Puzzle-bauen vor?

Zeitrahmen
110 Minuten

Zielgruppe
- Sekundarstufe I;
- Sekundarstufe II mit geringen Informatik-Vorkenntnissen

Lehrziel
- Effizienz von Algorithmen
- Zusammenwirken von Datenstrukturen und Algorithmen
- Informatik ::= zuerst denken, dann erst implementieren

Motivation
In vielen Lebenssituationen müssen wir Dinge sortieren. Oft findet dies unter erschwerenden Rahmenbedingungen statt. Wir sortieren Rechnungen oder Belege einmal nach Betrag und kurz darauf nach Datum und haben dazu meist zu wenig Tischfläche vor uns, um alle Objekte mit gebührendem Abstand nebeneinander zu legen. Vor dem Bau eines Dachstuhls sollten die Balken so aufgelegt werden, dass sie passend nach Länge und Verwendungszeitpunkt liegen und beim Bau einer Steinstützmauer sollten die Steine auch so liegen, dass man die nach Fertigstellung einer Reihe noch verfügbaren Steine wieder in einer Anordnung hat, aus der leicht ersichtlich ist, welche Steine als nächste in die Mauer passen.
Die Bewegung großer Blöcke ist aufwändig. Handelt es sich um Gesteinsblöcke ist dies offensichtlich, handelt es sich um Datenblöcke, ist das nicht so offensichtlich. Aber es ist dennoch der Fall und im Computer finden sehr oft Umordnungen von Datenbeständen statt, um effizienter auf einzelne Elemente zugreifen zu können. *(Siehe auch Module und Einheiten zu Suche)*.
Man wird sich in beiden Fällen überlegen, wie man die Zahl der kraftraubenden (Zeit raubenden) Operationen minimiert. Bei den Gesteinsblöcken wird dies die Bewegung der Felsteile sein. Wenn man sie zur Bestimmung der richtigen Anordnung allerdings zuerst vermessen oder abwiegen muss, damit man sie zueinander in Beziehung setzen kann, ist dies auch mit einem nicht zu vernachlässigenden Aufwand verbunden.
Freilich bedeutet dies, dass man die Messergebnisse irgendwie auf die Steine schreiben wird. – Wir nähern uns daher von einem physischen Problem, bei dem die Steine einfach so lange bewegt werden, bis sie in der richtigen Reihenfolge liegen, einem informatischen Problem.
- Die Steine werden vermessen und das Ergebnis angeschrieben.
- Nun wird eine Ordnung zwischen den Vermessungsergebnissen hergestellt und diese Ordnung auf die Position der aktuellen Lage der Steine zurück abgebildet.
- Schließlich werden die Steine so angeordnet, wie es der vorher erstellten Reihenfolge der Messergebnisse entspricht.

Bleibt man bei Steinen, ist klar, dass das Problem einfacher ist, wenn man die Steine vorerst abseits jener Stellen gelagert hat, an denen sie letztlich platziert werden sollen. Dies gilt auch für das Sortieren von Daten. Doch während Steine in der Regel abseits des Bauplatzes abgekippt werden, sind Daten meist in der Struktur, in der sie auch sortiert werden (in situ Sortierverfahren). Dieser Unterschied ist dadurch begründet, dass man bei zu sortierenden Steinblöcken meist wenig Objekte sortieren muss (10, 100, oder vielleicht auch 1.000) während bei Daten meist sehr viele Datensätze (10.000, 100.000, mehrere Millionen) zu sortieren sind.
Die besondere Rolle von Sortieralgorithmen in der Informatik besteht darin, dass man innerhalb vieler Verfahren erst einmal mit Sortierproblemen beschäftigt ist, da es sich mit sortierten Datenbeständen viel einfacher und effizienter arbeiten lässt als mit unsortierten. *(Siehe auch Module und Einheiten zu Suche)*.

Requisiten
Geburtstagsordnung: keine

Partizipanden
Siehe Beschreibung der Partizipanden bei den einzelnen Algorithmen.

Grobstruktur
1. Motivation
2. Aufgabenstellung, ein Sortierverfahren zu entwickeln
   a) für große Gesteinsblöcke
   b) für „normale“ Datensätze
3. Durchspielen eines elementaren Sortierverfahrens
   siehe So1– Bubble-Sort, So2 – Selection-Sort, So3 – Insertion-Sort

Aufgrund des Gesteinsblocks-Paradigmas ist es wahrscheinlich, dass die Schüler für die Gesteinsblöcke von sich aus auf ein Verfahren kommen, das im Wesentlichen dem Selection-Sort entspricht. Man kann hier aber flexibel jenes Verfahren üben, das den Schülervorschlägen am ehesten entspricht. Wesentlich ist bloß, dass im nächsten Schritt die Kosten von Vergleich und Bewegung getrennt analysiert werden und hier für die Systemumgebung der Klasse realistische Zeitangaben getroffen werden.

4. Komplexitätsabschätzung des elementaren Sortierverfahrens
5. Durchspielen eines O(n · ld(n)) Sortierverfahrens. Vorschlag: So4 – Mergesort

Für die Gesteinsblock-Analogie passt hier Mergesort (So4) sicherlich besser als Quicksort (So5). Für die informatischen Zielsetzungen kann man jedoch sowohl mit Mergesort oder Quicksort als kontrastierendem Verfahren fortsetzen. Mergesort hat wohl auch den Vorteil, dass es für Jugendliche leichter nachvollziehbar ist und auch in Alltagssituationen, etwa Sortieren von Belegen auf einem Tisch mit relativ kleiner Oberfläche in der Variante eines m-Wege-Merges (Vergleichsoperation nicht zwischen 2 sondern zwischen m Alternativen) stattfinden kann. Quicksort ist demgegenüber das elegantere und intellektuell anspruchsvollere Verfahren, durch das man schön die der Rekursion innewohnende Eleganz zeigen kann.

6. Analyse der Komplexität von Mergesort
7. Abschlussdiskussion mit Vergleich des Sortieraufwands

Detailüberlegungen zur Komplexitätsabschätzung (4. und 6.)
Ausgehend von der Motivation, überlegt man, wie viel die beiden mit Sortieren verbundenen elementaren Operationen an Zeit kosten:

- Vergleich (Wir können im Computer immer nur 2 Objekte vergleichen!) Es handelt sich um eine Elementaroperation im Mikrosekundenbereich. Weniger als 10 μs.
- Tausch (benötigt 3 Schritte *(wenn unklar ist, warum 3 Schritte, dann Dreieckstausch kurz ansprechen)*, der Einfachheit halber, ebenfalls etwa 10 μs.

Unter Rückgriff auf die in Schritt 3 bzw. Schritt 5 entwickelten Tafelskizzen berechnet man auf der Tafel in Schritten, die Nachvollziehbarkeit durch Kopfrechnen erlauben:

Für 100.000 Datensätze:

n Operationen:	$100.000\ 10\mu s = 10^5 \cdot 10 \cdot 10^{-6} sec = 1 sec$
n^2 Operationen:	$100.000 \cdot 100.000 \cdot 10\mu s = 10^5 \cdot 10^5 \cdot 10 \cdot 10^{-6} sec = 10^5 sec =$ $= 100.000 sec = 27{,}7$ Stunden ≈ 1 Tag, 3 Stunden
Zwischenrechnung:	$100.000 \leq 1024 \cdot 128 = 2^{10} \cdot 2^7 = 2^{17}$, daher $ld(100.000) \leq 17$
$n \cdot ld(n)$ Operationen:	$100.000 \cdot 17 \cdot 10\mu s = 10^5 \cdot 10 \cdot 2 \cdot 10 \cdot 10^{-6} sec = 20 sec$

Vergleichen wir dieses Ergebnis (20 sec) mit dem Ergebnis des zuerst angewendeten Sortierverfahrens, sieht man, dass sich der Aufwand, der in die Konstruktion eines leistungsfähigen Verfahrens gesteckt wurde, bei 100.000 und mehr Datensätzen jedenfalls lohnt. Wir können auch abschätzen, dass er sich schon früher (10.000) sicherlich lohnt, während bei Feldern kleinen Umfangs ein rasch entwickeltes elementares Verfahren wohl ausreichend leistungsfähig ist.