PAD Woche 6: Sortieralgorithmen II

Inhalte: Vergleichskomplexität, Merge Sort, Divide&Conquer Prinzip

Wiederholung:

- Selection Sort: Finde wiederholt die kleinste Karte des verbleibenden Stapels
- Komplexität: Anzahl elementarer Operationen

Vergleichskomplexität von Selection Sort:

→ Anzahl an Vergleichen für Minimum aus k Karten: k-1
→ Selection Sort für Stapel von n Karten: Minimum mit k von n bis 1
→ Die Gesamtzahl an Vergleichen ist höchstens:

```
 n-1 + n-2 + n-3 + … + 2 + 1 =                    (doppelt zählen)
(n-1 + n-2 + n-3 + … + 2 + 1 +
 n-1 + n-2 + n-3 + … + 2 + 1) / 2 =               (umformen )
(1   + 2   + 3   + … + n-3 + n-2 + n-1 +
 n-1 + n-2 + n-3 + … + 3   + 2   + 1) / 2 =       (umformen )
[(1 + n-1) + (2 + n-2) + (3 + n-3) + … + (n-1 + 1)] / 2 =
[n         + n         + n         + … + n        ] / 2 =
n*(n-1)/2
```

Aufgabe 1 (10 min):

- Entwickeln Sie einen Algorithmus der aus zwei gegebenen sortierten Stapeln einen einzigen sortierten Stapel erstellt.
- Natürlich könnte man einfach die beiden Stapel zusammenfügen und dann den Selection Sort Algorithmus ausführen, es gibt aber auch eine effizientere Lösung.

Merge Prozedur:

```
Lege den ersten gegebenen Stapel auf die linke Seite und den zweiten
gegeben Stapel auf die rechte Seite, so dass jeweils die Vorderseite nach
oben zeigt
Fange einen leeren Stapel in der Mitte an
Wiederhole die folgenden Anweisungen bis der linke oder der rechte Stapel
leer ist
      Vergleiche die oberste Karte des linken Stapels mit der obersten
      Karte des rechten Stapels und wähle die kleinere der beiden Karten
      aus
      Lege die ausgewählte Karte mit der Vorderseite nach unten auf den
      mittleren Stapel
Falls der linke oder der rechte Stapel noch nicht leer ist, lege den
ganzen verbleibenden Stapel mit der Vorderseite nach unten auf den
mittleren Stapel
Liefere den mittleren Stapel als Ergebnis zurück
```

Analyse Merge Prozedur:

- Nach jedem Vergleich wächst der mittlere Stapel um eine Karte
- Der mittlere Stapel enthält am Ende alle Karten
- Falls es k Karten gibt, können daher höchstens k Vergleiche durchgeführt werden

Merge Sort Algorithmus:

```
Falls der Stapel aus einer einzigen Karte besteht, tue nichts und liefere
die Karte als Ergebnis zurück
Andernfalls, führe die folgenden Anweisungen aus:
      Teile den gegebenen Stapel in zwei Hälften aus, deren Größe sich um
      höchstens eine Karte unterscheidet.
      Sortiere jeweils diese beiden Stapel (mit Merge Sort!)
      Führe die Merge Prozedur mit den beiden resultierenden Stapeln aus
      Liefere das Ergebnis der Merge Prozedur zurück
```

Aufgabe 2 (5 min)

- Führen Sie den Merge Sort Algorithmus mit 16 Karten aus.
- Zählen Sie mit, wie viele Vergleiche zweier Karten Sie dabei durchführen.

Divide&Conquer Prinzip:

- **Divide&Conquer**: Das Problem wird ein mehrere kleine Teilprobleme aufgeteilt deren Lösungen zu einer Gesamtlösung für das initiale Problem kombiniert werden.
- **Rekursion:** Algorithmus ruft sich selbst auf
- Divide&Conquer Algorithmen nutzen in der Regel Rekursion, aber können im Prinzip auch andere Algorithmen für die Teilprobleme verwenden.

Aufgabe 3 (20 min):

- Wie viele Vergleich führt Merge Sort für 4, 8, 16 Karten höchstens durch?
- Können Sie eine allgemeine obere Schranke für die Anzahl an Vergleichen angeben? (Sie dürfen annehmen, dass die Anzahl an Karten eine Zweierpotenz ist, also $n=2^i$)

Analyse Merge Sort:

- Analyse-Trick: Rekursionsbaum

```
                n                    1 Merge mit n Karten → ≤ n Vergleiche
      n/2               n/2          2 Merges mit n/2 Karten → ≤2*n/2=n Vergleiche
n/4       n/4     n/4       n/4      4 Merges mit n/4 Karten → ≤4*n/4=n Vergleiche
                  ...
2 2 2 2           ...       2 2 2    n/2 Merges mit 2 Karten → ≤n/2*2=n Vergleiche
```

- Auf jeder Ebene: höchstens n Vergleiche
- Wie viele Ebenen gibt es?
- Erste Ebene: n Karten, letzte Ebene: 2 Karten; Anzahl an Karten halbiert sich mit jeder Ebene
- Aufwärts-Analyse: $2^{\#Ebenen} = n \rightarrow \#Ebenen = \log_2(n)$
- Somit höchstens $n*\log_2(n)$ Vergleiche