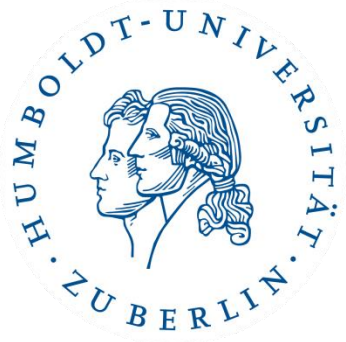
Übung Algorithmen und Datenstrukturen



Sommersemester 2016

Marc Bux, Humboldt-Universität zu Berlin

Agenda

1. Klärung von Fragen zum dritten Übungsblatt
2. Vorstellen des vierten Übungsblatts
3. Vorbereitende Aufgaben für das vierte Übungsblatt

Allgemeine Sortierverfahren

Ein allgemeines Sortierverfahren ist ein Sortierverfahren, welches nur Wissen über die Anordnung der zu sortierenden Elemente erlangen kann, indem es Elemente vergleicht. Ansonsten können keine weiteren Eigenschaften der Elemente vorausgesetzt bzw. ausgenutzt werden.

Welche der aus der Vorlesung bekannten Sortierverfahren sind allgemeine Sortierverfahren?

The Sound of Sorting:
https://www.youtube.com/watch?v=kPRA0W1kECg

Aufgabe (Schreibtischtest, lexikographische Ordnung)

Führen Sie einen Schreibtischtest für den Algorithmus Bubblesort für die folgenden Eingabe-Arrays durch. Geben Sie das Array A nach jedem Durchlauf der Zeilen 5-7 an.

1. $S = [6, 2, 4, 9, 1, 7]$

   Ordnung: natürliche Ordnung auf den natürlichen Zahlen

2. $S = [ogr, rov, obb, rrv, vgb, ogv]$

   Ordnung: lexikographische Ordnung auf $\Sigma = \{r, o, g, b, v\}$ mit $r < o < g < b < v$

Bubblesort(Array A)

Input: Array A von n ganzen Zahlen
Output: Array A aufsteigend sortiert.

```
1:  repeat
2:      swapped := false
3:      for i := 1 to n − 1 do
4:          if A[i] > A[i + 1] then
5:              temp := A[i]
6:              A[i] := A[i + 1]
7:              A[i + 1] := temp
8:              swapped := true
9:          end if
10:     end for
11: until not swapped
```

Lexikographische Ordnung für Zeichenketten aus Σ^m (wobei Σ ein geordnetes Alphabet ist): Es gilt $a_1, \dots, a_m < b_1, \dots, b_m$ g.d.w. ein i mit $1 \leq i \leq m$ existiert, so dass $a_i < b_i$ und $a_j = b_j$ für alle $j < i$.

Aufgabe (Schreibtischtest, Algorithmenanalyse)

Führen Sie einen Schreibtischtest für den Algorithmus PositionSort für das folgenden Eingabe-Array durch. Geben Sie nach jedem Durchlauf der for-Schleife mit Laufvariablen *i* das Array *B* an. Gehen sie davon aus, dass *B* mit den Werten 0 initialisiert ist.

1. $S = [5, 2, 7, 5, 5, 7]$

   Ordnung: natürliche Ordnung auf den natürlichen Zahlen

Erläutern Sie den Algorithmus.

PositionSort(S)

```
 1: n := |S|;
 2: B := neues Array der Länge n
 3: for i = 1 .. n do
 4:     pos := 1;
 5:     for j = 1 .. i − 1 do
 6:         if S[j] ≤ S[i] then
 7:             pos := pos+1;
 8:         end if
 9:     end for
10:     for j = i + 1 .. n do
11:         if S[j] < S[i] then
12:             pos := pos+1;
13:         end if
14:     end for
15:     B[pos] := S[i];
16: end for
17: return B;
```

Aufgabe (Stabilität)

In dieser Aufgabe betrachten wir Arrays mit Einträgen des abstrakten Datentyps Element. Grundlage dafür bilden eine Menge K von Schlüsseln, eine Menge V von Werten und eine lineare Ordnung $\leq$ auf K. Ein Element e hat einen Schlüssel e.key $\in K$ und einen Wert e.val $\in V$. Wir notieren ein Element e auch als Paar (e.key, e.val). Ist beispielsweise $K = \{a, b\}$ und $V = \mathbb{N}$, dann schreiben wir $(a, 17)$ für ein Element e mit e.key = a und e.val = 17.

Sind e_1 und e_2 vom Typ Element, können wir e_1 und e_2 auf Basis ihrer Schlüssel vergleichen: Es gelte genau dann $e_1 \leq e_2$, wenn e_1.key $\leq e_2$.key ist. Ist beispielsweise $K = \mathbb{N}$, $V = \{a, b, c\}$ und $\leq$ die natürliche Ordnung auf $\mathbb{N}$, dann gilt $(2, \mathrm{c}) \leq (4, a)$.

Ein Sortierverfahren heißt stabil, wenn Elemente mit gleichen Schlüsseln nach der Sortierung in der gleichen Reihenfolge aufeinander folgen wie vor der Sortierung.

Entscheiden Sie, ob die folgenden Verfahren stabil sind. Falls das Verfahren nicht stabil ist, geben Sie eine (bitte möglichst kleine) Instanz als Gegenbeispiel an. Begründen Sie auf jeden Fall Ihre Entscheidung.

1. PositionsSort
2. SelectionSort
3. InsertionSort
4. QuickSort
5. MergeSort
6. BucketSort

Aufgabe (Sortierung spezieller Arrays)

Beweisen oder widerlegen Sie: Es gibt ein Sortierverfahren, welches ein Array von n beliebigen (vergleichbaren) Elementen in Laufzeit $O(n)$ sortiert, falls …

1. … im Array nur Zahlen zwischen 1 und 1000 vorkommen.
2. … die ersten drei Viertel des Arrays bereits sortiert sind.

Ausblick: Nächste Woche

- Klärung von Fragen zum vierten Übungsblatt
- Besprechung der Lösungen zum dritten Übungsblatt
- Fragen?