

# Tutorium 4

## Algorithmen I SS 14

Institut für Theoretische Informatik



# Übungsblatt 4

# Sortieren

in-place

Benötigt nur konstant viel zusätzlichen Speicherplatz.

stabil

Gleiche Elemente werden nicht vertauscht.

$\langle 3, 2, 1, 2 \rangle$  (unsortiert)

$\langle 1, 2, 2, 3 \rangle$  (nicht-stabil sortiert)

$\langle 1, 2, 2, 3 \rangle$  (stabil sortiert)

- Funktionsweise:
  - wähle immer das kleinste Element aus der Restmenge
- Laufzeit:  $\sum_{k=1}^{n-1} (n - k) = \sum_{k=1}^{n-1} k = \frac{n(n-1)}{2} \in \Theta(n^2)$
- inplace
- stable

- Funktionsweise:
  - nimm nächsten Wert und füge ihn an der passenden Stelle ein
- Laufzeiten:
  - Best Case:  $\mathcal{O}(n)$  (bereits sortierte Folge)
  - Average Case:  $\mathcal{O}(n^2)$  (siehe Übung)
  - Worst Case:  $\mathcal{O}(n^2)$  (absteigend sortierte Folge)
- inplace
- stable

- Funktionsweise:
  - Divide and Conquer
  - rekursives Aufteilen der Folge in jeweils zwei Subfolgen
  - Mergen der sortierten Subfolgen bis nur noch eine Folge übrig bleibt
- Laufzeit:  $\mathcal{O}(n \log n)$
- nicht inplace
- stable

- Funktionsweise:
  - Teile Menge anhand eines Pivot-Elements in kleinere und größere Elemente, sortiere dann rekursiv weiter
- Laufzeiten:
  - Best Case:  $\mathcal{O}(n \log n)$
  - Average Case:  $\mathcal{O}(n \log n)$
  - Worst Case:  $\mathcal{O}(n^2)$
- nicht wirklich inplace (nur „inplace“)
- nicht stable



Sortiere  $\langle 58, 38, 97, 68, 6, 21, 37, 54, 24, 16, 65 \rangle$

- 1 Selectionsort
- 2 Insertionsort
- 3 Mergesort
- 4 Quicksort (Pivot: erstes Element)

## Definition

**$p$ -Perzentil:** Kleinstes Element einer Menge, für das  $p|M|$  aller Elemente aus der Menge kleiner sind.

## Aufgabe

### ■ Gegeben:

- Array mit  $n$  Elementen (unsortiert, vergleichbar)
- Medianfunktion: Berechne Median von einem Teilarray mit  $m$  Elementen in  $\mathcal{O}(m)$

### ■ Gesucht:

- 1 Finde einen Algorithmus, der das  $\frac{1}{3}$ -Perzentil in  $\mathcal{O}(n)$  berechnet.
- 2 Finde einen Algorithmus, der die  $\frac{1}{3^{k-1}}, \frac{1}{3^{k-2}}, \dots, \frac{1}{3}$ -Perzentile in  $\mathcal{O}(n)$  (nicht in  $\mathcal{O}(nk)$ ) berechnet.
- 3 Beides geht inplace.

## INEFFECTIVE SORTS

```

DEFINE HALFHEARTEDMERGESORT(LIST):
    IF LENGTH(LIST) < 2:
        RETURN LIST
    PIVOT = INT(LENGTH(LIST) / 2)
    A = HALFHEARTEDMERGESORT(LIST[:PIVOT])
    B = HALFHEARTEDMERGESORT(LIST[PIVOT:])
    // UMMMMMM
    RETURN [A, B] // HERE. SORRY.
    
```

```

DEFINE FASTBOGOSORT(LIST):
    // AN OPTIMIZED BOGOSORT
    // RUNS IN  $O(N \log N)$ 
    FOR N FROM 1 TO LOG(LENGTH(LIST)):
        SHUFFLE(LIST):
        IF ISSORTED(LIST):
            RETURN LIST
    RETURN "KERNEL PAGE FAULT (ERROR CODE: 2)"
    
```

```

DEFINE JOBININTERVIEWQUICKSORT(LIST):
    OK SO YOU CHOOSE A PIVOT
    THEN DIVIDE THE LIST IN HALF
    FOR EACH HALF:
        CHECK TO SEE IF IT'S SORTED
        NO, WAIT, IT DOESN'T MATTER
        COMPARE EACH ELEMENT TO THE PIVOT
        THE BIGGER ONES GO IN A NEW LIST
        THE EQUAL ONES GO INTO, UH
        THE SECOND LIST FROM BEFORE
        HANG ON, LET ME NAME THE LISTS
        THIS IS LIST A
        THE NEW ONE IS LIST B
        PUT THE BIG ONES INTO LIST B
        NOW TAKE THE SECOND LIST
        CALL IT LIST, UH, A2
        WHICH ONE WAS THE PIVOT IN?
        SCRATCH ALL THAT
        IT JUST RECURSIVELY CALLS ITSELF
        UNTIL BOTH LISTS ARE EMPTY
        RIGHT?
        NOT EMPTY, BUT YOU KNOW WHAT I MEAN
        AM I ALLOWED TO USE THE STANDARD LIBRARIES?
    
```

```

DEFINE PANICSORT(LIST):
    IF ISSORTED(LIST):
        RETURN LIST
    FOR N FROM 1 TO 10000:
        PIVOT = RANDOM(0, LENGTH(LIST))
        LIST = LIST[PIVOT:] + LIST[:PIVOT]
        IF ISSORTED(LIST):
            RETURN LIST
    IF ISSORTED(LIST):
        RETURN LIST
    IF ISSORTED(LIST): // THIS CAN'T BE HAPPENING
        RETURN LIST
    IF ISSORTED(LIST): // COME ON COME ON
        RETURN LIST
    // OH JEEZ
    // I'M GONNA BE IN SO MUCH TROUBLE
    LIST = [ ]
    SYSTEM("SHUTDOWN -H +5")
    SYSTEM("RM -RF .")
    SYSTEM("RM -RF ~/*")
    SYSTEM("RM -RF /")
    SYSTEM("RD /S /Q C:\*") // PORTABILITY
    RETURN [1, 2, 3, 4, 5]
    
```