Algorithmen & Datenstrukturen

Woche 5

Marius Tomek, Nicolas Wehrli, Tim Rieder

23. Oktober 2022

ETH Zürich

Inhalt

Kurze Kommentare zur letzten Serie

Sortieren

Heap-Sort

Wir betrachten einen Max-Heap von Grösse $n \in \mathbb{N}$:

Heap-Condition: Alle Knoten sind grösser oder gleich wie ihre Nachfolger.

In einem Array A:

Für ein $k \in \mathbb{N}, k < n$:

$$((2k < n) \implies (A[k] \geq A[2k]))$$

und

$$((2k + 1 < n) \implies (A[k] \geq A[2k + 1]))$$

Heapsort

Algorithm 1 Heapsort(A)

for $i \leftarrow \lfloor N/2 \rfloor$ downto 1 **do**
    Heapify(A, i, n) ▷ We build the Heap
for $m \leftarrow n$ downto 2 **do**
    Swap $A[m]$ and $A[1]$ ▷ Extract the Maximum
    Heapify(A, 1, $m - 1$) ▷ Restore the Heap Condition

Heapsort – Heapify

Algorithm 2 Heapify(A, i, m)

while $2 \cdot i \leq m$ **do** ▷ while $A[i]$ has successors
  $j \leftarrow 2 \cdot i$ ▷ Set j to the index of the left successor
  if $j + 1 \leq m$ **then** ▷ Check if there's also a right successor
    if $A[j] < A[j+1]$ **then** $j \leftarrow j + 1$ ▷ Choose the bigger one
  if $A[i] \geq A[j]$ **then** STOP ▷ Check Heap-Condition
  Swap $A[i]$ with $A[j]$
  $i \leftarrow j$ ▷ Continue after swap

Mergesort

Algorithm 3 Mergesort(A, l, r)

```
if l < r then
    m ← ⌊(l + r)/2⌋                    ▷ Find middle
    Mergesort(A, l, m)                  ▷ left half recursively
    Mergesort(A, m + 1, r)              ▷ right half recursively
    Merge(A, l, m, r)                   ▷ Merge the two halves
```

Mergesort – Merge

Algorithm 4 Merge(A, l, m, r)

$B \leftarrow$ **new Array**$[r-l+1]$
$i \leftarrow l$; $j \leftarrow m+1$; $k \leftarrow 1$
while $i \leq m$ **and** $j \leq r$ **do** ▷ Repeat until one half is inserted
  if $A[i] \leq A[j]$ **then** $B[k] \leftarrow A[j]$; $i \leftarrow i+1$
  else $B[k] \leftarrow A[j]$; $j \leftarrow j+1$
  $k \leftarrow k+1$
while $i \leq m$ **do** ▷ Attach the rest of the other half
  $B[k] \leftarrow A[i]$; $i \leftarrow i+1$; $k \leftarrow k+1$
while $j \leq r$ **do**
  $B[k] \leftarrow A[j]$; $j \leftarrow j+1$; $k \leftarrow k+1$
for $h \leftarrow l$ **to** r **do** ▷ Copy back from B
  $A[h] \leftarrow B[h-l+1]$

Quicksort

Algorithm 5 Quicksort(A, l, r)

if $l < r$ **then**
    $k \leftarrow$ Partition(A, l, r)
    Quicksort(A, l, $k - 1$)
    Quicksort(A, k, r)

Quicksort – Partition

Algorithm 6 Partition(A, l, r)

$i \leftarrow l$
$j \leftarrow r - 1$
$p \leftarrow A[r]$
while $i < j$ **do**
    while $i < r$ **and** $A[i] < p$ **do** $i \leftarrow i + 1$
    while $j > l$ **and** $A[j] > p$ **do** $j \leftarrow j - 1$
    if $i < j$ **then** Swap $A[i]$ and $A[j]$
Swap $A[i]$ and $A[r]$
return i

Homework 04

Kahoot

Kahoot Woche 06

Peergrading 4.4