Datenstrukturen, Algorithmen und Programmierung 2

Amin Coja-Oghlan

April 27, 2023

Lehrstuhl Informatik 2
Fakultät für Informatik

Heapsort

Sortieren nochmal

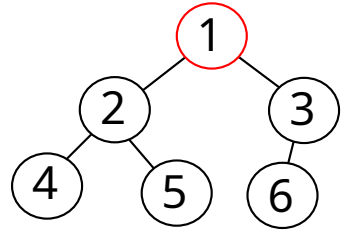
- mit Quicksort haben wir einen (in Erwartung) schnellen randomisierten Sortieralgorithmus kennengelernt
- mit Heapsort lernen wir einen effizienten *deterministischen* Sortieralgorithmus kennen
- Hilfsmittel ist eine (ansatzweise) raffinierte Datenstruktur

Heapsort

Heaps

- ein Heap ist eine Datenstruktur in Form eines gewurzelten Baums
- die Knoten sind von 1 bis n durchnumeriert
- Knoten 1 ist die Wurzel
- die Kinder von Knoten i sind $2i$ und $2i+1$, falls diese Werte $\leq n$ sind
- jeder Knoten hat also höchstens zwei Kinder
- der Elternknoten von Knoten $i \geq 2$ ist der Knoten $\lfloor i/2 \rfloor$

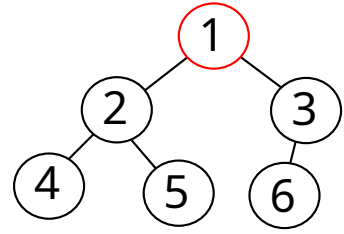
Heapsort



Beispiel

- ein Heap mit 6 Knoten
- Knoten 1 ist der Wurzelknoten

Heapsort



Repräsentation im Speicher

- ein Heap kann im Speicher einfach in einem Array $\boldsymbol{A} = (A_1, \ldots, A_n)$ der Länge n dargestellt werden
- die Indizierung der Eltern/Kinder wird mit den Speicherstellen des Arrays identifiziert

Heapsort

Max-Heaps (und Min-Heaps)

- in den Knoten des Heaps speichern wir *vergleichbare Werte*
- ein Max-Heap hat die Eigenschaft, daß der Wert eines Kindknotens niemals größer ist als der Wert des Elternknotens
- ein Min-Heap hat die Eigenschaft, daß der Wert eines Kindknotens niemals kleiner ist als der Wert des Elternknotens
- *wir befassen uns im folgenden nur mit Max-Heaps*

Heapsort

Erzeugen eines Max-Heaps

- *Aufgabe:* ein Array $\boldsymbol{A} = (A_1, \ldots, A_n)$ soll in einen Max-Heap überführt werden
- `BuildMaxHeap`$(\boldsymbol{A})$ löst dieses Problem für uns
- Hilfsfunktion `MaxHeapify`$(\boldsymbol{A}, i)$: die Nachkommen von Knoten i sind bereits Max-Heaps, aber möglicherweise ist der Wert A_i kleiner als der Wert eines Kindes

Heapsort

`MaxHeapify`($\boldsymbol{A}, i$)

1. Falls A_i nicht kleiner ist als seine Kinder (oder kein Kind hat), halte.
2. Sonst bestimme das größere Kind A_j
3. Vertausche die Werte A_i und A_j
4. `MaxHeapify`($\boldsymbol{A}, j$)

Heapsort

Definition

Die Höhe eines Knotens i ist der maximale direkte Abstand von einem Blatt.

Lemma

Die Höhe der Wurzel ist $O(\log n)$.

Heapsort

Proposition

Wenn der Knoten i Höhe h hat, benötigt `MaxHeapify`$(\boldsymbol{A}, i)$ Laufzeit $O(h)$.

Korollar

`MaxHeapify` hat Laufzeit $O(\log n)$.

Heapsort

Aufbau eines max-heaps aus einem Array

- aus einem beliebigen gegebenen Array A möchten wir einen max-heap machen
- dazu arbeiten wir “von hinten nach vor”
- die letzte Hälfte der Einträge werden jedenfalls Blätter
- auf die anderen Einträge wenden wir `MaxHeapify` an

Heapsort

`BuildMaxHeap`($\boldsymbol{A}$)

1. Für $i = \lfloor n/2 \rfloor, \ldots, 1$
2. `MaxHeapify`($\boldsymbol{A}, i$)

Proposition

`BuildMaxHeap` hat Laufzeit $O(n)$

Heapsort

`Heapsort`(A)

1. `BuildMaxHeap`(A)
2. Für $i = n, n-1, \ldots, 2$
3. vertausche A_1 und A_i
4. `MaxHeapify`$((A_1, \ldots, A_{i-1}), 1)$

Satz

`Heapsort` sortiert ein gegebenes Array in Zeit $O(n \log n)$

Heapsort

Priority queues

- max-heaps haben weitere Anwendungen
- wir versehen die Datenstruktur mit weiteren Operationen
- `ExtractMax` (findet und) entfernt das maximale Element
- `IncreaseKey` erhöht den Wert eines Elements
- `Insert` fügt ein neues Element mit einem gegebenen Wert ein

Heapsort

`ExtractMax`($\boldsymbol{A}$)

1. falls $n = 0$, abbrechen; falls $n = 1$, gib $A_1, \varnothing$ aus
2. vertausche A_1 und A_n
3. `MaxHeapify`$((A_1, \ldots, A_{n-1}), 1)$
4. gib A_n und $(A_1, \ldots, A_{n-1})$ aus

Heapsort

`IncreaseKey`$(\boldsymbol{A}, i, \alpha)$

1. falls $\alpha < A_i$, brich ab
2. setzte $A_i = \alpha$
3. solange $i > 1$
4. setze $j = \lfloor i/2 \rfloor$ # j =Elternknoten von A_i
5. falls $A_j \geq \alpha$, halte
6. falls $A_j < \alpha$, vertausche A_j und α
7. setze $i = j$

Heapsort

`Insert`(A, α)

1. füge ein Element $A_{n+1} = -\infty$ zu A hinzu
2. wende `IncreaseKey`$((A_1, \ldots, A_{n+1}, n+1, \alpha)$ an

Heapsort

Proposition

Die Operationen `ExtractMax`, `IncreaseKey`, `Insert` haben Laufzeit $O(\log n)$.

Heapsort

Zusammenfassung

- `Heapsort` ist ein deterministischer Sortieralgorithmus mit Laufzeit $O(n \log n)$
- wesentlicher Baustein ist die max-heap Datenstruktur
- diese haben wir zu einer Priority Queue erweitert