Data Structures and Algorithms I

Homework Assignment 4 - Halden

Wachmann Elias

7. November 2021

1 Aufgabenstellung

A d-ary heap is like a binary heap, but nonleaf nodes have d children instead of 2 children.

(A) How would you represent a d-ary heap in a array?

(B) What is the height of a d-ary heap of n elements in terms of n and d? Justify your answer.

(C) Give an efficient implementation of `HEAPIFY` in a d-ary max-heap. Analyze its running time in terms of d and n.

2 A - Darstellung in einem Array

Gleich wie bei einer binären Halde wird der Maximale wert an den Anfang des Arrays (im folgendem `A`), also `A[0]` gespeichert. In einer d-äre Halde hat nun jeder Knoten (außer leafs) d Kinder. Diese d Kinder werden nun an die nächsten d Stellen im eindimensionalen Array gespeichert. Damit sind die Kinder der Wurzel `A[0]` im Array an den Stellen `A[1]` bis `A[d]`. Die Nächste Ebene im Baum hat $d \cdot d$ Kinder, welche im Array von `A[d+1]` bis $\texttt{A[d}^2\texttt{+d]}$. Abermals die nächste von $\texttt{A[d}^2\texttt{+d+1]}$ bis $\texttt{A[d}^3\texttt{+d}^2\texttt{+d]}$ Allgemein liegt die n-te Ebene (Wurzel ist 0. Ebene) von $\texttt{A[}\sum_{i=0}^{n}\texttt{d}^i\texttt{]}$ bis $\texttt{A[(}\sum_{i=0}^{n+1}\texttt{d}^i\texttt{)-1]}$ im 0-indiziertem Array.

3 B - Höhe einer d-ären Halde

Wie schon in Abschnitt 2 ersichtlich ist, wächst eine d-ären Halde um h^d - Kinder wobei h die Ebene/Höhe angibt. Die Anzahl n der in einer Halde gespeicherten Elemente liegt nun sicherlich wie folgt:

$$
\begin{aligned}
1 + \sum_{i=0}^{h-1} d^i \leq n \leq \sum_{i=0}^{h} d^i \\
1 + \frac{d^h - 1}{d-1} \leq n \leq \frac{d^{h+1} - 1}{d-1}
\end{aligned}
$$

Nimmt man nun davon $\log_d$ und formt beide Seiten jeweils auf h um erhält man:

$$
\begin{aligned}
\log_d(n(d-1)+1) - 1 \leq h \leq \log_d((n-1)(d-1)+1) \\
\log_d(n(d-1)+1) \leq h + 1 \rightarrow \lfloor \log_d(n(d-1)+1) \rfloor = h \rightarrow h = \Omega(\log_d(n)) \\
\log_d((n-1)(d-1)+1) \geq h \rightarrow \lceil \log_d((n-1)(d-1)+1) \rceil = h \rightarrow h = \mathcal{O}(\log_d(n))
\end{aligned}
$$

Daraus folgt nun schließlich, dass die Höhe sich zu

$$h = \Theta(\log_d(n))$$

ergibt.

4 C - Implementation von HEAPIFY

Es folgt eine Implementation von `HEAPIFY`.

Algorithm 1 `HEAPIFY`

```
function HEAPIFY(A,i,d)
    // Input: A array to heapify
    // i index which should be heapified
    // d order of d-ary heap
    n ← length of A
    kids ← [ ]                                   ▷ Setup empty array for kid indices
    for count ← 1 to d do                        ▷ including d
        kids[count-1] ← d*i+count
    index ← i
    for count ← 0 to (length of kids)-1 do
        if kid < n and A[kids[count]] > A[index] then
            index ← k
    if i does not equal index then
        switch A[index] and A[i]
        HEAPIFY(A,index,d)
```

4.1 Laufzeitanalyse von HEAPIFY

Bla Bla Bla