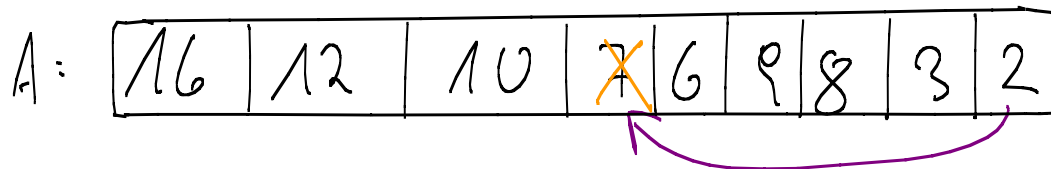
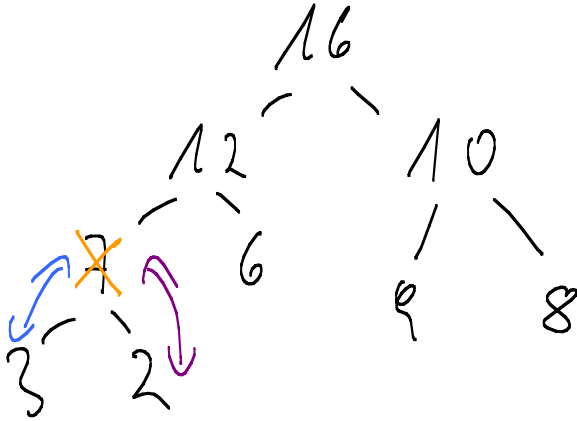


1. Die Operation $HEAPDEL(A, i)$ löscht das i -te Element $A[i]$ der Halde A . Dadurch kann die Haldebedingung verletzt werden, die wieder hergestellt werden muß. Geben Sie eine Implementation von $HEAPDEL$ mit einer Laufzeit von $O(\log n)$ an (n ist die Anzahl der Elemente der Halde A). Erklären Sie genau und verständlich Ihren Algorithmus. Beweisen Sie die Laufzeitschranke für Ihren Algorithmus.



- 1 HEAPDEL (A,i) $\rightarrow \Theta(n)$
- 2 $A[i] = A[n]$ $\rightarrow \Theta(1)$
- 3 $n = n - 1$ $\rightarrow \Theta(1)$
- 3 Verhalde (A,n) $\rightarrow \Theta(\log(n))$ siehe Vo

- 1.) Das letzte Element an die Stelle i bringen
- 2.) Das Feld um eine Stelle verringern
(\sim das letzte Element wird nicht mehr berücksichtigt)
- 3.) die Halde neu aufbauen

Laufzeit: $T(n) = 3 \cdot \Theta(1) + \Theta(\log(n))$

$$T(n) = \Theta(\log(n))$$

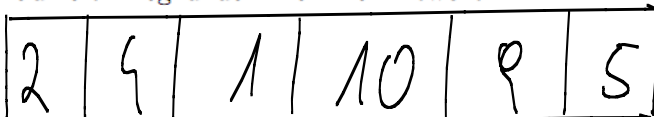
2. Sei $A[1, \dots, n]$ ein (unsortiertes) Feld das natürliche Zahlen enthält. Die Partition Funktion von Quicksort kann verwendet werden um die i -kleinste Zahl des Feldes A zu bestimmen (d.h. die Zahl die an i -ter Stelle stehen würde wenn das Feld sortiert wäre). Der Median eines Feldes $A[1 \dots n]$ für ungerades n ist die Zahl die an Stelle $\lceil \frac{n}{2} \rceil$ stehen würde wenn das Feld sortiert wäre. Der Algorithmus sieht folgendermaßen aus:

```

ALGO(A, l, r, i)
1: IF l=r
2:   Return A[l]
3: k = Partition(A,l,r)
4: s = k-1+1
5: IF i <= s
6:   Return ALGO(A,l,k,i)
7: ELSE
8:   Return ALGO(A,k+1,r,i-s)

```

Warum funktioniert dieser Algorithmus? Geben Sie eine genaue Erklärung an. Die Laufzeit hängt von der Auswahl des Pivotelementes ab. Angenommen man nimmt in Partition immer das erste Element des übergebenen Arrays als Pivotelement. Wie ist dann die beste und die schlechteste Laufzeit? Begründen Sie Ihre Antwort.



$Algo(A, 1, 6, 5)$



Durch den rekursiven Aufruf von Algo wird das pivot Element so lange verschoben bis es an der i -ten Stelle steht; \Rightarrow das ist k ; Zeile 2

Da Partition läuft bis sich l und k überschneiden;
 \Rightarrow liefert das i kleinste

Zeile 01-02:

Abbruchbedingung;

Es wurde l so lange verschoben bis es die Zahl an i -ter Stelle repräsentiert

Zeile 03:

Partition liefert in Abhängigkeit vom Pivotelement jene Stelle an der eine Zahl $<$ pivot steht

Zeile 04:

Neuberechnung eines Startplatzes

Zeile 03 + 04:

Es wird ein Teilfeld gebildet das von Algo noch einmal durchsucht wird

Zeile 05-08:

Es wird entweder das obere, oder das untere Teilfeld durchsucht

$$\text{ALGO}(A, l, r, i) \longrightarrow \mathcal{O}(n)$$

$$1: \text{ IF } l=r \longrightarrow \mathcal{O}(1)$$

$$2: \text{ Return } A[l] \longrightarrow \mathcal{O}(1)$$

$$3: k = \text{Partition}(A, l, r) \longrightarrow \mathcal{O}(n) \text{ siehe Vo}$$

$$4: s = k-1+1 \longrightarrow \mathcal{O}(1)$$

$$5: \text{ IF } i \leq s \longrightarrow \mathcal{O}(1)$$

$$6: \text{ Return } \text{ALGO}(A, l, k, i) \longrightarrow T(h)$$

$$7: \text{ ELSE } \longrightarrow \mathcal{O}(1)$$

$$8: \text{ Return } \text{ALGO}(A, k+1, r, i-s) \longrightarrow T(n-h)$$

$$\Rightarrow T(n) = 5 \mathcal{O}(n) + \mathcal{O}(n) + T(h) + T(n-h)$$

Best Case: $h = \frac{n}{2}$ in jedem Schritt

$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + \mathcal{O}(n)$$

$$T(n) = 2T\left(\frac{n}{2}\right) + \mathcal{O}(n)$$

$$\Rightarrow T(n) = \mathcal{O}(n \log(n))$$

Worst Case: $h = 1$

$$T(n) = T(1) + T(n-1) + \mathcal{O}(n)$$

$$= \mathcal{O}(n) + T(n-1) + \mathcal{O}(n)$$

$$T(n) = T(n-1) + \mathcal{O}(n)$$

$$T(n-1) = T(n-2) + \mathcal{O}(n-1)$$

$$T(n) = T(n-3) + \mathcal{O}(n-2) + \mathcal{O}(n-1) + \mathcal{O}(n)$$

$$\Rightarrow = T(n-h) + \sum_{i=0}^{h-1} \mathcal{O}(n-i)$$

$$n-h \stackrel{!}{=} 1 \Rightarrow h = n-1$$

$$= T(1) + \sum_{i=0}^{n-2} \mathcal{O}(n-i)$$

$$= \mathcal{O}(n) + \mathcal{O}(n-1) + \dots + 3+2$$

$$= \mathcal{O}\left(\frac{n(n+1)}{2} - 1\right)$$

$$= T(n^2)$$

siehe Vo für Quick sort
pivot muss $A[l]$;
 $A[r] > A[l]$ muss $<$ pivot sein

Feld aufsteigend sortiert
und $i = n$, pivot = $A[l]$

\Rightarrow "Best Case = Worst Case"

i -tes
Element

steht

ganz

rechts: $A[n]$

$$2^n = 1 = n = 2^h \quad | \text{ log}$$

$$\text{log}(n) = h \cdot 1$$

$$\Rightarrow h = \text{log}(n)$$