Kontaktstudium IMP
Lerneinheit Informatik 9
Zusatzaufgaben

Anwendung

Aufgabe 1: *QuickSort auf Arrays*

Sei gegeben das Array A:

8	3	9	2	5	4	1	7	6

Sortieren Sie A mit QuickSort. Veranschaulichen Sie den Ablauf so wie in der Darstellung des Beispiels auf den Folien, so dass man den Ablauf der rekursiven Aufrufe und die Positionen des Pivots p sowie der Pointer i und j verfolgen kann.

8 p	3 i	9	2	5	4	1	7	6 j
8 p	3	6 i	2	5	4	1	7	9 j
7 p	3	6	2	5	4	1	**8** j	9 i
7 p	3 i	6	2	5	4	1 j	**8**	9
1 p	3	6	2	5	4	**7** j	**8** i	9
1 p	3 i	6	2	5	4 j	**7**	**8**	9
1 p,j	3 i	6	2	5	4	**7**	**8**	9
1	3 p	6 i	2	5	4 j	**7**	**8**	9

1	3 p	2 i	6 j	5	4	**7**	**8**	9
1	2 p	**3** j	6 i	5	4	**7**	**8**	9
1	**2**	**3**	6	5	4	**7**	**8**	9
1	**2**	**3**	6 p	5 i	4 j	**7**	**8**	9
1	**2**	**3**	4 p	5	**6** j	**7** i	**8**	9
1	**2**	**3**	4 p	5 i,j	**6**	**7**	**8**	9
1	**2**	**3**	**4** p,j	5 i	**6**	**7**	**8**	9
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	9
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**

Transfer

Aufgabe 2: *Santa's Dirty Socks*

a) Beschreiben Sie das Verfahren des kleinen Elfen in Pseudocode oder als Struktogramm.

Der folgender Algorithmus findet (für eine Zweierpotenz n) in einer n-elementigen Menge von gleichwertigen Elementen ein einzelnes mit höherem Wert.
Die Werte seinen in einem Array $A[1...n]$ gegeben für ein $n = 2^p$. Für die Eingabe sei garantiert, dass es ein k gibt mit $A[i] < A[k]$ für alle $i \neq k$ und $A[i] = A[j]$ für alle $i, j \neq k$.
Die Methode `weight`$(A[i, j])$ gibt dabei in $\mathcal{O}(1)$ die Summe aller Werte zwischen den Stellen i und j aus.

Algorithm1: SockSearch

Aufruf: `sockSearch`$(A, 1, n)$

```
sockSearch(A, l, r) begin
    while l ≠ r do
        m ← (l+r−1)/2
        if weight(A[l, m]) < weight(A[m + 1, r]) then
            l ← m + 1
        else
            r ← m
    return (gefunden an Stelle l)
```

b) Beschreiben Sie (als Text) ein Verfahren, welches auch für n Boxen funktioniert, wenn n keine Zweierpotenz ist und höchstens doppelt so viel Vergleiche braucht, wie das Verfahren des kleinen Elfen für die nächst größere Zweierpotenz.

Eine Mögleichkeit ist, das evtl. übrige Paket jedes Mal erst mit einem weiteren Paket zu vergleichen und wenn beide gleich sind, kann man das übrige ausschliessen und mit den beiden Hälften ganz normal weiter machen. So hat man jedes Mal einen zuätzlichen Wiegevorgang, falls man nicht gleichmässig teilen kann. Bei 15 Boxen zum Beispiel wären das ingesamt 7 Wiegevrgänge, wenn die Socken nie in der übrigen Box sind.

Noch besser ist die Lösung von Teilnehmer Björn Braun:

Ganze doppelt so viel wie für fast zwei mal mehr?
Das unterbiete ich deutlich, also hier, bitte sehr:
Man verteile die Päckchen wie zuvor auf zwei Haufen,
gehts nicht auf, bleibt eins übrig, darauf wird's hinaus laufen.
Schlägt beim Wiegen der Haufen die Waage dann aus,
ist der leichtere Haufen und das Restpaket raus.
Bleibt stattdessen die Waage in der Mitte fest stehen
Sind im Restpaket Socken, wer es öffnet, wird's sehen.
Wiederholt man nun dieses, bis die Socken gefunden,
Braucht's maximal log zwei n, doch gerundet nach unten!
(Und das statt zwei log zwei n, gerundet nach oben,
doch wer hat so viel Zeit schon, die Gewerkschaft tät toben)

Doch wenn die Elfen protestieren,
weil auch $\log_2 n$ zu viel
kann ich's noch weiter reduzieren
und nehme log drei n zum Ziel:
Man teilt was ist in drei Portionen
und rundet, dass nichts übrig bleibt.
Zwei sind dann gleich, hier wird's sich's lohnen,
dass wiegend man Vergleich betreibt.
Geht hier ein Stapel dann nach unten,
gehts nur mit diesem weiter dann.
Sonst mit dem dritten, bis gefunden,
die Schmutzwäsche vom Weihnachtsmann.
Und weil man hier den Rest stets drittelt
(Plus minus ein Paket als Rest,
was sich auf Dauer aber mittelt)
Steht's Ziel schon nach $\log_3 n$ fest.

Programmierung

Aufgabe 3: *Sortieren mit Scratch*

Programmieren Sie in Scratch ein Verfahren, welches erst 10 zufäällige Zahlen zwischen 1 und 20 in ein Array (nutzen Sie die Scratch-Datenstruktur wie ein Array) einfügt und dann mit einer rekursiven Variante von InsertionSort sortiert.

Falls Sie auch andere rekursive Verfahren in Scratch implementieren möchten, werden Sie bei Verfahren mit zwei rekursiven Aufrufen an die Grenzen von Scratch stossen und müssen ggf. selbst einen Rekursionsstack anlegen.