

Aufgabe 15: Simultane Suche nach Maximum und Minimum

1. Das Problem, das Maximum in einer Liste (von ganzen Zahlen etwa) durch paarweisen Grössenvergleich herauszufinden, genannt MAX, ist eines der einfachsten algorithmischen Probleme dessen Komplexitätsanalyse zudem sehr einfach ist:
 - (a) Welcher Algorithmus zur Lösung dieser Aufgabe kommt Ihnen spontan in den Sinn? Wieviel paarweise Vergleiche von Listenelementen benötigt Ihr Algorithmus im “worst case” bei Listen der Länge n ?
 - (b) Vermutlich kommt Ihr Algorithmus mit $n - 1$ paarweisen Vergleichen aus. Wenn nicht, suchen Sie nach einem Algorithmus, der das leistet. Es gibt hierfür verschiedene Möglichkeiten. Denken Sie z.B. ganz aktuell an Tennisturniere.
 - (c) Zeigen Sie, dass $n - 1$ tatsächlich auch die untere Schranke für die “worst-case”-Komplexität ist: besser geht es nicht!

2. Stellen Sie sich jetzt das Problem, sowohl das grösste als auch das zweitgrösste Element einer Liste ausfindig zu machen. Überlegen Sie, wie sie das, evtl. unter Verwendung Ihres Algorithmus für die Maximumsuche, machen können. Aber der Mehraufwand soll natürlich möglichst gering sein!

3. Jetzt zu einer anderen Variante: es geht darum, gleichzeitig das grösste und das kleinste Element einer Liste ausfindig zu machen. Dieses Problem soll MAXMIN heissen.
 - (a) Welche naheliegende, jedoch möglicherweise kostspielige Lösung für MAXMIN fällt Ihnen zunächst ein?
 - (b) Gehen Sie jetzt das Problem mit einem divide-and-conquer-Ansatz an! Der Einfachheit halber sei die Länge n der gegebenen Liste A zunächst als $n = 2^k$ mit $k \geq 1$ angenommen.
 - Man zerlegt die Liste A in zwei Hälften A_1 und A_2 ,
 - Man bestimmt in beiden Teillisten (durch rekursiven Aufruf eben dieser Methode) die Maxima M_1 und M_2 und die Minima m_1 und m_2
 - Man bestimmt schliesslich durch zwei weitere Vergleiche das Maximum M und das Minimum m .

Der Kostenaufwand dieser rekursiven Prozedur `maxmin`, d.h. die Anzahl der benötigten Vergleichsoperationen, wird mit $K(n)$ bezeichnet.

Nach Konstruktion gilt:

$$K(n) = \begin{cases} 1 & \text{für } n = 2, \\ 2 \cdot K(n/2) + 2 & \text{für } n > 2. \end{cases}$$

Bestimmen sie die exakte Lösung dieser Rekursion, wenn n über Potenzen von 2 läuft.

- (c) Jetzt sei n eine beliebige natürliche Zahl. Geben Sie unter Verwendung eines optimalen Algorithmus für MAX, der natürlich eine Entsprechung für das Problem MIN hat, einen Algorithmus für das Problem MAXMIN an, der exakt $\lceil \frac{3n}{2} \rceil - 2$ Vergleichsoperationen benötigt. Dabei ist zu unterscheiden,

ob n gerade oder ungerade ist.

(NB. Auch hier kann gezeigt werden, dass die angegebene Zahl eine untere Schranke für das Problem MAXMIN ist. Einen Beweis für diese nichttriviale Tatsache findet man z.B. in dem Buch *Combinatorial Search* von M.Aigner, p.225 ff.)

Im folgenden wird diese Kostenfunktion mit

$$K_{optimal}(n) := \left\lceil \frac{3n}{2} \right\rceil - 2$$

bezeichnet.

- (d) Für beliebige Listenlängen n wird man nach dem Divide-and-Conquer Paradigma das Problem in zwei möglichst gleich grosse Teilprobleme zerlegen und diese rekursiv weiterverarbeiten. Ist n gerade, so hat man zwei Teile der Grösse $\frac{n}{2}$. Ist n ungerade, so wird man ein Problem der Grösse $\lceil \frac{n}{2} \rceil$ und ein Problem der Grösse $\lfloor \frac{n}{2} \rfloor$ verwenden. Dies führt auf eine Rekursionsformel für den Aufwand $K_1(n)$ von Vergleichsoperationen, der durch

$$K_1(n) = \begin{cases} 0 & \text{für } n = 1, \\ 1 & \text{für } n = 2, \\ K_1(\lceil \frac{n}{2} \rceil) + K_1(\lfloor \frac{n}{2} \rfloor) + 2 & \text{für } n > 2 \end{cases}$$

gegeben ist.

- i. Berechnen Sie die ersten Werte von $K_1(n)$ und vergleichen Sie diese mit den entsprechenden Werten von $K_{optimal}(n)$. Wo tritt zum ersten Mal ein Unterschied auf? Können Sie sich erklären, warum für diese mit n_0 bezeichnete Zahl $K_{optimal}(n_0) < K_1(n_0)$ gilt?
 - Moral: "Naives" Divide-and-Conquer ist nicht unbedingt optimal! •
- ii. Experimentieren Sie mit folgender Strategie: ist $n = 2^k + m$ mit $0 \leq m < 2^k$, so teile man ein Problem der Grösse n in ein Problem der Grösse 2^k (welches nach b) behandelt wird) und eines der Grösse m auf. Untersuchen Sie also

$$K_2(n) := K_2(2^k) + K_2(m) + 2 \quad , \quad K_2(1) = 0$$

Ist $K_2(n) = K_{optimal}(n)$?