Aufgabe 1 (20 Punkte)

Betrachtet wird folgende Aufgabenstellung:

Gegeben: eine nichtleere Datenstruktur D, welche natürliche Zahlen enthält.

Gesucht: die kleinste Zahl $k1$ und die zweitkleinste Zahl $k2$ aus D.

Hinweis: Enthält D z. B. die Zahlen $5, 3, 4, 0, 6, 0, 3$, so ist $k1 = 0, k2 = 3$.

a) In welchen Situationen existiert $k2$ nicht? Ergänzen Sie den „Gesucht"-Teil der obigen Spezifikation so, dass sie vollständig ist.

b) Geben Sie jeweils einen Algorithmus an, welcher die Aufgabe löst, wenn

   - D ein Stack ist,
   - D ein Heap ist.

   Die Algorithmen sollen außer D keine weitere Datenstruktur verwenden. D darf nach Beendigung des Algorithmus leer sein. Standardfunktionen für das jeweilige D wie `empty`, `pop` oder `Maximum entfernen` usw. brauchen Sie nicht weiter zu detaillieren.

Aufgabe 2 (20 Punkte)

a) Führen Sie Quicksort mit der Folge 7, 1, 5, 3, 6, 2, 4 durch. Geben Sie dabei jede neue Belegung des Feldes nach einer Vertauschung an.

b) Wie viele Datensatzvertauschungen werden für folgende Schlüsselfolgen bei Verwendung von Auswahlsort und Bubblesort jeweils benötigt? (exakt, nicht $\mathcal{O}$)

   - $N, 1, 2, 3, \ldots, N-1$
   - $N, N-1, \ldots, 3, 2, 1$

c) Formulieren Sie die beiliegende C-Funktion `Bubblesort` so um, dass die Folge `a` von `n` Datensätzen indirekt sortiert wird. Dazu wird ein weiterer Parameter `int pi[]` übergeben, der nach Abschluss von `Bubblesort` die Reihenfolge der Datensätze enthält, in der sie sortiert sind, d.h. es gilt dann

$$\texttt{a}[\texttt{pi}[i]] \leq \texttt{a}[\texttt{pi}[j]] \quad \text{für } i < j.$$

```
void Bubblesort (datensatz a[], int pi[], int n)
{

}
```

Aufgabe 3 (20 Punkte)

Gegeben ist eine Folge von *ganzen* Zahlen $a_{\text{anfang}}, a_{\text{anfang}+1}, \ldots, a_{\text{ende}}$ mit der Eigenschaft

$$a_{\text{anfang}} < a_{\text{anfang}+1} < \ldots < a_{\text{ende}} \quad (\text{ende} \geq \text{anfang}).$$

Gesucht ist ein Index $i \in \{\text{anfang}, \ldots, \text{ende}\}$ mit der Eigenschaft $a_i = i$.

a) Begründen Sie kurz, dass die folgenden Aussagen richtig sind:

   1. Gilt $a_j > j$ für ein j, so liegt der gesuchte Index nicht im Bereich $\{j, \ldots, \text{ende}\}$.
   2. Gilt $a_j < j$ für ein j, so liegt der gesuchte Index nicht im Bereich $\{\text{anfang}, \ldots, j\}$.

b) Entwerfen Sie einen Algorithmus, der ein $i \in \{\text{anfang}, \ldots, \text{ende}\}$ mit der Eigenschaft $a_i = i$ ermittelt oder mitteilt, dass es einen solchen Index nicht gibt. Die Laufzeit Ihres Algorithmus soll $\mathcal{O}(\log n)$ sein.

c) Zu welchem Algorithmen-Grundtyp zählt Ihr Verfahren?

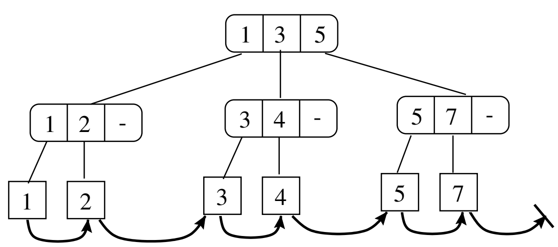
d) Arbeitet Ihr Algorithmus auch dann noch korrekt, wenn für die Eingabefolge

$$a_{\text{anfang}} \leq a_{\text{anfang}+1} \leq \ldots \leq a_{\text{ende}} \quad (\text{ende} \geq \text{anfang})$$

   gilt? Begründen Sie Ihre Antwort.

Aufgabe 4 (20 Punkte)

Das Durchlaufen eines B-Baumes erfordert Aufwand $\mathcal{O}(n \log n)$. Dies lässt sich auf $\mathcal{O}(n)$ reduzieren, wenn die Blätter zusätzlich durch Zeiger verkettet werden.



Hier soll diese zusätzliche Information nicht direkt beim Einfügen oder Löschen aktualisiert, sondern nachträglich erzeugt werden. Ergänzen Sie hierzu im beiliegenden Programm die Funktion `nachf_erzeugen1` (Zeile 533).

Hinweise:

- Durchlaufen Sie die Blätter (und damit den ganzen Baum) nach absteigenden Schlüsseln.
- Das (auszugsweise) abgedruckte Programm entspricht dem aus der Vorlesung, bis auf die Ergänzung in der Datenstruktur der Blätter (Zeilen 30 – 34). Außerdem ist die Funktion zur Ausgabe der Blätter, die die neue Verzeigerung verwendet, bereits eingefügt (Zeilen 549 – 568).

Aufgabe 5 (20 Punkte)

Gegeben ist ein Graph $G = (V, E)$. Zu $v \in V$ bezeichne $d_{in}(v)$ den Eingangsgrad des Knotens v.

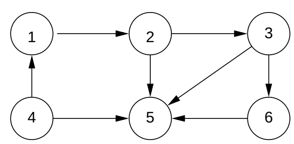
a) Schreiben Sie zu den beigefügten Datentypen `DCEL` und `AdjMatrix` eine `C`-Funktion

`void DCEL2Adj(DCEL *d, AdjMatrix *a)`,

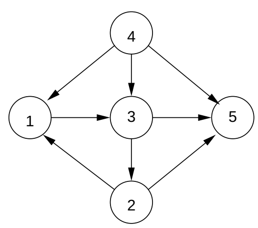
welche einen gerichteten Graphen in `DCEL`-Darstellung in eine Adjazenzmatrix vom Typ `AdjMatrix` überführt.

b) Begründen Sie, warum bei einer `DCEL`-Darstellung ein Aufwand $\Theta(d_{in}(j))$ nötig ist, um alle bei einem Knoten j endenden Kanten zu finden. Wie groß ist der Aufwand bei einer Adjazenzmatrix?

Gegeben sind die folgenden Graphen
G_1:



G_2:



c) Geben Sie für beide Graphen an, ob der jeweilige Graph topologisch sortierbar ist. Wenn nein, geben Sie eine Begründung an. Wenn ja, sortieren Sie den Graphen topologisch und skizzieren Sie ihn, indem Sie die Knoten in der topologischen Sortierung waagerecht anordnen.

d) Ist eine topologische Sortierung eindeutig?