Fakultät IV - Elektrotechnik und Informatik
Einführung in die Programmierung WS 2015/16
Feldmann / Semmler / Lichtblau / Streibelt / Pujol / Rost

Aufgabenblatt 9

letzte Aktualisierung: 13. Januar, 14:39 Uhr

(a6dcfcd2ad5b33f1f1685981e7f7f038cef569fd)

Ausgabe: *Mittwoch, 13.01.2016*
Abgabe: *spätestens Freitag, 22.01.2016, 20:00*

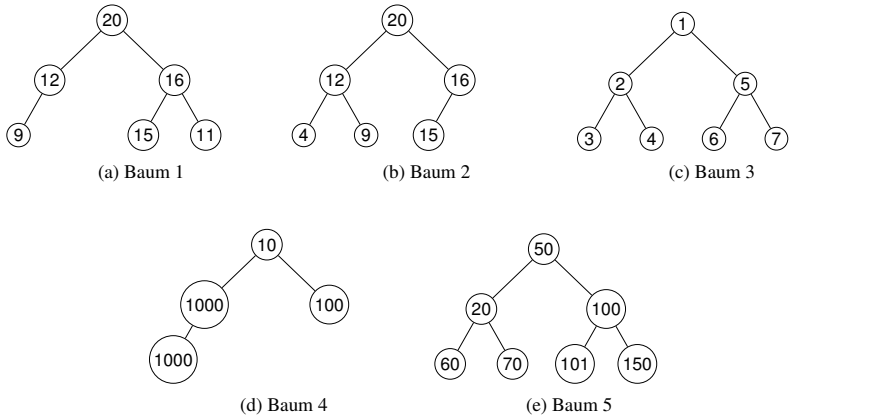
Thema: *Heap Sort*

Abgabemodalitäten

- Alle abzugebenden Quelltexte müssen ohne Warnungen und Fehler auf den Rechnern des tubIT/IRB mittels `gcc -std=c99 -Wall` kompilieren.
- Abgaben erfolgen prinzipiell immer in Gruppen à 2 Personen, welche in den Tutorien festgelegt wurden. Einzelabgaben sind explizit als solche gekennzeichnet.
- Die Abgabe erfolgt ausschließlich über SVN. Die finale Abgabe
  - für Gruppenabgaben erfolgt im Unterordner `Tutorien/t<xx>/Gruppen/g<xx>/Abgaben/Blatt<xx>/`
  - für Einzelabgaben erfolgt im Unterordner `Tutorien/t<xx>/Studierende/<tuBIT-Login>/Abgaben/Blatt<xx>/`

  wobei die Ordner von uns erstellt werden.
- Benutze für alle Abgaben von Programmcode das folgende Namensschema: `introprog_blatt0X_aufgabe0Y_Z.c`, wobei X durch die Blattnummer, Y durch die Aufgabe und Z durch die Unteraufgabe zu ersetzen ist.
  Beispiel: Aufgabe 1.2 wird zu: `introprog_blatt01_aufgabe01_2.c`
  Gib für jede Unteraufgabe maximal eine Quellcodedatei ab, es sei denn, die Aufgabenstellung erfordert explizit die Abgabe mehrerer Dateien pro Aufgabe.

  Benenne alle anderen Abgaben (Pseudocode, Textaufgaben) wie oben beschrieben. Die zugelassenen Abgabeformate sind PDF, ODT und Text (txt). Verwende auch hier eine Datei pro Aufgabe, nicht jedoch pro Unteraufgabe.



(a) Baum 1 (b) Baum 2 (c) Baum 3

(d) Baum 4 (e) Baum 5

Abbildung 1: Heaps oder nicht?

1. Aufgabe: Binäre Heaps (1 Punkt)

1.1. Heap-Kriterien (0,5 Punkte) Betrachte die Bäume in der Abbildung 1 und gebe jeweils an, ob es sich um einen Max-Heap, einen Min-Heap oder gar keinen Heap handelt.

Gebe für zwei der gültigen Heaps die Anordnung der Daten in den Array an, mit derer Hilfe Heaps gespeichert werden.

Gebe für die Bäume die keinen Heap darstellen an, warum diese kein Heap sind.

Gebe Deine Lösung der Teilaufgabe in einem Textdokument mit einem der folgenden Namen ab:
`introprog_blatt09_aufgabe01_1.{txt|odt|pdf}`

1.2. Heap erstellen (0,5 Punkte) Wende die in der Vorlesung vorgestellt Funktion `build_heap` auf die angegebenen Arrays an und überführe diese in einen Max- bzw. Min-Heap.

Zeichne die resultierenden Heaps. Du musst keine Zwischenergebnisse abgeben.

a) Max-Heap: `[8,1,2,5,3,1]`
b) Min-Heap: `[42,23,15,16,8,4]`
c) Min-Heap: `[10,100,1,10,100,10]`

Gebe Deine Lösung der Teilaufgabe in einem Textdokument mit einem der folgenden Namen ab:
`introprog_blatt09_aufgabe01_2.{txt|odt|pdf}`

2. Aufgabe: Implementierung eines binären Max-Heaps (2 Punkte)

Eine Fluggesellschaft möchte das Einsteigen der Passagiere in ihre Flugzeuge beschleunigen. (Es wird angenommen, dass die Flugzeuge nur über den Eingang hinter dem Cockpit betreten werden.) Dazu sollen möglichst immer die Sitzreihen, welche sich am weitesten hinten im Flugzeug befinden, zuerst besetzt werden. (Die Sitzreihen werden vom Cockpit an hochgezählt.) Sobald ein Passagier am Gate eintrifft, wird er mit seiner Sitzreihe registriert. Neue Sitzreihen werden jeweils erst aufgerufen, wenn im Kabinengang wieder Platz vorhanden ist. Für diese Aufgabe soll ein System entwickelt werden.

Das System soll nach dem folgenden Muster funktionieren:

Eingabe <Zahl> Wenn ein Passagier beim Gate erscheint, wird dessen Sitzreihe (eine positive ganze Zahl) in die Warteliste eingetragen.

Eingabe 'n' Falls im Kabinengang des Flugzeugs wieder ausreichend Platz ist, soll das Bodenpersonal die Sitzreihe mit der größten Nummer angezeigt bekommen, für die ein Passagier am Gate wartet. Eine leere Warteliste soll durch die Ausgabe von "Leer" angezeigt werden.

Eingabe 'q' Das Programm soll sich durch Eingabe von "q" (für "quit") beenden lassen.

Andere Eingaben Auf alle anderen Eingaben sollen eine Fehlernachricht angezeigt werden.

Um die Möglichkeit von Betriebsfehlern zu reduzieren, müssen die folgenden Bedingungen erfüllt sein:

- Die Implementierung verwendet einen binären Max-Heap. Der Heap ist in `main_blatt09.c` als `heap` vorgegeben.
- Die Implementierung basiert auf dem Pseudocode der Vorlesung.
- Eventueller dynamischer Speicher muss freigegeben werden.

Ein beispielhafter Aufruf des Programmes ist in Listing 1 dargestellt.

Listing 1: Programmbeispiel

```
> gcc -std=c99 -Wall introprog_blatt09_aufgabe02.c main_blatt09.c \
                     -o introprog_blatt09_aufgabe02
> ./introprog_blatt09_aufgabe02
[...]
> n
Leer!
> 12
> 33
> 3
> n
33
> n
12
> n
3
> n
Leer!
> foo
Fehler: Eingabe nicht erkannt!
> q
```


Die Codevorgabe im SVN besteht aus drei Dateien: `blatt09.h`, `main_blatt09.c` und `introprog_blatt09_aufgabe02_vorgabe.c` (siehe Listing 2). Die letzte Datei soll angepasst werden und als `introprog_blatt09_aufgabe02.c` eingereicht werden.

Folgende Funktionen müssen implementiert werden:

2.1. Eingabe erkennen (0,5 Punkte)

Die Funktion `read_user_input` soll jeweils eine Zeile von der Standardeingabe (`stdin`) einlesen und abhängig von der folgenden Fallunterscheidung einen Integer zurückgeben:

Positive Zahl Gebe die eingelesene Zahl zurück.

'n' Gebe -1 zurück.

'q' Gebe -2 zurück.

Rest Gebe -3 zurück, wenn keiner der anderen Fälle zutrifft.

2.2. Funktion heapify (0,5 Punkte)

Schreibe die Funktion `void heapify(heap* h, int i)`, die sicherstellt, dass der Knoten i und seine Kinderknoten die Heapeigenschaft erfüllen, falls die Unterbäume der Kinderknoten die Heapeigenschaft erfüllen.

2.3. Funktion heap_insert (0,5 Punkte)

Schreibe die Funktion `int heap_insert(heap* h, int val)`, die ein neues Element in den Heap einfügt. Achte dabei darauf, dass die Datenstruktur auch nach dem Einfügen ein gültiger Heap ist. Orientiere dich an dem Pseudocode der Vorlesung.

Sollte der Funktion ein Heap übergeben werden, der schon vollständig aufgefüllt worden ist (d.h. ein Heap mit genau `MAX_HEAP_SIZE` Elementen), soll -1 zurückgeben werden.

2.4. Funktion heap_extract_max (0,5 Punkte)

Schreibe die Funktion `int heap_extract_max(heap* h)`, die das größte Element des Heaps zurückgibt und gleichzeitig vom Heap entfernt. Achte dabei darauf, dass auch nach dem Entfernen des Elements der Heap die Heapeigenschaft erfüllt. Orientiere dich an dem Pseudocode der Vorlesung.

Sollte der Funktion einen Heap übergeben werden, der leer ist (d.h. ein Heap mit 0 Elementen), soll -1 zurückgeben werden.

Listing 2: Codevorgabe `introprog_blatt09_aufgabe02_vorgabe.c`

```c
#include "blatt09.h"

/* Stellt die Heap Bedingung wieder her, wenn der linke und rechte Unterbaum
   ↪ die Heap
 * Bedingung schon erfüllen.
 */
void heapify(heap* h, int i) {
    // HIER implementieren
}

/* Holt einen Wert aus dem Heap
 *
```

```
 * Rückgabewert: Kleinster Wert der Heap
 */
int heap_extract_max(heap* h) {
    // HIER implementieren
}

/* Fügt einen Wert in den Heap ein
 *
 * Gibt einen Fehler auf stderr aus und beendet das Programm mit EXIT Code 1,
 * wenn der Heap voll ist.
 */
void heap_insert(heap* h, int key) {
    // HIER implementieren
}

/* Lese die Eingabe von STDIN. Lese jeweils nur eine Zeile ein.
 * Gebe folgende Werte zurück
 *
 * Eine positive Zahl, wenn eine solche eingegeben wurde (z.b. "10").
 * -1 Bei der Eingabe von 'n'
 * -2 Bei der Eingabe von 'q'
 * -3 Wenn die Eingabe sich nicht eindeutig zuordnen ließ.
 *
 */
int read_user_input() {
    // HIER implementieren
}
```