Fakultät IV - Elektrotechnik und Informatik
Einführung in die Programmierung WS 2014/15
Feldmann / Rost / Streibelt / Lichtblau / Deimel / Lucas

Aufgabenblatt 10

letzte Aktualisierung: 21. Januar, 20:55 Uhr

(fa662a02e6a6333fd1155871c1b876ae91824cfe)

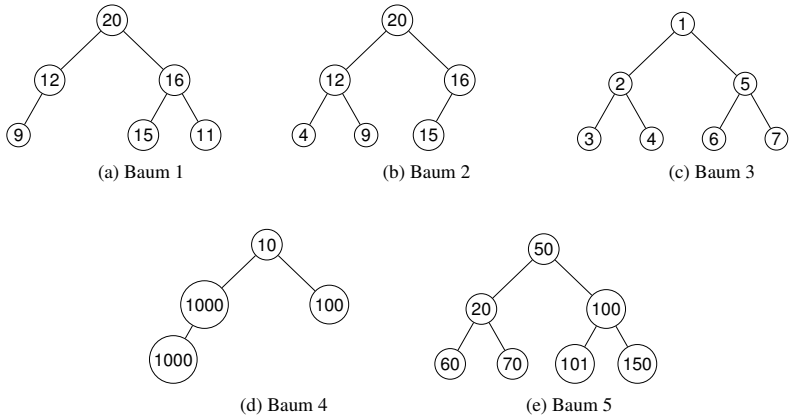
Ausgabe: Mittwoch, 21.01.2015
Abgabe: spätestens Freitag, 30.01.2015, 18:00

Thema: *Implementierung Binäre Heaps*

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>/Blatt<xx>/submission/`
  - für Einzelabgaben erfolgt im Unterordner `Tutorien/t<xx>/Studierende/<tuBIT-Login>/Blatt<xx>/submission/`
- Benutzen Sie 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 ersetzen ist.
  Beispiel: Aufgabe 1.2 wird zu: `introprog_blatt01_aufgabe01_2.c`
  Für jede Unteraufgabe geben Sie maximal eine Quellcodedatei ab, es sei denn, die Aufgabenstellung erfordert explizit die Abgabe mehrerer Dateien pro Aufgabe.

  Alle anderen Abgaben (Pseudocode, Textaufgaben) benennen Sie wie oben beschrieben. Die zugelassenen Abgabeformate sind PDF, ODT und Text (txt). Auch hier verwenden Sie 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 (10 Punkte)

1.1. Heap-Kriterien (7 Punkte) Betrachten Sie die Bäume in der Abbildung 1 und geben Sie jeweils an, ob es sich um einen Max-Heap, einen Min-Heap oder gar keinen Heap handelt.

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

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

Geben Sie Ihre Lösung der Teilaufgabe in einem Textdokument mit einem der folgenden Namen ab: `introprog_blatt10_aufgabe01_1.{txt|odt|pdf}`

1.2. Heap erstellen(3 Punkte) Wenden Sie die in der Vorlesung vorgestellt Funktion `build_heap` auf die angegebenen Arrays an und überführen Sie diese in einen Max- bzw. Min-Heap.

Zeichnen Sie die resultierenden Heaps. Sie müssen 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]`

Geben Sie Ihre Lösung der Teilaufgabe in einem Textdokument mit einem der folgenden Namen ab: `introprog_blatt10_aufgabe01_2.{txt|odt|pdf}`

2. Aufgabe: Implementierung eines binären Min-Heaps (25 Punkte)

Eine Fluggesellschaft möchte das Einsteigen der Passagiere in ihre Flugzeuge beschleunigen. Dazu sollen möglichst immer die Passagiere, die am weitesten vorne im Flugzeug sitzen, zuerst in das Flugzeug gelassen werden. Dazu werden die am Gate ankommenden Passagiere mit ihrer Boardkarte registriert. Anschließend werden die Sitzreihen aufgerufen, die als nächstes einsteigen sollen. Allerdings sind zu Beginn des Boardings noch nicht alle Passagiere am Gate, sondern treffen nach und nach während des Boardings ein.

Um diesen Prozeß zu bewerkstelligen, verwendet das Boarding System eine Warteliste, die die Sitzreihen der derzeit wartenden Passagiere ordnet. Wenn ein Passagier beim Gate erscheint, wird dessen Sitzreihe in die Warteliste eingetragen. Falls im Kabinengang des Flugzeugs wieder ausreichend Platz ist, soll dem Bodenpersonal angezeigt werden, welche Sitzreihe als nächstes einsteigen soll. Ist die Warteliste leer, soll dies ebenfalls angezeigt werden.
Schreiben Sie ein Programm um diesen Prozeß zu automatisieren. Das Programm soll seine Befehle zeilenweise von der Standardeingabe empfangen und Antworten zeilenweise auf der Standardausgabe ausgeben. Wenn ein Passagier neu zum Boarding erscheint, wird seine Sitzreihe als Zahl eingegeben. Mittels "f" wird gemeldet, dass ein weiterer Passagier einsteigen kann. Daraufhin soll das Programm die Sitzreihe des nächsten Passagiers ausgeben und diesen aus der Warteliste austragen Ist kein weiterer Passagier in der Warteliste vermerkt, soll das Programm "Leer" zurückmelden. Außerdem soll das Programm mittels "q" jederzeit beendet werden können.
Verwenden Sie zur Implementierung der Warteliste einen Heap!

Benutzen Sie unsere Codevorgabe zur Implementierung (siehe Listing 1).

Hinweis: Ihr Programm soll die auszuführenden Operationen zeilenweise vom Standardinput einlesen. Verwenden sie dazu die vorgegebene Funktion!

Geben Sie den Quelltext in einer Datei mit folgendem Namen ab:
`introprog_blatt10_aufgabe02.c`

2.1. Heap-Struktur (3 Punkte)

Schreiben Sie eine passende Struktur **`struct`** `heap` um alle nötigen Informationen, über einen binären Heap zu speichern.

Schreiben Sie zusätzlich die Funktion `heap* heap_init()` um den Heap zu initialisieren.

2.2. Funktion heapify() (6 Punkte)

Schreiben Sie 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() (8 Punkte)

Schreiben Sie die Funktion **`void`** `heap_insert(heap* h, int val)`, die einen neuen Element in den Heap einfügt. Achten Sie dabei darauf, dass die Datenstruktur auch nach dem Einfügen ein gültiger Heap ist.

Stellen Sie mit Hilfe von `assert()` sicher, dass die maximale Größe des Heaps nicht überschritten wird.

2.4. Funktion heap_extract_min() (8 Punkte)

Schreiben Sie die Funktion **`int`** `heap_extract_min(heap* h)`, die das kleinste Element des Heaps zurückgibt und gleichzeitig vom Heap entfernt. Achten Sie dabei darauf, dass auch nach dem Entfernen des Elements der Heap gültig bleibt.

Stellen Sie mit Hilfe von `assert()` sicher, dass der Heap bei Aufruf der Funktion mindestens ein Element enthält.

Listing 1: Codevorgabe `introprog_blatt09_aufgabe02_vorgabe.c`

```c
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>

#define MAX_HEAP_SIZE 400

typedef struct heap heap;

/* Dieses struct definiert einen Heap

   Der Heap kann maximal MAX_HEAP_SIZE Elemente enthalten
*/
struct heap {
    int array[MAX_HEAP_SIZE];
    int size;
};

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

/* Holt einen Wert aus dem Heap.
 *
 * Gibt einen Assert Fehler zurück wenn die Heap leer ist.
 *
 * Rückgabewert: Kleinster Wert der Heap
 */
int heap_extract_min(heap* h) {
    // Hier bitte Ihr Code
}

/* Fügt einen Wert in den Heap ein.
 *
 * Gibt einen Assert Fehler zurück wenn die bereits Heap voll ist.
 */
void heap_insert(heap* h, int key) {
    // Hier bitte Ihr Code
}

/* Diese Funktion initialisiert die Heap-Struktur
 *
 * Rückgabewert: Pointer auf den Heap
 */
heap* heap_init() {
    // Hier bitte Ihr Code
}

/* Programm zum Führen einer Warteliste
 *
```

```
 * Das Programm nimmt Befehle zeilenweise auf der Standardeingabe entgegen:
 * "q": Beende das Programm.
 * "f": Der nächste Passagier soll einsteigen. Das Programm gibt als Antwort die
 *      vorzugebenede Sitzreihe aus.
 * positive Ganzzahl: Ein Passagier mit dieser Sitzreihennummer wurde registriert.
 */
int main(int argc, char** argv)
{
    heap* warteliste = heap_init();
    while (1)
    {
        char string[15];
        scanf("%s",string);
        // Gehe aus dem Programm raus
        if (string[0] == 'q') {break;}
        // Hole die niedrigste Sitzreihennummer
        if (string[0] == 'f')
        {
            if (warteliste->size > 0) {
                printf("%i\n", heap_extract_min(warteliste));
            } else {
                printf("Leer!\n");
            }
            continue;
        }
        // Füge eine postive Zahl als Sitzreihennummer ein
        int input_as_integer = atoi(string);
        if (input_as_integer > 0) {
            heap_insert(warteliste, input_as_integer);
            continue;
        }
        // Bei fehlerhafter/unbekannte Eingabe
        printf("Fehler: Eingabe nicht erkannt!\n");
    }
}
```