Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Markus Püschel
Peter Widmayer
Thomas Tschager
Tobias Pröger
Tomáš Gavenčiak

Department Informatik
11. Januar 2017

Algorithmen & Datenstrukturen — Probeprüfung — HS 16

Das ist die erste Aufgabe der Probeprüfung. Die Probeprüfung hat keinen Einfluss auf die Endnote. Die Schlussprüfung wird umfangreicher sein und aus zwei Programmieraufgaben (und Theorieaufgaben) bestehen. Die zweite Aufgabe der Probeprüfung finden Sie als PDF im Ordner `documentation` *und nach der Probeprüfung online. Wir empfehlen, diese zweite Aufgabe zu Hause zu lösen (am besten mit einer Stoppuhr).*

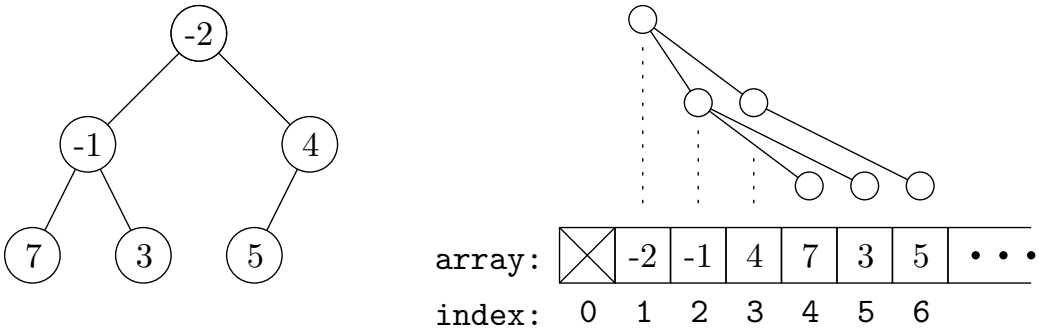
Implementierung eines Min-Heaps

Ihre Aufgabe besteht darin, eine Implementierung eines binären *Min-Heaps*, der im Anfangsstück eines Arrays gespeichert ist, zu vervollständigen. Die folgenden Operationen sollen unterstützt werden:

- `Insert`(v) fügt das Element v in den Heap ein und stellt die Heap-Eigenschaft wieder her.
- `ExtractMin` liefert und entfernt das Minimum aus dem Heap und stellt die Heap-Eigenschaft wieder her.
- `QueryLast` liefert das Element an der letzten Position des Anfangsstückes des Arrays, in dem der Heap gespeichert ist.

Wenn beispielsweise die Zahlen $4, 7, -1, 3, -2, 5$ in dieser Reihenfolge in einen anfangs leeren Heap eingefügt werden, erhalten wir folgenden Heap, sowie die folgende Repräsentation als Array. In der Abbildung ist die Zuordnung der Knoten des Heaps zu den Elementen im Array gekennzeichnet.

Das Heap-Array in der gegebenen Vorlage verwendet Indizes beginnend mit 1, um die Berechnungen der Indizes zu vereinfachen. Das Element an Position 0 wird nicht verwendet und der Heap füllt nicht notwendigerweise das gesamte Array. Im untenstehenden Beispiel liefert die Operation `QueryLast` als Ergebnis 5.



Für n gegebene Ganzzahlen $v_1, v_2, \ldots, v_n$ soll Ihr Programm ausgehend von einem leeren Heap die Operationen `Insert(`v_i`)`, jeweils gefolgt von einer `QueryLast` Operation, für alle $i = 1, \ldots, n$ ausführen und den Rückgabewert jeder `QueryLast` Operation ausgeben. Anschliessend soll das Programm die Operation `ExtractMin` n mal ausführen und jeweils den Rückgabewert ausgeben.

Die gesamte Abfolge von Operationen ist also: `Insert(`v_1`), QueryLast, Insert(`v_2`), QueryLast, ..., Insert(`v_n`), QueryLast, ExtractMin, ..., ExtractMin` bis der Heap leer ist.

Der Grossteil des Codes steht bereits in der Vorlage – Ihre Aufgabe ist es, die fehlenden Teile der Operationen `Insert` und `ExtractMin` zu vervollständigen.

Eingabe Die erste Zeile der Eingabe enthält lediglich die Anzahl der Tests.

Jeder Test ist als eine einzige Zeile gegeben: Diese besteht aus einer Ganzzahl $1 \leq n \leq 1000$, der Anzahl der Einfüge-Operationen, gefolgt von n Ganzzahlen $v_1, v_2, \ldots, v_n$, jeweils mit $-1000 \leq v_i \leq 1000$, die eingefügt werden sollen. Beachten Sie, dass Zahlen mehrmals vorkommen können und dass mehrere identische Zahlen im Heap erhalten bleiben müssen.

Ausgabe Für jeden Test sollen zwei Zeilen ausgegeben werden: Die erste Zeile enthält die Ausgabe der n `QueryLast` Operationen (durch Leerzeichen getrennt). Die zweite Zeile enthält die Ausgabe der n `ExtractMin` Operationen (durch Leerzeichen getrennt).

Beispiel

Eingabe (der erste Test entspricht dem Beispiel oben):

```
2
6 4 7 -1 3 -2 5
5 5 7 3 4 2
```

Ausgabe:

```
4 7 4 7 3 5
-2 -1 3 4 5 7
5 7 5 7 4
2 3 4 5 7
```

Punkte Sie können bis zu 100 Punkte am Judge bekommen. Um alle Punkte zu bekommen, sollte Ihre Lösung jede Operation mit Laufzeit in $\mathcal{O}(\log n)$ implementieren.

Senden Sie Ihr `Main.java` unter folgendem Link ein: `https://judge.inf.ethz.ch/team/websubmit.php?cid=18986&problem=DNA14_5`. Das Passwort für die Einschreibung ist "`testitwell`". Der Judge wird mindestens bis zur Schlussprüfung Lösungen für diese Aufgabe annehmen. Sie können auch nach der Probeprüfung noch Lösungen einreichen.

Hinweise Wir stellen für diese Aufgabe eine Programmvorlage als Eclipse-Projektarchiv zur Verfügung. Die Vorlage implementiert bereits den Grossteil der Funktionen. Ihre Aufgabe besteht darin, den Code der Funktionen `MinHeap.insert()` und `MinHeap.extractMin()` zu vervollständigen.

Wie üblich enthält das Archiv Testdaten, damit Sie lokal testen können. Ausserdem stellen wir ein `Judge.java` Programm zur Verfügung, das Ihr Programm `Main.java` mit allen verfügbaren Testdaten testet – öffnen und starten sie dazu einfach `Judge.java`. Die zur Verfügung gestellten Testdaten sind nicht die Testdaten, welche der Online-Judge verwendet.

In dieser Prüfung dürfen Sie *keine* Java Libraries ausser `java.util.Scanner` einbinden. Dies wird erst nach der Prüfung getestet und bestraft, falls Ihr Code andere Bibliotheken `import`iert.