Übungen 8

Patrick Bucher

13.04.2017

Inhaltsverzeichnis

1 Grundlegende Zusammenhänge

a)

Man könnte die Dreiecke nach Flächeninhalt oder Umfang sortieren. Ich schlage den Flächeninhalt als natürliche Ordnung vor.

b)

Die Klasse `Triangle` implementiert das Interface `Comparable` und somit die Methode `compareTo(Triangle o)`. Der Parameter ist ein Dreieck, das für den Vergleich hinzugezogen werden soll. Der Rückgabewert ist eine Zahl, für die gilt:

- o: das aktuelle Dreieck ist *grösser* als der Parameter
- = o: beide Dreiecke sind gleich gross
- < o: das aktuelle Dreieck ist *kleiner* als der Parameter

c)

`hashCode()` sollte auch mit `equals()` korrespondieren. Gleiche Objekte *müssen* den gleichen `hashCode` haben. Unterschiedliche Objekte *sollten* einen unterschiedlichen `hashCode` haben. `equals()` und `hashCode()` überschreibt man von der Klasse `Object`.

d)

Eine spezielle Ordnung kann mit einem `Comparator` und der Methode `compareTo(Triangle a, Triangle b)` implementiert werden. Dazu braucht es kein UML-Klassendiagramm.

e)

- `Arrays.sort(Object[])` für die natürliche Ordnung
- `Arrays.sort(Object[], Comparator<? super T>)` für die spezielle Ordnung

Aus der JavaDoc:

`Implementation note: This implementation is a stable, adaptive, iterative mergesort...`

`sort()` arbeitet *stabil*.

f)

Welches UML-Klassendiagramm von d?

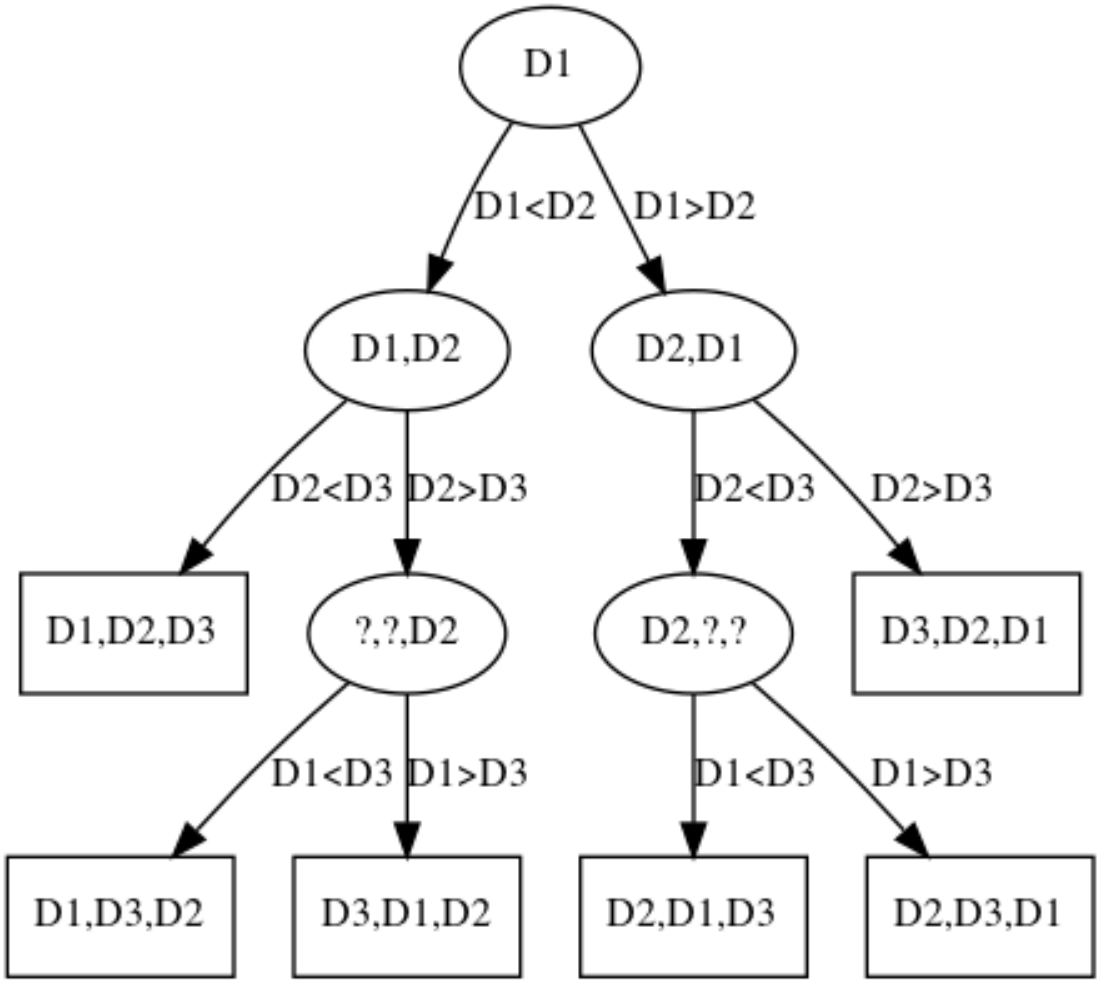

Abbildung 1: Entscheidungsbaum

2 Entscheidungsbaum

a)

3 Aus instabil mach stabil

TODO

4 Direktes Einfügen

TODO

5 Direktes Auswählen

TODO

6 Direktes Austauschen

TODO

7 Optional: Shellsort

TODO