technische universität dortmund

fakultät für informatik

Andre Droschinsky
Roman Kalkreuth
Denis Kurz
Bernd Zey

Dortmund, den 10. Januar 2019

Praktikum zur Vorlesung Einführung in die Programmierung WS 18/19

Blatt 10

Es können 28 Punkte erreicht werden.

Aufgabe 1: Binäre Suchbäume (14 Punkte)

In dieser Aufgabe sollen verschiedene Funktionen eines binären Suchbaums mit Hilfe von Templates und dynamischem Speicher für beliebige Elementtypen implementiert werden. Wie in der Vorlesung sollen die Elemente des Suchbaums durch einen Hilfs-Struct `Node` repräsentiert werden, welches ein Datum des Typs `T` und jeweils einen Zeiger auf die Wurzel des linken und des rechten Unterbaums hat. Der Suchbaum selbst besitzt lediglich einen Zeiger auf den Wurzelknoten. Zur Erinnerung: Ein binärer Suchbaum unterscheidet sich gegenüber einem einfachen binären Baum dadurch, dass alle Daten im linken Unterbaum kleiner und die Daten im rechten Unterbaum größer als das Datum eines Knotens sind. Fügen Sie dazu die auf der Website bereitgestellten Dateien zu einem neuen Projekt `Aufgabe_10_1` hinzu. Erweitern Sie zur Bearbeitung dieser Aufgabe die Header-Datei `bintree.h`.

a) Implementieren Sie die rekursive `BinTree`-Methode `int height(Node* node)`, welche die Höhe des Suchbaums zurückgibt. Die Zeit, die dafür benötigt wird, sollte proportional zur Anzahl der Knoten des Baums sein (*lineare Laufzeit*). Zur Erinnerung: Die Höhe eines Suchbaums ist die Länge des längsten Pfades von der Wurzel zu einem Blatt. Die Höhe eines leeren Suchbaumes ist also 0; wenn nur die Wurzel existiert, ist die Höhe 1 usw.

_____ (4)

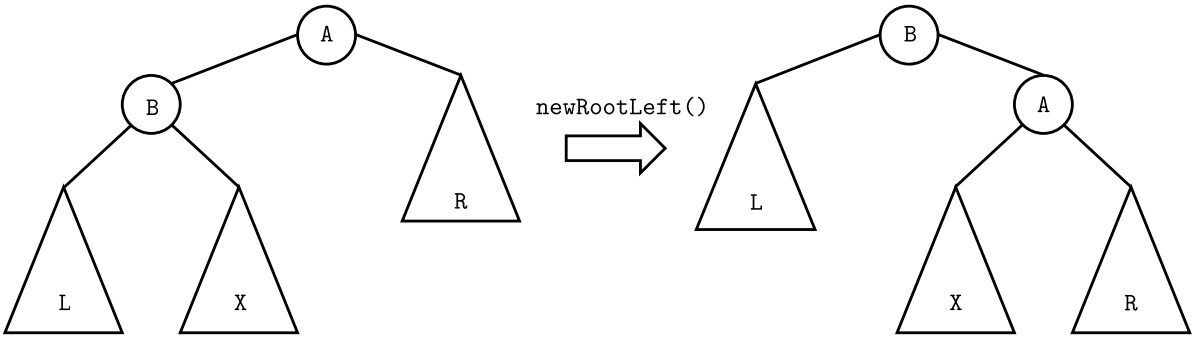
b) Schreiben Sie die `BinTree`-Methode `range`, die alle Werte des Binärbaums, welche in einem Intervall [min , max] liegen, sortiert ausgibt. Gehen Sie dazu folgendermaßen vor:
Durchlaufen Sie den Baum in In-Order-Reihenfolge. Rufen Sie allerdings – abweichend zur `inOrder`-Methode – die Methode `range` nur dann rekursiv für den linken Unterbaum auf, wenn der Wert des aktuellen Knotens größer als `min` ist, und nur für den rechten Unterbaum, wenn der Wert des aktuellen Knotens kleiner als `max` ist.

_____ (5)

c) Schreiben Sie die `BinTree`-Methode `void rotateLeft()`, welche die Wurzel des linken Unterbaums zur neuen Wurzel des Baums macht. Wenn der Baum oder der linke Unterbaum leer ist, soll der Baum unverändert bleiben. Ansonsten wird, um die Suchbaum-Eigenschaft zu erhalten, der Baum wie folgt modifiziert:

- Die Wurzel des linken Unterbaums wird die neue Wurzel; die alte Wurzel wird Wurzel des rechten Unterbaums.
- Der linke Unterbaum der neuen Wurzel bleibt unverändert.
- Die alte Wurzel mit ihrem rechten Unterbaum wird rechter Unterbaum der neuen Wurzel.
- Der rechte Unterbaum der neuen Wurzel wird linker Unterbaum der alten Wurzel.

Da in einem Suchbaum nicht nur Zahlen in den Knoten, sondern auch große Objekte in den Knoten stehen können, sollen hier die Zeiger zwischen den `Node`-Objekten manipuliert werden; es soll keine Zuweisungen an deren `data`-Felder geben. Orientieren Sie sich dabei an dem folgenden Schema:



HINWEIS: Dieses Verfahren ist Teil vom sogenannten *Balancieren* von Suchbäumen. Es kann genutzt werden, um die Tiefe und somit die benötigte Zeit mancher Operationen eines Suchbaums zu reduzieren.

EINSCHRÄNKUNG: Ihre Implementierung darf keinen dynamischen Speicher allokieren.

_____ (5)

Aufgabe 2: Poker (14 Punkte)

In dieser Aufgabe sollen verschiedene Pokerhände durch Klassen in einer gemeinsamen Hierarchie modelliert werden. Importieren Sie dazu die auf der Website bereitgestellten Quelltextdateien in den `src`-Ordner eines neuen Projektes `Aufgabe_10_2`.
Beim Poker haben alle Hände, also Kombinationen aus fünf Karten, bestimmte Namen. Eine Kombination aus fünf Karten derselben Farbe (z.B. Herz 4, Herz 7, Herz 8, Herz Bube, Herz Ass) wird etwa *Flush* genannt. Es sollen Klassen für einige solcher Hände geschrieben werden, die von der bereitgestellten Klasse `PokerResult` erben. Diese Klasse hat im Wesentlichen zwei Methoden:

- **`std::string name()`**: Gibt eine textuelle Repräsentation der Hand zur Ausgabe auf der Kommandozeile zurück.
- **`bool match(Hand const &cards)`**: Stellt fest, ob die übergebenenen Karten `cards` die Anforderungen an die entsprechende Hand erfüllen.

Die Klasse `Hand` kann so ähnlich wie ein Array benutzt werden: Mit `hand[2]` kann etwa auf die dritte Karte der Hand zugegriffen werden. Mehr Details sind zum Lösen der Aufgaben nicht notwendig, können aber im Zweifelsfall in den Dateien `hand.h` und `hand.cpp` nachgesehen werden. **Zur Vereinfachung** wird angenommen, dass schon die richtigen fünf Karten ausgesucht wurden, was z.B. die Erkennung eines Flush vereinfacht; eine Hand hat also nur fünf, keine sieben Karten.
Alle neuen Klassen sollen in der Datei `poker_result.h` definiert und in der Datei `poker_result.cpp` implementiert werden. Sobald eine der folgenden Klassen implementiert wurde, sollten die zwei zugehörigen Kommentare in der bereitgestellten `main`-Funktion entfernt werden (*auskommentieren*), um die Klasse beim Ausführen des Programms automatisch testen zu lassen.

a) Ergänzen Sie eine von `PokerResult` abgeleitete Klasse `Flush`. Überschreiben Sie deren Methode `name` so, dass die Zeichenkette `"Flush"` zurückgegeben wird.

_____ (2)

b) Überschreiben Sie in der Klasse `Flush` die Methode `match` so, dass `true` zurückgegeben wird, wenn bei den fünf als `Hand`-Objekt übergebenen Karten ein Flush vorliegt, und `false` sonst. Dazu müssen alle fünf Karten dieselbe Farbe (Attribut `suit`) haben.
Für diese Aufgabe ist unerheblich, ob eine höherwertige Hand vorliegt. Bei einem Royal Flush soll `Flush::match` also trotzdem `true` zurückgeben.

_____ (4)

c) Die Hände High Card, Paar und Drilling sind sich sehr ähnlich: Sie sind darüber definiert, dass n Karten mit demselben Wert vorliegen, mit $n = 1$ für die High Card, $n = 2$ für das Paar und $n = 3$ für den Drilling.
Ergänzen Sie eine von `PokerResult` abgeleitete Klasse `ManyOfAKind` mit einem konstanten, ganzzahligen Attribut `m_howMany`. Ergänzen Sie einen Konstruktor, mit dem dieses Attribut initialisiert werden kann. Überschreiben Sie die Methode `match` so, dass `true` zurückgegeben wird, wenn mindestens `m_howMany` viele Karten mit demselben Wert vorliegen, und `false` sonst. Sie sollten dabei ausnutzen, dass die `Card`-Objekte von der `Hand`-Klasse immer aufsteigend nach Ihrem Wert sortiert vorgehalten werden.
Da `ManyOfAKind` keine eigenständige Hand repräsentiert, sondern nur als Hilfsklasse für die nächste Teilaufgabe dient, sollte die Methode `name` rein virtuell bleiben.
Auch hier soll wieder nicht überprüft werden, ob eine höherwertige Hand vorliegt. Bei einem Full House (zwei Karten mit einem Wert, drei Karten mit einem anderen, z.B. 4, 4, 4, Bube, Bube) sollte `ManyOfAKind::match` also `true` zurückgeben, wenn `m_howMany` höchstens 3 ist.

_____ (5)

d) Leiten Sie die Klassen `HighCard`, `Pair` und `ThreeOfAKind` (entspricht dem Drilling) von `ManyOfAKind` ab. Überschreiben Sie deren Standardkonstruktoren so, dass sinnvoll ein Konstruktor der Oberklasse aufgerufen wird. Überschreiben Sie die Methode `name` so, dass jeweils `"High Card"`, `"Paar"` und `"Drilling"` zurückgegeben werden. Die Methode `match` soll nicht überschrieben werden, aber durch die Vererbung trotzdem korrekt funktionieren.

_____ (3)