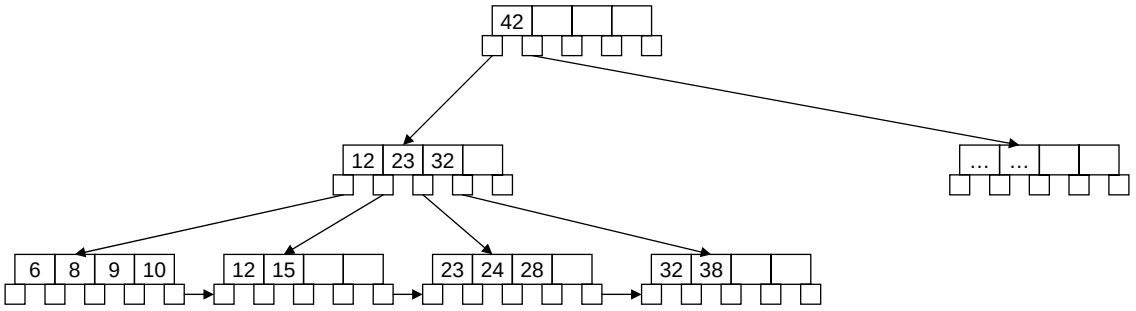
Aufgabenblatt 2: Indexstrukturen

- Abgabetermin: **Mittwoch, 23.05.07**
- Das Aufgabenblatt gilt als bestanden, wenn mindestens *22,5* der 45 Punkte erreicht werden.
- Die Aufgaben *sollen* in Zweiergruppen bearbeitet werden. Für Einzelabgaben ziehen wir 4 Punkte ab.
- Bitte verwendet für jede Aufgabe ein separates Blatt und beschriftet *jedes* Blatt der Abgabe mit Namen, Matrikelnummern und Seitenzahl!
- Abgabe: Auf Papier im Fach „Datenbanksysteme II“ im Foyer oder per E-Mail als pdf oder doc an dbs2@hpi.uni-potsdam.de mit cc an brigitte.hobro@hpi.uni-potsdam.de mit Subject „Abgabe DBS II: Aufgabenblatt n [Namen | Matrikelnummern]“

Aufgabe 1: Einfügen und Löschen in B^+-Bäumen

Gegeben sei der folgende B^+-Baum:

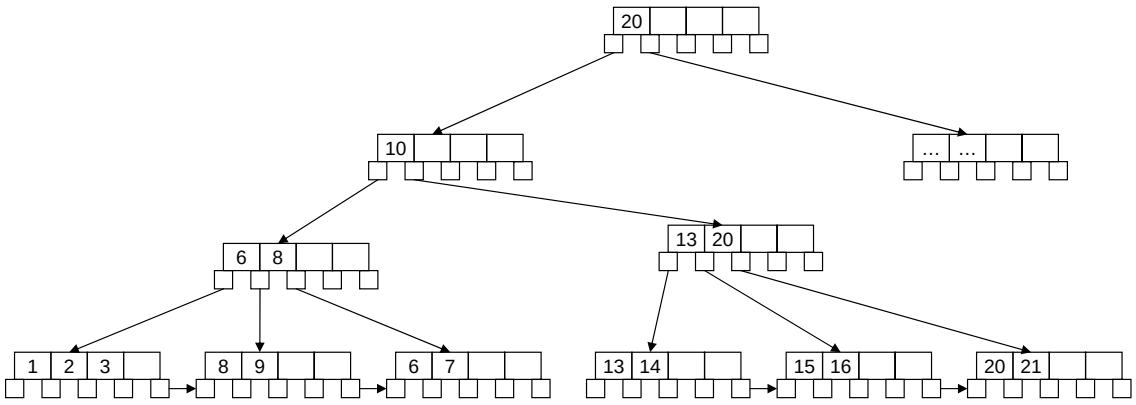


a) Füge die Schlüssel 17, 18, 19 und 11 (in dieser Reihenfolge) ein. Gib den B^+-Baum nach dem Einfügen der 19 und der 11 an. **6 P**

b) Lösche die Schlüssel 18 und 23 (in dieser Reihenfolge). Gib den B^+-Baum nach jeder Löschoperation an. **6 P**

Aufgabe 2: korrekte B^+-Bäume

Ist der folgende gegebene B^+-Baum ein gültiger B^+-Baum mit max. 4 Schlüsseln (und damit max. 5 Pointern) pro Knoten? Falls nein, nenne alle Verletzungen der B^+-Baum-Definition und begründe sie kurz. **10 P**



Aufgabe 3: Dichtbesetzte und dünnbesetzte Indexe

Betrachte Blöcke, die entweder 30 Datensätze oder 200 Schlüssel-Pointer-Paare aufnehmen können. Die Blöcke dürfen nur zu 80 % gefüllt werden. Wie viele Blöcke werden in Abhängigkeit der Anzahl der Datensätze n benötigt für

a) die Datensätze und einen dichtbesetzten Index? **2 P**

b) die Datensätze und einen dünnbesetzten Index? **1 P**

Aufgabe 4: Hashtable Index vs. Full Table Scan

Implementiere die folgenden beiden Datenstrukturen. Die Daten sollen zur Vereinfachung ausschließlich im Hauptspeicher gehalten werden. Es sollen Datensätze mit zwei Attributen gespeichert werden – einem Schlüssel („key“, integer) und einem zugehörigen Wert („value“, String). 32 Datensätze werden zu einem Block zusammengefasst.

- Variante 1: Organisiere die Daten mit Hilfe eines Hashtable Index mit 256 Buckets. Der vollständige Datensatz soll im entsprechenden Bucket (ein Block plus Overflow-Blöcke) gespeichert werden. Verwende als Hashfunktion h für den key k die Funktion $h(k) = k \bmod |\text{Buckets}|$.
- Variante 2: Die Datensätze werden in der Einfüge-Reihenfolge in eine Sequenz von Blöcken gespeichert.

Die zu speichernden Datensätze sind in der Datei BeispielDaten.txt (lehrveranstaltungen\DBSII_naumann) gegeben. Jede Zeile entspricht einem Datensatz, die Attribute sind durch ein Leerzeichen getrennt.

a) Lies die Datensätze ein und suche die Werte zu den folgenden Schlüsseln: 371018, 395689, 668010, 175587, 175585. Gib jeweils die Anzahl der gelesenen Blöcke und die Anzahl der verglichenen Schlüsselwerte an. **10 P**

b) Wie sind die Daten auf die Buckets in Variante 1 verteilt, d. h. wieviele Blöcke sind (durchschnittlich, minimal, maximal) in einem Bucket. Entspricht das deiner Erwartung? Warum (nicht)? **5 P**

c) Was ist die Gesamtanfragezeit für alle unter a) gesuchten Werte für beide Speichervarianten? Entspricht das deiner Erwartung? Warum (nicht)? Wie würde sich das Verhältnis zwischen beiden Laufzeiten verändern, wenn die Blöcke im Sekundärspeicher gehalten würden? **5 P**

Hinweise:

- Implementiere die beschriebenen Datenstrukturen und die Anfragen in Java 1.5 oder höher.
- Verwende für Blöcke das unter lehrveranstaltungen\DBSII_naumann gegebene Interface.
- Der Java Heapspace muss vermutlich erhöht werden, um die Daten zu halten. (java -Xmx<size> -Xms<size> ...; sinnvolle Größe für <size> = 200M)
- Ausführbare Klasse: de.unipotsdam. hpi. is. dbsii. indexes.DataStoreExec
- Ausgabeformat:

```
nonIndexed:
<key>: <value>, #blocks read: <count>, #values compared: <count>
...
runtime: <runtime in ms>

indexed:
<key>: <value>, #blocks read: <count>, #values compared: <count>
...
runtime: <runtime in ms>
```

- Sende die java-Dateien an die oben angegebenen mail-Adressen.