
 Aufgabe 1 - *Slotted Page*.

1 Punkt

Gegeben sei eine Slotted Page mit folgenden Eigenschaften:

- Größe: $2^{13} = 8192$ Bytes,
- Adressierungstyp: **Byte-Adressierung** (es kann jedes Byte adressiert werden)

In dieser Slotted Page werden **3 Tupel** O , P , Q gespeichert:

- d_1 : $|O| = 64$ Bytes
- d_2 : $|P| = 128$ Bytes
- d_3 : $|Q| = 256$ Bytes

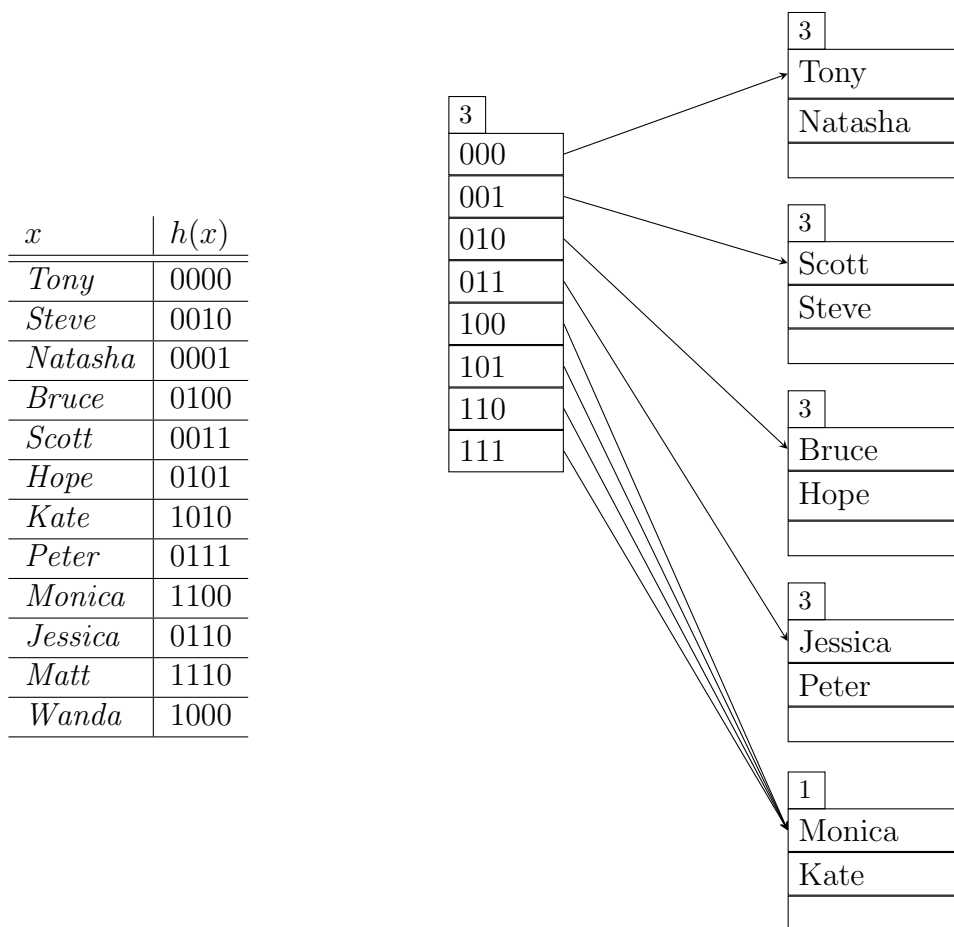
Ergänzen Sie die Slotted Page um die **fehlenden Werte/Adressen** (numerische Werte erwartet, Pfeile reichen nicht aus), wobei p_i und g_i sich auf den jeweiligen Datensatz d_i beziehen. (**0.125 Punkte pro Wert/Adresse**)

a	f	g_3	p_3	g_2	p_2	g_1	p_1	\dots	d_1	d_2	d_3
									O	P	Q

Aufgabe 2 - Erweiterbares Hashing.

1 Punkt

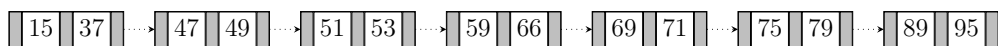
Die Hashfunktion $h(x)$ liefert die in der Tabelle angegebenen Binärwerte. Es sollen die Tupel **Matt** und **Wanda** in dem gegebenen Hashcontainer **eingefügt** werden. Ein Bucket im Hashcontainer kann bis zu 3 Tupel speichern. Das Verzeichnis soll so klein wie möglich gehalten werden. **Illustrieren Sie den resultierenden Hashcontainer.**



Aufgabe 3 - B^+ -Baum-Konstruktion.

1 Punkt

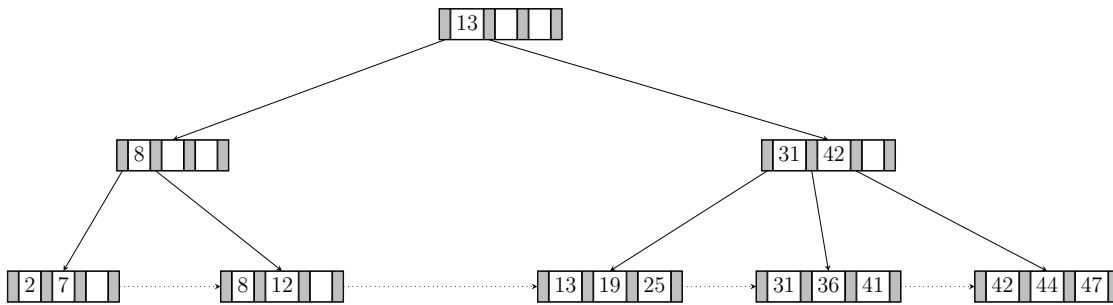
Gegeben sind die Blätter eines B^+ -Baumes ($m = 3$). Konstruieren Sie die **darüberliegenden Ebenen** (d.h. die inneren Knoten) so, dass ein **gültiger B^+ -Baum mit minimaler Höhe** entsteht.



Aufgabe 4 - B^+ -Baum-Löschen.

1 Punkt

Gegeben ist ein B^+ -Baum mit $m = 4$. Zeichnen Sie den B^+ -Baum, der nach dem Löschen von **8** entsteht.



Aufgabe 5 - Indexstrukturen.

1 Punkt

Zeichnen Sie für die folgende Tabelle einen **3-stufigen Sekundärindex** auf dem Attribut **Stadt**. Die **erste Indexstufe** soll **dense** und die **zweite und dritte Indexstufe** sollen **sparse** sein. In einen **Indexblock** können **3 Einträge** gespeichert werden.

Stadt	KFZ
Rom	I
London	GBM
Prag	CZ
Kiew	UA
Berlin	D
Athen	GR
Krakau	PL
Oslo	N
Dublin	IRL
Wien	A

Aufgabe 6 - Effiziente Anfragebearbeitung.**1 Punkt**

Gegeben sei eine Relation $R[A, B, C, D]$ mit folgenden Eigenschaften:

- $|R| = 2.000.000$ Tupel.
- Pro Datenblock werden 400 Tupel gespeichert.
- Attribut A hat ganzzahlige Werte gleichverteilt im Bereich $[1; 2.000.000]$.
- Attribut B hat ganzzahlige Werte gleichverteilt im Bereich $[400.001; 500.000]$.
- Attribut C hat ganzzahlige Werte gleichverteilt im Bereich $[100.001; 1.000.000]$.
- Attribut D hat ganzzahlige Werte gleichverteilt im Bereich $[1; 10.000.000]$.
- Es existieren folgende Indizes:
 - sparse B⁺-Baum-Index auf Attribut A .
 - dense B⁺-Baum-Index auf Attribut B .
 - dense B⁺-Baum-Index auf Attribut D .
- Für alle B⁺-Baum-Indizes gilt:
 - $m = 2^8 = 256$.
 - Sie besitzen minimale Höhe.
 - 1 Knotenzugriff entspricht 1 Blockzugriff.

Es soll folgende Anfrage beantwortet werden:

$$\sigma_{A>1.800.000 \wedge B=450.000}(R)$$

Geben Sie die **Strategie und die Blockzugriffe** an, um die Anfrage **möglichst effizient** zu beantworten.

Aufgabe 7 - Join-Algorithmen.**1 Punkt**

Welcher Join Algorithmus (**Merge Join vs. Index Nested Loop Join**) generiert die minimalen Kosten für das folgende Szenario? Geben Sie in der Lösung die **Algorithmen** und die **dazugehörigen** Kosten an.

Berechnen Sie einen **Natural Join** zwischen zwei Relationen $R[A, B]$ und $S[A, C]$, wobei $|R| = 50$ Tupel und $|S| = 5000$ Tupel. Die Relationen sind auf $b_R = 10$ bzw. $b_S = 2000$ hintereinander liegenden Blöcken gespeichert. Es existiert ein sparse B⁺-Baum Index auf $S.A$ mit $m = 8$. Auf $R.A$ existiert kein Index, falls Sie eine Sortierung benötigen, nehmen Sie an dass diese $\lceil b_R \cdot \log_2(b_R) \rceil$ viele Blockzugriffe zusätzlich kostet.

Aufgabe 8 - Effiziente Anfragebearbeitung.**1 Punkt**

Gegeben sei eine Relation $R[A, B]$ mit folgenden Eigenschaften:

- $|R| = 10^7$ Tupel,
- Pro Datenblock werden 32 Tupel gespeichert,
- Attribut A hat ganzzahlige Werte gleichverteilt im Bereich $[1; 4 \cdot 10^4]$,
- Attribut B hat ganzzahlige Werte gleichverteilt im Bereich $[1; 3 \cdot 10^4]$,
- Duplikate werden mittels Tuple Identifier (TID) aufgelöst,
- Folgende Indizes existieren:
 - Clustered B⁺-Baum-Index auf Attribut A , $m = 64$
 - Dense B⁺-Baum-Index auf Attribut B , $m = 64$

Es soll folgende Anfrage beantwortet werden:

$$\sigma_{A < 8000 \wedge B > 15000}(R)$$

Geben Sie die **Strategie (0.5 Punkte)** an und berechnen Sie die **Anzahl der Blockzugriffe (0.5 Punkte)** um die Anfrage **möglichst effizient** zu beantworten (1 Knotenzugriff im B⁺-Baum entspricht 1 Blockzugriff).

Aufgabe 9 - Join Kardinalität.**1 Punkt**

Gegeben sind folgende Relationen:

- $|R[A, B, C]| = 1000, V(R, A) = 100, V(R, B) = 200, V(R, C) = 300$
- $|S[A, D, E]| = 4000, V(S, A) = 50, V(S, D) = 200, V(S, E) = 300$
- $|T[C, D, F]| = 2000, V(T, C) = 100, V(T, D) = 200, V(T, F) = 600$

Die Werte in den Tupeln sind gleichverteilt und unabhängig. Schätzen Sie die Kardinalität des Ergebnisses der folgenden Anfrage ab ($\sigma_{A=100 \vee A=200}(R) \neq \emptyset$).

$$(\sigma_{A=100 \vee A=200}(R)) \bowtie S \bowtie T$$

Aufgabe 10 - Anfrageoptimierung, Join-Reihenfolge.**1 Punkt**

Gegeben seien 3 Relationen $R[A, B, C]$, $S[C, D, E]$ und $T[C, F, G]$ mit folgenden Eigenschaften:

- $|R| = 10^5$ Tupel, $V(R, A) = 10^3$, $V(R, B) = 4 \cdot 10^3$, $V(R, C) = 10^4$
- $|S| = 10^7$ Tupel, $V(S, C) = 2 \cdot 10^3$, $V(S, D) = 5 \cdot 10^3$, $V(S, E) = 8 \cdot 10^3$
- $|T| = 10^9$ Tupel, $V(T, C) = 10^9$, $V(T, F) = 6 \cdot 10^3$, $V(T, G) = 9 \cdot 10^3$

Weiters sei folgende SQL-Anfrage gegeben:

```
SELECT R.C, S.E, T.F
FROM   R, S, T
WHERE  R.C = S.C AND S.C = T.C
```

- a. Zeichnen Sie die **algebraische Normalform als Operatorbaum** für die gegebene SQL-Anfrage. **(0.5 Punkte)**
- b. Wenden Sie **heuristische Optimierung** an, um den **Operatorbaum zu optimieren**. Im resultierenden Operatorbaum soll die **Join-Reihenfolge optimal** sein (d.h. es soll zuerst der Join mit dem kleinsten Zwischenergebnis durchgeführt werden). **(0.5 Punkte)**