Aufgabe 8.1:

Lösung:

Sei $N = 10^6$ die Anzahl Tupel, $R = 50$ die Anzahl Tupel pro Seite und $T = 10msec$ die im Mittel benötigte Zeit für einen Externzugriff. Die Mindestanzahl benötigter Seiten ist dann $B = \frac{N}{R} = 2 \cdot 10^4$.

Für eine Abschätzung der benötigten Anzahl Externzugriffe müssen Annahmen getroffen werden, ob eine lückenlose Füllung der Seiten betrachtet werden soll. Falls ja, muss einbezogen werden, wie dies bei Einfügungen und Löschungen aufrechterhalten werden kann. Falls nein, muss die mittlere Füllung festgelegt werden. Wir nehmen an, dass im Mittel die Seiten zu 67% gefüllt sind.

Für die Dateiorganisationsform Sortierung nehmen wir an, dass die Seiten der Datei lückenlos gefüllt sind. Die Anzahl der benötigten Seiten beträgt somit gerade B. Man beachte, dass bei Einfügungen und Löschungen im Mittel $\frac{B}{2}$ Seiten gelesen und wieder geschrieben werden müssen, da unter der Annahme der lückenlosen Füllung z.B. bei Einfügungen Platz geschaffen werden muss.

Für die Organisationsformen Haufen, Hashfunktion und Suchbaum betrachten wir keine lückenlose Füllung und erhalten für die Anzahl benötigter Seiten den Wert $B' = \frac{3}{2} \cdot B = 3 \cdot 10^4$. Für die Hashfunktion können wir somit Überlaufseiten ignorieren.

Bei Einsatz eines B-Baums als Suchbaum für den Index müssen die Zugriffe innerhalb des Indexes berücksichtigt werden. Wir nehmen an, dass die Länge eines Indexeintrags 10% der Länge des betreffenden Satzes ist und dass die Seiten zu 67% gefüllt sind. Wir wollen des Weiteren annehmen, dass die Wurzel im Internspeicher gehalten wird. Der Verzweigungsfaktor des Baumes beträgt somit $F = 333$ und die resultierende Höhe beträgt $H = 2$.

Es ergeben sich die folgenden Abschätzungen:

	Suchen	Einfügen	Löschen	Ändern
Haufenorganisation	$\frac{B'}{2}$	2	$\frac{B'}{2}+1$	$\frac{B'}{2}+1$
Sortierung	$\log_2 B$	$\log_2 B + B$	$\log_2 B + B$	$\log_2 B + 1$
Suchbaum	$\log_F B'$	$\log_F B' + 1$	$\log_F B' + 1$	$\log_F B' + 1$
Hashfunktion	1	2	2	2

Setzen wir die konkreten Werte ein, so ergeben sich die folgenden ungefähren Zeiten:

	Suchen	Einfügen	Löschen	Ändern
Haufenorganisation	2,5 min	20 msec	2,5 min	2,5 min
Sortierung	150 msec	3,3 min	3,3 min	160 msec
Suchbaum	20 msec	30 msec	30 msec	30 msec
Hashfunktion	10 msec	20 msec	20 msec	20 msec

Aufgabe 8.2:

Lösung:

(a) Wir nehmen an, dass die zu sortierende Datei nicht komplett in den Hauptspeicher passt. Für *extSort*() wird eine Variante des Mergesort verwendet. Es werden zunächst Teilfolgen mittels *Sort*() im Internspeicher sortiert und dann mittels Mischen zusammengeführt.

Seien k Pufferseiten verfügbar und sei N die Anzahl Seiten der Datei.

Durchlauf 0: Es werden jeweils k aufeinander folgende Seiten der Datei gelesen, intern mittels *Sort*() sortiert und wieder ausgegeben. Es werden $\lceil N/k \rceil$ sortierte Teilfolgen (sog. *Runs*) ausgegeben.

Durchlauf $i = 1, 2, \ldots$**:** Verwendet $k-1$ Puffer-Seiten für die Eingabe und die verbleibende Seite für die Ausgabe. Mische $k-1$ Runs, die innerhalb des vorherigen Durchlaufs erzeugt wurden, zu einem neuen Run.

Es werden $\lceil \log_{k-1} \lceil \frac{N}{k} \rceil \rceil + 1$ Durchläufe benötigt; in jedem Durchlauf werden alle N Seiten gelesen und geschrieben. Anzahl Seitenzugriffe:

$$(\lceil \log_{k-1} \lceil \frac{N}{k} \rceil \rceil + 1) 2 \cdot N$$

(b) Für $k = 3$ erhalten wir somit $(\lceil \log_2 \lceil \frac{N}{3} \rceil \rceil + 1) \cdot 2 \cdot N$ Seitenzugriffe.

(c) Sei k beliebig und $N = 10^6$. Wir erhalten $(\lceil \log_{k-1} \lceil \frac{10^6}{k} \rceil \rceil + 1) \cdot 2 \cdot 10^6$ Seitenzugriffe. Falls $k = 100$ folgen $(\log_{99} 10^4 + 1) \cdot 2 \cdot 10^6 \approx 6 \cdot 10^6$ Seitenzugriffe.

Aufgabe 8.3:

(a)

Lösung: Für $i = 0$ liefert h_0 gerade die letzten b Bits der durch $B(v)$ erzeugten Binärzahl. Für $i > 0$ dividieren wir durch 2^{b+i}. Damit erhalten wir jeweils die letzten $b + i$ Bits von $B(v)$.

(b)

Lösung: (vergl. *Linear Hashing* (Ramakrishnan und Gehrke, 2003 (Chapter 11.3))

1) Nach Ende einer Runde hat sich die Anzahl Seiten verdoppelt. Es existieren 2^{b+i} Seiten.
2) $next' = 2^{b+i} + next$.
3) Es kann das $b+i$-letzte Bit von $B(v)$ getestet werden. Ist dies gleich 0, so bleibt das Tupel mit Suchschlüsselwert v in der Seite $next$. Anderenfalls wird es in die Seite $next'$ umgespeichert.
4) Gilt gerade $next \leq h_i(v) \leq 2^i \cdot N$, dann ist die Seite $h_i(v)$ die zugehörige Seite. Anderenfalls wende zur Bestimmung der Seite h_{i+1} an.