Solution Set 3

Solution Set provided courtesy of Shankar Ponnekanti

1 Problem 1.

(a) Since the records are stored contiguously, we only need to store a pointer to the first block. The location of all other blocks can be calculated by the offset from the beginning of the index. Therefore, the index will have one block pointer followed by a sequence of key values giving the first key of each block. A block has 4096 bytes and each key value is 5 bytes. So number of key values in one block $= \left\lfloor \frac{4096}{5} \right\rfloor = 819$ keys per block. But the first index block can have only 818 keys since it also stores the block pointer pointing to the fist block of the file. Hence

$$\text{Number of blocks in the index} = 1 + \left\lfloor \frac{10^6 - 818}{819} \right\rfloor = 1222$$

(b) Since the index blocks are contiguous, with an organization similar to (a), only 1222 key values need be stored in the second level index.

$$\text{Number of blocks in the second level index} = 1 + \left\lfloor \frac{1222 - 818}{819} \right\rfloor = 2$$

(c) Since we can easily search for a record within a block, we can minimize the index size by storing block pointers only. A (key, block pointer) takes 9 bytes. Number of such pairs that can be put in 1 block $= \left\lfloor \frac{4096}{9} \right\rfloor = 455$. Total number of pairs = Number of records = $10^7$. Hence

$$\text{Number of blocks in the index} = \left\lfloor \frac{10^7}{455} \right\rfloor = 21979$$

(d) Since the first level index blocks are contiguous, we can build the second level index as 1 block pointer followed by first key of every block.

$$\text{Number of blocks in the second level index} = 1 + \left\lfloor \frac{21979 - 818}{819} \right\rfloor = 27$$
2 Problem 2.

(a) \[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12 \\
13 & 14 & & \\
\end{array}
\]

(b) \[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12 \\
13 & 14 & & \\
\end{array}
\]

(c) \[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12 \\
13 & 14 & & \\
\end{array}
\]

3 Problem 3.

(a) Root should have at least 2 children. Each child has at least \( \left\lceil \frac{n}{2} \right\rceil \) record pointers. Hence,

\[\text{Minimum number of record pointers} = 2 \left\lfloor \frac{n}{2} \right\rfloor\]

(b) Again, root should have at least 2 children and each child has at least \( \left\lceil \frac{n}{2} \right\rceil \) pointers. Number of leaf nodes = \( 2 \left\lfloor \frac{n}{2} \right\rfloor \). Each leaf node has at least \( \frac{n}{2} \) record pointers. Hence,

\[\text{Minimum number of record pointers} = 2 \left\lfloor \frac{n}{2} \right\rceil \]

(c) Proceeding similar to case (b), we have for \( j \) levels

\[\text{Minimum number of record pointers} = 2 \left\lfloor \frac{n}{2} \right\rceil \left( \left\lceil \frac{n}{2} \right\rceil \right)^{j-2}\]

(d) As seen in (c), a B-tree with \( j \) levels points to at least \( 2 \left\lfloor \frac{n}{2} \right\rceil \left( \left\lceil \frac{n}{2} \right\rceil \right) \) records. Hence

\[r \geq 2 \left\lfloor \frac{n}{2} \right\rceil \left( \left\lceil \frac{n}{2} \right\rceil \right)^{j-2}\]

\[j - 2 \leq \frac{\log \frac{r}{2}}{\log \frac{n}{2}}\]

\[j \leq 2 + \frac{\log \frac{r}{2}}{\log \frac{n}{2}}\]

Hence the maximum number of levels is given by \( 2 + \frac{\log \frac{r}{2}}{\log \frac{n}{2}} \)
4 Problem 4

(i) Time for search = \( t \) (say). Then

\[
t = \text{No. of blocks} \times \text{Time for retrieving and searching each block}
\]

\[
= \log_m N (70 + 0.05m + a + b \log_2 m)
\]

Using \( a << 70 \), we have

\[
t = \log_m N (70 + 0.05m + b \log_2 m)
\]

\[
= \frac{\ln N}{\ln m} (70 + 0.05m + b \log_2 m)
\]

\[
= \ln N \frac{70 + 0.05m}{\ln m} + b \ln N \log_2 e
\]

Since \( \ln N \) and \( \log_2 e \) are independent of \( m \), we are left with minimizing \( \frac{70 + 0.05m}{\ln m} \). Differentiating w.r.t \( m \), we get

\[
m(\ln m - 1) = 1400
\]

This can be solved using the simple iterative method \( m = 1400 / (\ln m - 1) \) which converges very fast. The integral value of \( m \) that minimizes \( \frac{70 + 0.05m}{\ln m} \) and hence minimizes \( t \) is 298. It can be easily seen that at this value of \( m \), \( t \) has a minima and not a maxima. This check needs to be made since derivative is 0 both at minima and maximas.

(ii) Let seek and latency time = \( t_s \). In the first part, we had \( t_s = 70 \text{ms} \). We have

\[
\text{Net seek and latency time} = t_s \log_m N
\]

\[
\text{Net transfer time} = 0.05m \log_m N = 0.05 \log_2 N \frac{m}{\log_2 m}
\]

\[
\text{Net binary search time} = \log_m N \times b \log_2 m = b \log_2 N
\]

Thus, as \( m \) varies, only net seek and latency time and net transfer time vary. Further while net seek and latency time is a decreasing function of \( m \), net transfer time is an increasing function of \( m \). If \( t_s \) is very small, then net seek and latency time is negligible and net transfer time becomes dominant.

To reduce that, we choose a small value of \( m \). If \( t_s \) is very large, then net seek and latency time becomes the dominant factor and we choose a large \( m \) to reduce that. From these limiting cases, it is clear that as \( t_s \) decreases, the optimum value of \( m \) also decreases.

For instance, when \( t_s = 35 \text{ms} \), then the expression to minimize becomes \( \frac{35 + 0.05m}{\ln m} \). This is minimum when

\[
m(\ln m - 1) = 700
\]

The integral value of \( m \) which minimizes \( t \) is 169 and is obtained just as in part (i) above.