Warmup #1  (Textbook Problem 11.3)

Construct a B+-tree for the following set of key values:

\((2, 3, 5, 7, 11, 17, 19, 23, 29, 31)\)

Assume that the tree is initially empty and values are added in ascending order. Construct B+-trees for the cases where the number of pointers that will fit in one node is as follows:

a. Four
b. Six
c. Eight

Answer:
The following were generated by inserting values into the B+- tree in ascending order. A node (other than the root) was never allowed to have fewer than \(\lceil n/2 \rceil\) values/pointers.

![B+-tree with 4 pointers](image)

b. 

![B+-tree with 6 pointers](image)

c. 

![B+-tree with 8 pointers](image)
Warmup #2  (Textbook Problem 11.5)

Consider the modified redistribution scheme for B+-trees described on page 501 (given below). What is the expected height of the tree as a function of n?

During deletion of a record, if the occupancy of a node falls below 2n/3, the system attempts to borrow an entry from one of the sibling nodes. If both sibling nodes have 2n/3 records, instead of borrowing an entry, the system redistributes the entries in the node and in the two siblings evenly between two of the nodes, and deletes the third node. We can use this approach because the total number of entries is 32n/3 - 1, which is less than 2n. With three adjacent nodes used for redistribution, each node can be guaranteed to have 3n/4 entries. In general, if m nodes (m - 1 siblings) are involved in redistribution, each node can be guaranteed to contain at least (m - 1)n/m entries. However, the cost of update becomes higher as more sibling nodes are involved in the redistribution.

Answer: If there are K search-key values and m - 1 siblings are involved in the redistribution, the expected height of the tree is: \( \log_{\left\lfloor \frac{(m-1)n}{m} \right\rfloor} (K) \)

Warmup #3  (Textbook Problem 11.14a)

Why might the leaf nodes of a B+-tree file organization lose sequentiality? Suggest how the file organization may be reorganized to restore sequentiality.

Answer: When a file organization is newly created on a set of records, it is possible to allocate blocks that are mostly contiguous on disk to leaf nodes that are contiguous in the tree. Thus a sequential scan of leaf nodes would correspond to a mostly sequential scan on disk. As insertions and deletions occur on the tree, sequentiality is increasingly lost, and sequential access has to wait for disk seeks increasingly often. An index rebuild may be required to restore sequentiality.

Warmup #4  (Textbook Problem 11.26)

How does data encryption affect index schemes? In particular, how might it affect schemes that attempt to store data in sorted order?

Answer: Note that indices must operate on the encrypted data or someone could gain access to the index to interpret the data. Otherwise, the index would have to be restricted so that only certain users could access it. To keep the data in sorted order, the index scheme would have to decrypt the data at each level in a tree. (This is very costly and sometimes regarded unsafe.) Note that hash systems would not be affected.
Problem 5 (To Be Graded)

Imagine that you have the following table from the Vapor online game distributor service from the last homework:

<table>
<thead>
<tr>
<th>gameId</th>
<th>gameTitle</th>
<th>distributorId</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Doom Classic Complete Edition</td>
<td>14</td>
<td>7.44</td>
</tr>
<tr>
<td>2</td>
<td>SUPERHOT</td>
<td>19</td>
<td>14.99</td>
</tr>
<tr>
<td>3</td>
<td>Grand Theft Auto V</td>
<td>53</td>
<td>35.99</td>
</tr>
<tr>
<td>4</td>
<td>Space Empires V</td>
<td>66</td>
<td>2.99</td>
</tr>
<tr>
<td>5</td>
<td>Monstrum</td>
<td>22</td>
<td>4.49</td>
</tr>
<tr>
<td>6</td>
<td>Fasaria World Online</td>
<td>87</td>
<td>4.49</td>
</tr>
<tr>
<td>7</td>
<td>The Kindred</td>
<td>62</td>
<td>7.49</td>
</tr>
<tr>
<td>8</td>
<td>The Black Watchmen</td>
<td>55</td>
<td>7.49</td>
</tr>
<tr>
<td>9</td>
<td>Fallout Classic Collection</td>
<td>34</td>
<td>9.99</td>
</tr>
<tr>
<td>10</td>
<td>DETOUR</td>
<td>6</td>
<td>0.49</td>
</tr>
<tr>
<td>11</td>
<td>Viking Brothers</td>
<td>17</td>
<td>1.99</td>
</tr>
<tr>
<td>12</td>
<td>Motorsport Manager</td>
<td>33</td>
<td>31.49</td>
</tr>
<tr>
<td>13</td>
<td>Metro 2033 Redux</td>
<td>92</td>
<td>19.99</td>
</tr>
<tr>
<td>14</td>
<td>Raiseiland</td>
<td>90</td>
<td>33.49</td>
</tr>
<tr>
<td>15</td>
<td>Poker Night 2</td>
<td>77</td>
<td>4.99</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1024</td>
<td>Rocket League</td>
<td>63</td>
<td>19.99</td>
</tr>
</tbody>
</table>

The table contains 1024 rows in total. Answer the following questions about this table:

1. Suppose the database administrator decides to store gameId as an 8-byte integer, gameTitle as a 48-byte character array, distributorId as a 8-byte integer, and price as a 4-byte float. If an instance of gameTitle is shorter than 48 bytes, the empty space will be filled with null characters.
   - All attributes of a tuple are stored in contiguous space within the same disk block\(^1\).
   - A disk block size is 512 bytes.
   - The disk on average performs the sequential read at 1\(ms\) per disk block and the random read at 10\(ms\) per disk block.

   (a) What is the maximum number of tuples that can be stored in a disk block?
   Answer: The maximum number of tuples can be stored in a disk block is
   \[
   \left\lfloor \frac{512B}{8B + 48B + 8B + 4B} \right\rfloor = 7
   \]

   (b) What is the minimum number of disk blocks that need to be allocated for all tuples in the table?
   Answer: The minimum number of disk blocks that need to be allocated for all tuples in the table is
   \[
   \left\lceil \frac{1024 \text{ tuples}}{7 \text{ tuples per block}} \right\rceil = 147 \text{ blocks}
   \]

   (c) What is the minimum time to read all tuples (in no particular order), assuming that the minimum number of disk blocks are allocated?
   Answer: The minimum time to read all tuples is when disk blocks are allocated next to each other. One random read incurs at the first block, and all sequential reads for the rest blocks thereafter.
   \[
   10ms \times 1 + 1ms \times (147 - 1) = 156ms
   \]

\(^1\)The minimum storage unit on a disk is called a disk block or a disk page.
2. Suppose that a secondary index of B+-tree is created on the distributorId column. Assume the following:

- Each tree node has a size of 64 bytes.
- The data in a tree node are stored in contiguous space within the same disk block.
- The tree has a fanout of 4 and each leaf node is 60% loaded\(^2\).
- A pointer is 8-byte long.
- You can assume distributorId is a candidate key with unique values for simplicity.

(a) How many bytes of space does the secondary index require?
Answer: The number of entries in a node is
\[
\left\lfloor \frac{64 \text{B per node} \times 60\%}{8 \text{B per distributorId} + 8 \text{B per pointer}} \right\rfloor = 2
\]
The number of leaf nodes is
\[
\frac{1024 \text{ distributorIds}}{2 \text{ distributorId per node}} = 512
\]
The height of the tree is
\[
\lceil \log_4(512) \rceil = 5
\]
The total number of nodes in the tree is
\[
512 + 128 + 32 + 8 + 2 + 1 = 683
\]
The total size of the tree is
\[
\text{node size} \times \text{num. nodes} = 64 \text{B} \times 683 = 43712\text{B}
\]

(b) How many disk reads (including index search and tuple retrieval) in the worst case are required to find a tuple by its distributorId?
Answer: The height of the tree is
\[
\lceil \log_4(512) \rceil = 5
\]
In the worst case, the number of disk reads is equal to the number of nodes visited plus one disk read to retrieve the tuple, or
\[
(1 + \text{tree height}) + (1 \text{ tuple read}) = 1 + 5 + 1 = 7
\]

\(^2\)The ratio of the number entries over the maximum number of entries is at least 60%.