Warmup #1  (Textbook Problem 11.15)

When is it preferable to use a dense index rather than a sparse index? Explain your answer.

Warmup #2  (Textbook Problem 11.16)

What is the difference between a clustering index and a secondary index?

Warmup #3  (Textbook Problem 11.6)

Suppose that we are using extendable hashing on a file that contains records with the following search-key values:

\[ 2, 3, 5, 7, 11, 17, 19, 23, 29, 31 \]

Show the extendable hash structure for this file if the hash function is \( h(x) = x \mod 8 \) and buckets can hold three records.

Warmup #4  (Textbook Problem 11.22)

Up until now, the indexes covered in class have all been single-field indexes. However, we can also have what’s called a compound index, which allows a dataset to be sorted by multiple attributes. Suppose there is a relation \( r(A,B,C) \), with a \( B^+ \)-tree index with search key \((A,B)\). This means that \( r \) is sorted first by \( A \), then by \( B \).

1. What is the worst-case cost of finding records satisfying \( 10 < A < 50 \) using this index, in terms of the number of records retrieved \( n_1 \) and the height \( h \) of the tree?

2. What is the worst-case cost of finding records satisfying \( 10 < A < 50 \land 5 < B < 10 \) using this index, in terms of the number of records \( n_2 \) that satisfy this selection, as well as \( n_1 \) and \( h \) defined above?

3. Under what conditions on \( n_1 \) and \( n_2 \) would the index be an efficient way of finding records satisfying \( 10 < A < 50 \land 5 < B < 10 \)?
Problem 5 (To Be Graded)

Given the above primary index of B+-tree of maximum fanout of 3, please answer the following questions.

1. List the nodes in the order of traversal for the following look up requests. You may label a node as a 2-tuple of the keys, e.g. (9, null) and (1, 3).
   (a) Look up key 13.
   (b) Look up key 8.
   (c) Look up keys in range [6, 16] (inclusive range).

2. Draw the tree after the following operations executed in the listed order. In case of leaf node split, assume two entries move to the new leaf.
   (a) Insert 6.
   (b) Insert 10.
   (c) Delete 9.

3. Assume the original tree. How many keys at most can be inserted without any internal node split at the second level of the tree (i.e. nodes (5, null) and (13, 17))? 

Problem 6 (To Be Graded)

Consider the following hash buckets, each with a maximum size of 4 elements:

<table>
<thead>
<tr>
<th>catalog</th>
<th>buckets</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>8, 12, 24, 36</td>
</tr>
<tr>
<td>01</td>
<td>5, 17, 21</td>
</tr>
<tr>
<td>10</td>
<td>2, 18, 22</td>
</tr>
<tr>
<td>11</td>
<td>3, 7, 23, 31</td>
</tr>
</tbody>
</table>

Add the elements 11, 30, 28, 10 and 39 (in this order) using the following hashing strategies:

1. Use extendible hashing to add the elements. The buckets you start with have a global depth of 2, and each individual bucket also has a local depth of 2. If multiple entries in the directory point to the same bucket, be sure to indicate it in your solution. Please illustrate the full state of your hash buckets after each addition, including changes to the catalog and any change to global or local depth.

2. Use linear hashing to add the elements. Note that the ”catalog” value is used purely for illustration, but you should maintain the values in your solution. The Next pointer starts by pointing at the first bucket (labeled 00), and your value for N = 4. Please illustrate the full state of your hash buckets after each addition, including where the Next pointer is pointing and additions to the ”catalog” values.