Fall, 2004 CIS 550

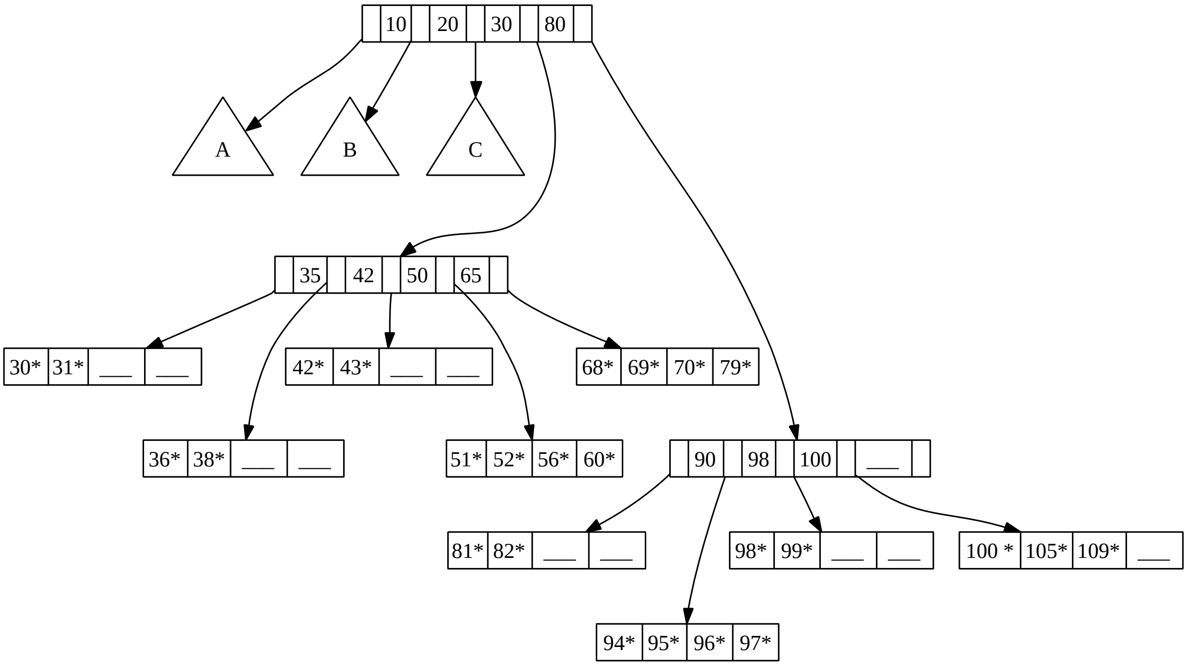
Database and Information Systems

Homework 6 Solutions

Problem 1 [50 points]:

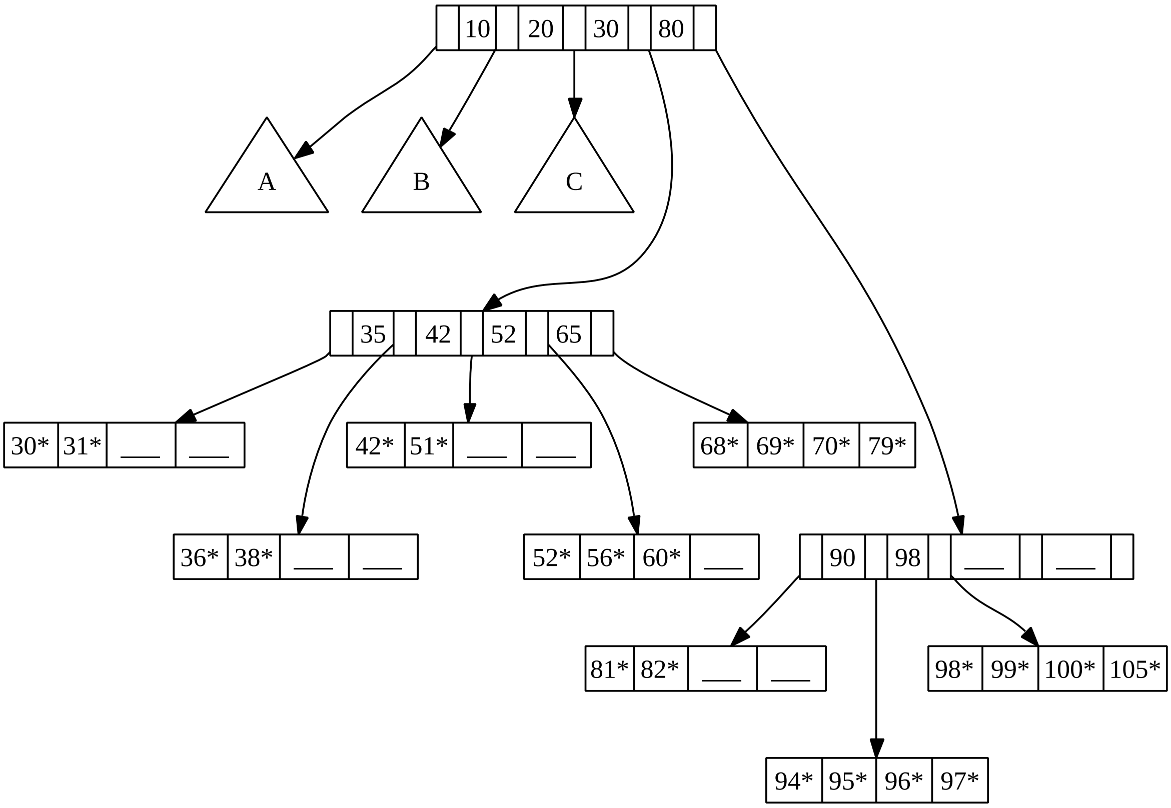
1. Insert a record with search key 109 into the tree.

   Answer:



2. Delete the record with search key 43 from the tree.

   Answer:



3. Name a search key value such that inserting it into the original tree would cause an increase in the height of the tree.

   Answer:

   Inserting 61, for example, would cause a split up to the root, increasing the height of the tree by 1.

Problem 2 [50 points]:

1. Start by running the query:

   ```
   select max(l_linenumber)
   from zives.lineitem
   where l_linenumber >= 0;
   ```

   which uses whatever indices Oracle prefers. Time the run and repeat it 4 times, averaging the total. What is that value?

   Answer: We got about 3 seconds. But, throughout this problem, your results may vary widely. Grading was based on quality of analysis.

2. Next, repeat the query with a special *optimizer hint* that says you want to scan the full table and avoid indices:

   ```
   select /*+ full(l) */ max(l_linenumber)
   from zives.lineitem l
   where l_linenumber >= 0;
   ```

   Average the running time for 5 runs and record the result. What can you say about the difference between the two runs? Do these results suggest that Oracle uses an index by default?

   Answer: We got about 30 seconds, an order of magnitude difference. This suggests strongly that Oracle does indeed use an index by default.

3. Try another hint, which makes use of the first index:

   ```
   select /*+ index(l sys_c0013539) */ max(l_linenumber)
   from zives.lineitem l
   where l_linenumber >= 0;
   ```

   Run 5 times and record the result. How did this compare to the first two cases? Did that particular index offer any benefits?

   Answer: About 3 seconds. Similar to the first case.

4. Which index was used in the default, un-hinted Oracle query? Generalizing from the Oracle SQL hints shown above, what is the query hint that is equivalent to that default query?

   Answer: From the results above, it appears that Oracle uses `sys_c0013539` by default.

5. For each of the two indices *and* for the non-indexed case, explain how what data and/or index structures will be examined in answering the query.

   Answer: For `lineno`, since the index is sorted on `l_linenumner`, only the last entry of the index needs to be accessed to find the maximum value. For `sys_c0013539`, the entire index must be scanned, looking for the maximum value only of the `l_linenumber` attribute. For the non-indexed case, all entries in the lineitem relation will be scanned sequentially.

6. Run the two queries:

   ```
   select /*+ index(l sys_c0013539) */ count(distinct l_suppkey)
   from zives.lineitem l
   where l_linenumber < 1100;
   ```

```
select /*+ index(l lineno) */  count(distinct l_suppkey)
from zives.lineitem l
where l_linenumber < 1100;
```

Record the running times. Explain the relative differences in performance, based on your knowledge of index and data layout characteristics, as well as your knowledge of "covering" indices.

Answer: About 35 seconds for the first query, over 3 minutes for the second query. The result is a bit counter-intuitive, because finding records satisfying the selection condition should be faster with the smaller, sorted `lineno` index. But then to retrieve the `l_suppkey` values, the actual records must be accessed. Hence the difference is likely explained by different I/O costs caused by different page-access orders. The first query scans the records in an order close to their physical order on disk. The second query, on the other hand, scans the records in an inefficient order, leading to time wasted on disk seek.