CISC637 Spring 2015, Homework 4

Due Tuesday, 4/21, at 11:55pm on Sakai.

Assignment

This assignment covers file organization, indexes, and query processing.

1. [**25 pts**] For each scenario below, calculate the number of blocks required to store the file and its indexes, and use that to estimate times to answer queries. You may assume disk blocks are 4 kilobytes (4096 bytes), and that it takes 1ms to read a block off disk. You may also assume that primary index records are 10% of the size of actual records. If you make any other assumptions, state them clearly.

   (a) Relation `course(course_id, title, dept_name, credits)` is stored as a sequential file with a primary index on `course_id`. There are 4,000 unique courses offered at the university (100 in each of 40 unique departments), and each one is stored in a 256-byte record. Estimate the time (in milliseconds) required to answer each of the following queries:

      i. `SELECT * FROM course`
      ii. `SELECT * FROM course WHERE course_id='CISC637'`
      iii. `SELECT * FROM course WHERE course_id LIKE 'CISC%'`

   (b) Relation `takes(ID, course_id, sec_id, semester, year, grade)` is stored as a heap file with a secondary index on `ID` and a secondary index on (`course_id, sec_id, semester, year`). The table has 3,500,000 unique records (35 records each for 100,000 students), and each is stored in a 512-byte record.

      The secondary index on `ID` stores 100,000 unique IDs (4 bytes each), each with the locations of all 35 records (8 bytes required to store each location).

      The secondary index on (`course_id, sec_id, semester, year`) stores 40,000 unique section identifiers (24 bytes each), each with the locations of all enrollment records for that section. Estimate the time (in milliseconds) required to answer each of the following queries:

      i. `SELECT * FROM takes`
      ii. `SELECT * FROM takes WHERE ID=1`
      iii. `SELECT * FROM takes WHERE course_id='CISC437' AND sec_id=1 AND semester='Spring' AND year=2015` (there are 45 students in the course)

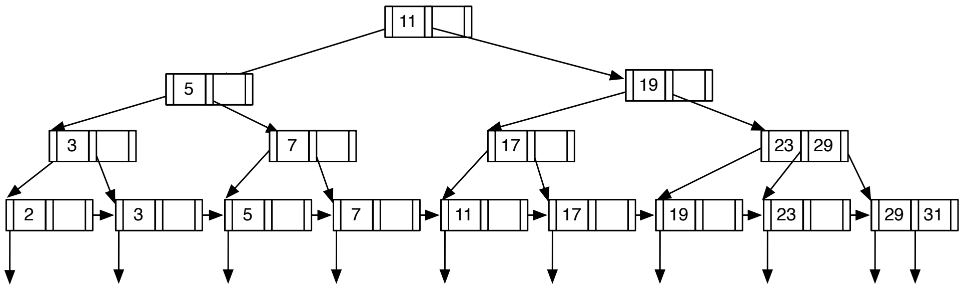
2. [**15 pts.**] The diagrams below show two small B+-tree indexes. Each node contains search key values and block pointers. Arrows represent pointers; an arrow pointing straight down from a leaf node represents a pointer to a file block in which the record with the search key value to the right of the pointer is stored. Each node in the tree is stored in one block on disk.

   Describe the steps involved in the following queries. What is the minimum number of block reads needed to complete each query for each of the indexes? (Do not assume records with consecutive search key values are stored in the same block.)

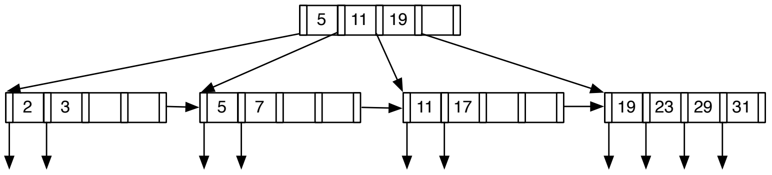
   - Find records with search-key value 19.

- Find records with search-key values between 5 and 19 (inclusive).
- Find records with search-key values less than 11.

(a) B+-tree index with $k = 3$



(b) B+-tree index with $k = 5$



3. **[25 pts.]** Relations $R_1(A, B, C)$ and $R_2(C, D, E)$ have the following properties:

   - All fields store positive integer data.
   - Field A is the primary key for R_1; field C is the primary key for R_2; $R_1.C$ is a foreign key to $R_2.C$.
   - R_1 has 20,000 records, with 200 records per block. There is a primary B+-tree index on A with height $h_A = 3$.
   - R_2 has 45,000 records, with 4,500 records per block. There is a primary B+-tree index on C with height $h_C = 3$ and a secondary B+-tree index on D with height $h_D = 2$.

   Assume that you only have enough memory to hold one block of any one relation file (whether index file or relation file) in memory at a time. For example, you can hold one block of R_1 and one block of R_2, or one block of R_1 and one block of an index on R_2. Estimate the minimum number of block reads required for each of the following operations. Explain your answers.

   - $\sigma_{B=1}(R_1)$
   - $\sigma_{C=1}(R_2)$
   - $\sigma_{D=5}(R_2)$
   - $R_1 \bowtie R_2$ (natural join; use indexed nested-loop join)
   - $R_1 \bowtie_{B=E} R_2$ (*not* a natural join; use block nested-loop join)
   - $R_1 \bowtie R_2$ (natural join; use sort-merge join and assume memory for one output block)

4. **[15 pts.]** Relation $R(A, B, C, D, E)$ has 5,000,000 records, with 10 records per block. A is a primary key with values between 0 and 100,000,000, and R is stored in sorted order of A. Each block in an index can store up to 100 index records. You have three options to locate records by search key A:

- Access the sequential file directly.
- Use a B+-tree index (of height 4).
- Use a hash index (hash function $h(a)$ is the last two digits of the value of a modulo 50; overflow buckets will be necessary)

Of those three options, for each of the relational algebra queries below, which is likely to be most efficient? Explain your answer.

(a) $\sigma_{a<50{,}000}(R)$

(b) $\sigma_{a>50{,}000}(R)$

(c) $\sigma_{a=50{,}000}(R)$

(d) $\sigma_{a>50{,}000 \wedge a<50{,}010}(R)$

(e) $\sigma_{a\neq 50{,}000}(R)$

5. **[20 pts.]** This problem is about writing algorithms to do *outer* joins. Suppose we have the following schema:

   Student($\underline{\text{ID}}$, name, dept_name, tot_cred)
   Takes($\underline{\text{ID}}$, $\underline{\text{course\_id}}$, $\underline{\text{sec\_id}}$, $\underline{\text{semester}}$, $\underline{\text{year}}$, grade)

   We want to compute a natural left outer join `Student` ⟕ `Takes`.

   - Modify the algorithm for block nested-loop join presented in the lecture slides to compute the left outer join (write pseudocode in the same style as the algorithm given).
   - Modify the algorithm for indexed nested-loop join presented in the lecture slides to compute the left outer join (write pseudocode in the same style as the algorithm given).
   - Give pseudocode for a merge-join algorithm to compute the left outer join. (Assume both files are already sorted by ID.)