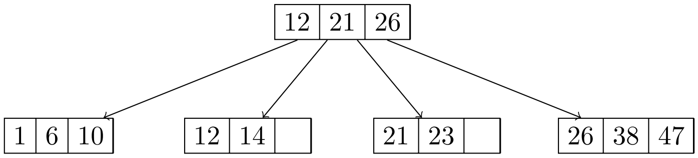
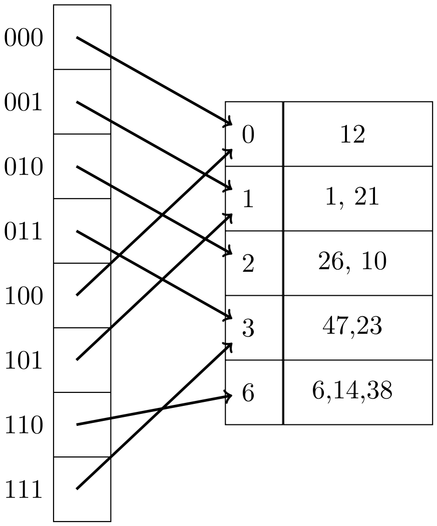
Introduction to Database Systems

Problem Set 2 Solution

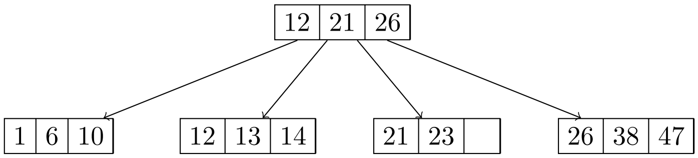
1. - B+-Tree:



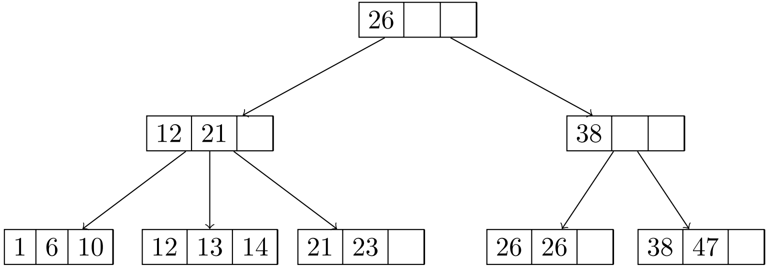
   - Extendible hash:



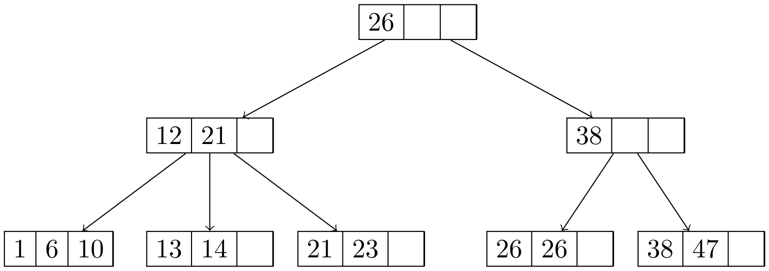
2. - Insert 13:



   - Insert 26:



   - Delete 12:



3. Page counts:

   - 1
   - 2
   - 32

4. Rewritten queries:

   -
     ```
     SELECT name, salary * 1.1 as new_salary
     FROM employees
     WHERE new_salary > 125000;
     ```

   -
     ```
     SELECT count(distinct first_name)
     FROM users;
     ```

   -
     ```
     SELECT account_id,count(*) as cnt
     FROM account
     GROUP BY account_id
     HAVING cnt > 5;
     ```

5. You are joining two tables with the following values:
   A(8,2,7,5,1)
   B(6,9,2,4,5)

   Show the steps for this join and its output using a nested loop, merge, and simple hash join. Presume that the inputs fit in memory. The hash function for the last join is v mod 3. Nested Loop

   ```
   for i in (8,2,7,5,1)
       for j in (6,9,2,4,5)
            if(i == j)
                 emit (i, j)
   ```

   Output:
   (5, 5) (2, 2)

   Merge join:

   Sort relations: (1, 2, 5, 7, 8) (2, 4, 5, 6, 9)

   Compare: (1,2) $\longrightarrow \emptyset$ (2,2) $\longrightarrow$ (2,2) (4,5) $\longrightarrow \emptyset$ (5,5) $\longrightarrow$ (5,5) (6,7) $\longrightarrow \emptyset$ (9,7) $\longrightarrow \emptyset$ (8,9) $\longrightarrow \emptyset$

   Output: (2,2) (5,5)

   Hash Join:

   Start on Bucket 0. See that none of A mods to 0, and move on to Bucket 1. Hash for 1 consists of (7, 1). We iterate over B and find that 4 mod 3 is also 1. We see that it does not match 7 nor 1 and move on.

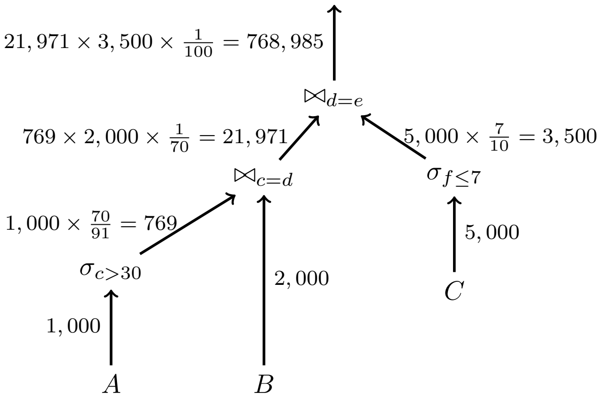
We then build the hash on A for Bucket 2. We insert 8, 2, and 5 into the bucket. We then iterate over B, comparing 2 and 5 to Bucket 2. We emit two matches (2,2) and (5,5).

For this solution, we have the build relation as A and probe relation as B. It is possible to do it the other way around (build a hash on B, probe with A).

6. For a database with three tables:

A - NCARD: 1000, for column c - ICARD: 91, MIN: 10, MAX: 100
B - NCARD: 2000, for column d - ICARD: 16, MIN: 5, MAX: 20
C - NCARD: 3000, for column e - ICARD: 96, MIN: 5, MAX: 100
D - NCARD: 5000, for column f - ICARD: 100, MIN: 1, MAX: 100
for column g - ICARD: 10, MIN: 1, MAX: 10,

- $(\sigma_{c>30}(A)) \bowtie_{c=d} B \bowtie_{d=e} (\sigma_{f\leq7}(C))$:


$21,971 \times 3,500 \times \frac{1}{100} = 768,985$
$\bowtie_{d=e}$
$769 \times 2,000 \times \frac{1}{70} = 21,971$
$5,000 \times \frac{7}{10} = 3,500$
$\bowtie_{c=d}$
$\sigma_{f\leq7}$
$1,000 \times \frac{70}{91} = 769$
$5,000$
$\sigma_{c>30}$
$2,000$
C
$1,000$
A
B

- Orders:
  ABC
  ACB
  BCA

  For the plan ABC, we two joins–AB and (AB)C–let's look at the cost of AB:
  Cost-outer(A) = Scan(A) + Sort(A) = 1,000 + 1,000 * log_2(1000) * 0.001 = 1001
  Cost-inner(B) = Scan(B) + Sort(B) = 2,022
  Comparison(AB) = $(RSICARD * w)_{AB}$ = 1,000 * 2,000 * 0.001 = 2,000
  AB cardinality estimate = 1,000 * 2,000 * $\frac{1}{91}$ = 21,978

  And now the cost of joining the intermediate results from AB to C:
  Cost-inner(C) = Scan(C) + Sort(C) = 5061
  Comparison((AB)C) = $(RSICARD * w)_{ABC}$ = 21,978 * 5,000 * 0.001 = 109890

  Total Cost = Cost-outer(A) + Cost-inner(B) + Comparison(AB) + Cost-inner(C) + Comparison((AB)C) = 119,974
  For all join orders:

Join Order	Cost-outer	Cost-inner	Comparison	Cost-inner	Comparison	Total Cost
ABC or BAC	1,001	2,022	2,000	5,061	109,890	119,974
ACB or CAB	1,001	5,061	5,000	2,022	100,000	113,084
BCA or CBA	2,002	5,061	1,0000	1,009	100,000	118,073

ACB is the lowest cost join.