Name	CWID

Exam 1

October 18, 2018, 5:00pm - 6:15pm

CS525 - Fall 2018 - Midterm Exam Solutions

Please leave this empty! 1	2	3.1	3.2	Sum

Instructions

- Things that you are **not** allowed to use
  - Personal notes
  - Textbook
  - Printed lecture notes
  - Phone
- You are allowed to bring one page (both two sides can be used) of cheat sheet that must be turned in with the exam
- The exam is **75** minutes long
- For your convenience the number of points for each part and questions are shown in parenthesis.
- There are `3` parts in this exam (`85` points total)
  1. `Representing Data Elements` (15)
  2. `Relational Algebra and SQL` (20)
  3. `Index Structures` (50)

Part 1 Representing Data Elements (Total: 15 Points)

Question 1.1 Record Layout (15 Points)

Suppose blocks of 1024 bytes are used to store variable-length records. The fixed part of the block header is 24 bytes, and includes a count of the number of records. In addition, the header contains a 4-byte pointer (offset) to each record in the block. The first record is placed at the end of the block, the second record starts where the first one ends, and so on. The size of each record is not stored explicitly, since the size can be computed from the pointers.

1. How many records of size 50 bytes could we fully fit (no spanning) in such a block? (The block is designed to hold variable size records, but these records by coincidence happen to be all the same size.)

   Solution

   Number of records: $\left\lfloor \frac{1024-24}{50+4} \right\rfloor = 18$

2. Suppose we want to fully store 20 records in the block, again all of the same size. What is the maximum size of such records?

   Solution

   Maximum record size: $\lfloor \frac{1024-24-4*20}{20} \rfloor = 46$

3. Now assume the block layout is changed to accommodate only fixed size records. In this case, the block header is still 24 bytes and contains the common length of the records in the block. How many records of size 50 bytes could we fully fit in such a block? **Solution**

   Number of records: $\lfloor \frac{1024-24}{50} \rfloor = 20$

Part 2 Relational Algebra and SQL (Total: 20 Points)

You have been hired to work on a web site that maintains customer reviews of products. The main data is stored in the following tables:

- `Product`(<u>`pid`:**string**</u>, `pname`:**string**, `description`:**string**)
- `Reviewer`(<u>`rid`:**string**</u>, `rname`:**string**, `city`:**string**)
- `Review`(<u>`rid`:**string**, `pid`:**string**</u>,`rating`:**integer**, `comment`:**string**)

The tables contain the following information:

- `Product`: unique product id (`pid`), product name (`pname`), and product description. All strings.
- `Reviewer`: unique reviewer id (`rid`), and reviewer name (`rname`) and `city`, also all strings.
- `Review`: One entry for each individual review giving the reviewer and product ids, an integer `rating` in the range `0-5`, and the reviewer comment, which is a string.

Write the following queries in `SQL`. No duplicates should be printed in any of the answers.

Question 2.1 (5 Points)

Return the ratings and comments from all reviews written by reviewers in Chicago for products named 'ABC'. The results should be sorted in descending order by rating.

Solution

```
SELECT r.rating, r.comment
FROM Product p, Review r, Reviewer r1,
WHERE p.pname ='ABC' AND r1.city = 'Chicago' AND p.pid = r.pid AND r1.rid = r.rid
ORDER BY r.rating DESC
```

Question 2.2 (5 Points)

Return the number of reviewers in each distinct city. The results should list the city name and the number of reviewers in that city, and should be sorted alphabetically by city name.

Solution

```
SELECT r.city, COUNT(r.rid)
FROM Reviewer r
GROUP BY r.city
ORDER BY r.city
```

Question 2.3 (5 Points)

Return the names of all grumpy reviewers. A grumpy reviewer is one whose average review rating is less than or equal to 2. If someone is in the Reviewer table but has no reviews in the Review table, they should not be included in the result. The reviewer names in the result can be in any order.

Solution

```
SELECT rr.rname
FROM Reviewer rr, Review r
WHERE rr.rid = r.rid                    -- Reviewers with no reviews are excluded from join
GROUP BY r.rid, rr.rid, rr.rname
HAVING AVG(r.rating) <= 2
```

Question 2.4 (5 Points)

Return the names and descriptions of all cool products. A "cool product" is one that has no reviews in the database with any rating less than 4. A product must have at least one review in the database to be considered as a possible "cool product". The results can be in any order.

Solution

```
SELECT p.pname, p.description
FROM Product p, Review r
WHERE p.pid = r.pid
GROUP BY p.pid, p.pname, p.description
HAVING MIN(r.rating) >= 4
```

Part 3 Index Structures (Total: 50 Points)

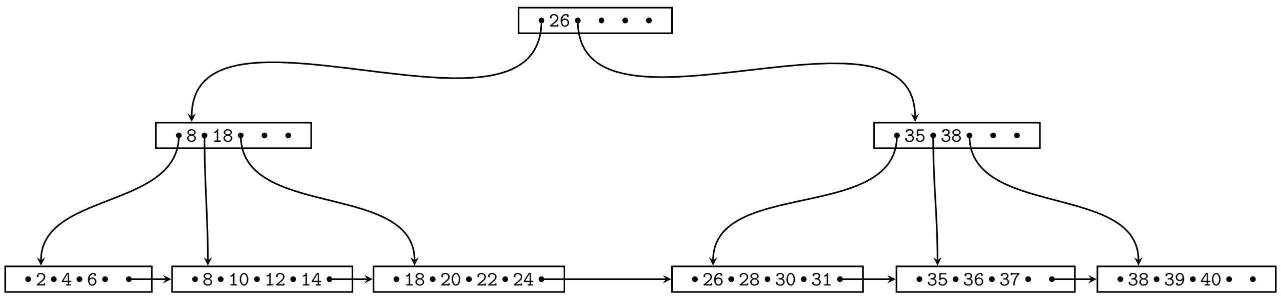
Question 3.1 B+-tree Operations (25 Points)

Given is the B+-tree shown below ($n = 4$). Execute the following operations and write down the resulting B+-tree after each step:

insert(1), insert(33), insert(17), delete(40), delete(26)

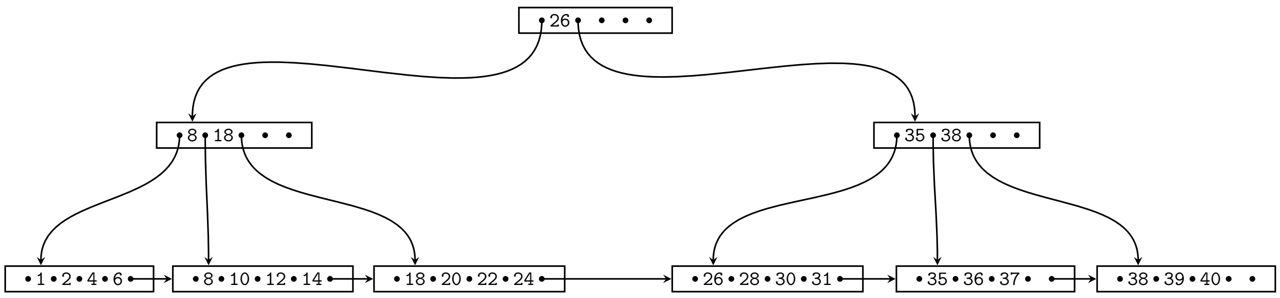
When splitting or merging nodes follow these conventions:

- **Leaf Split**: In case a leaf node needs to be split, the left node should get the extra key if the keys cannot be split evenly.
- **Non-Leaf Split**: In case a non-leaf node is split evenly, the "middle" value should be taken from the right node.
- **Node Underflow**: In case of a node underflow you should first try to redistribute and only if this fails merge. Both approaches should prefer the left sibling.

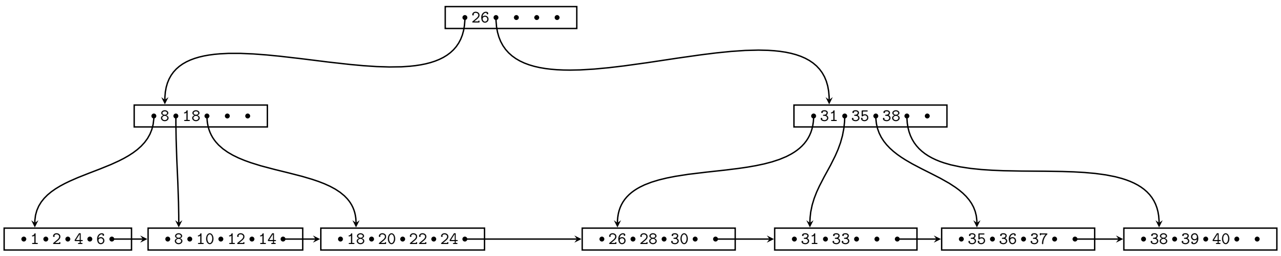


Solution

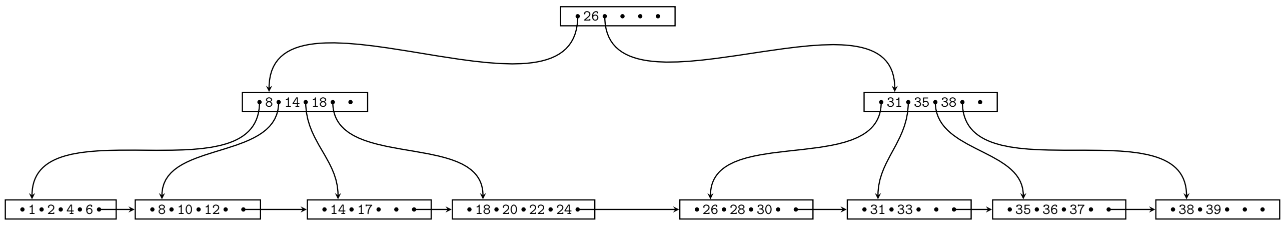
insert(1)



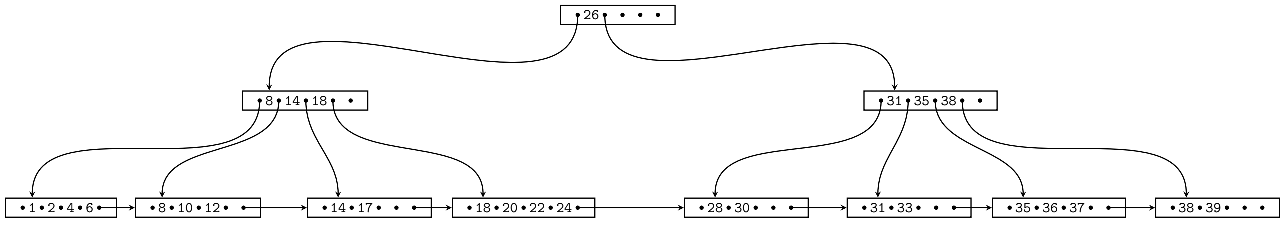
insert(33)



insert(17), delete(40)



delete(26)



Question 3.2 Extensible Hashing (25 Points)

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

`288, 8, 1064, 120, 148, 204, 641, 700, 258, 1586, 44, 68, 48, 575`

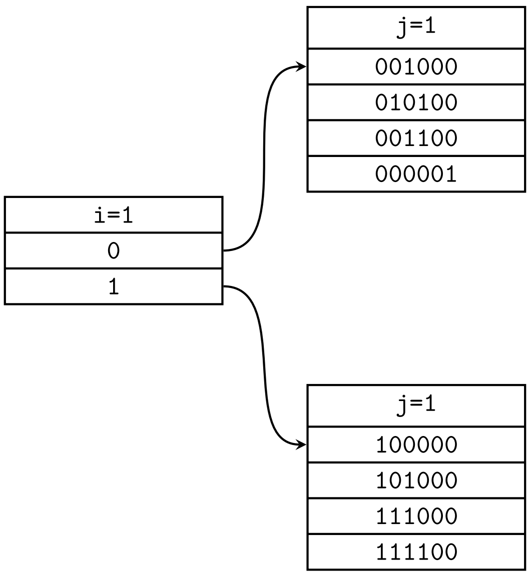
Show the `Extensible hash` structure for this file if the hash function is defined as below and buckets can hold `4` records each.

x	$h(x)$
288	100000
8	001000
1064	101000
120	111000
148	010100
204	001100
641	000001
700	111100
258	000010
1586	110010
44	101100
68	000100
48	110000
575	111111

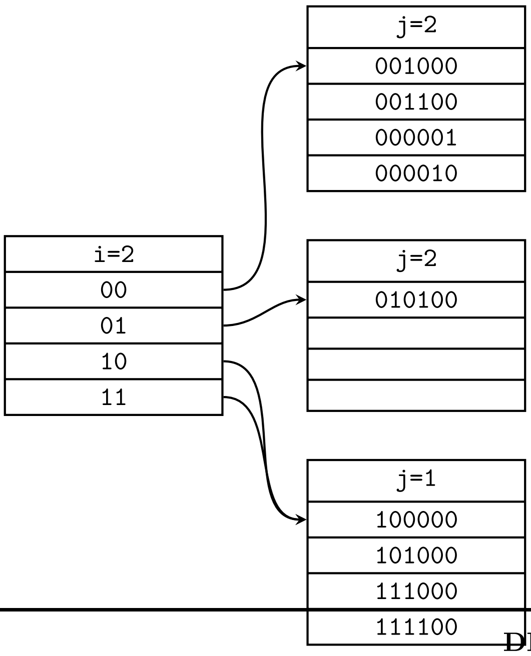
Solution

x	h(x)
288	100000
8	001000
1064	101000
120	111000
148	010100
204	001100
641	000001
700	111100
258	000010
1586	110010
44	101100
68	000100
48	110000
575	111111

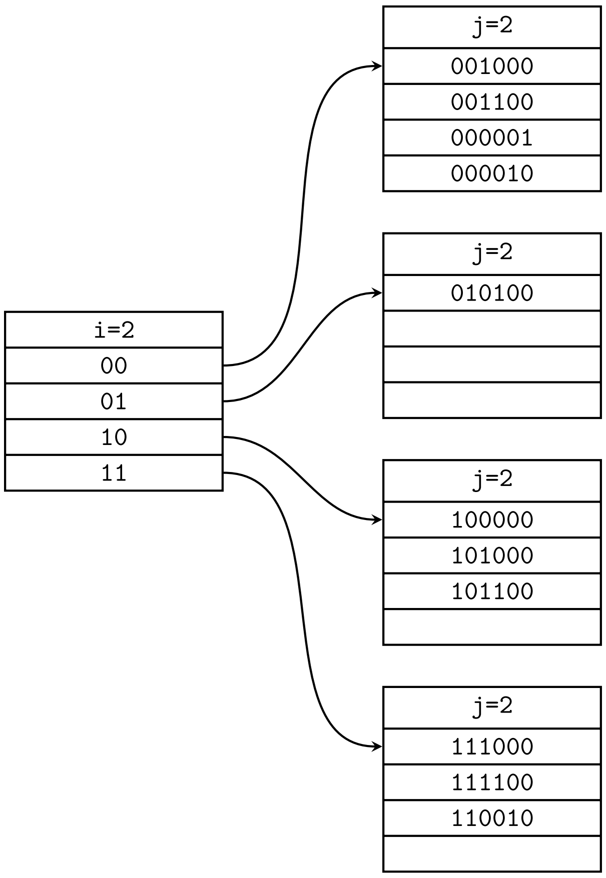
insert(288), insert(8), insert(1064), insert(120),insert(148), ,insert(204), ,insert(641), insert(700)



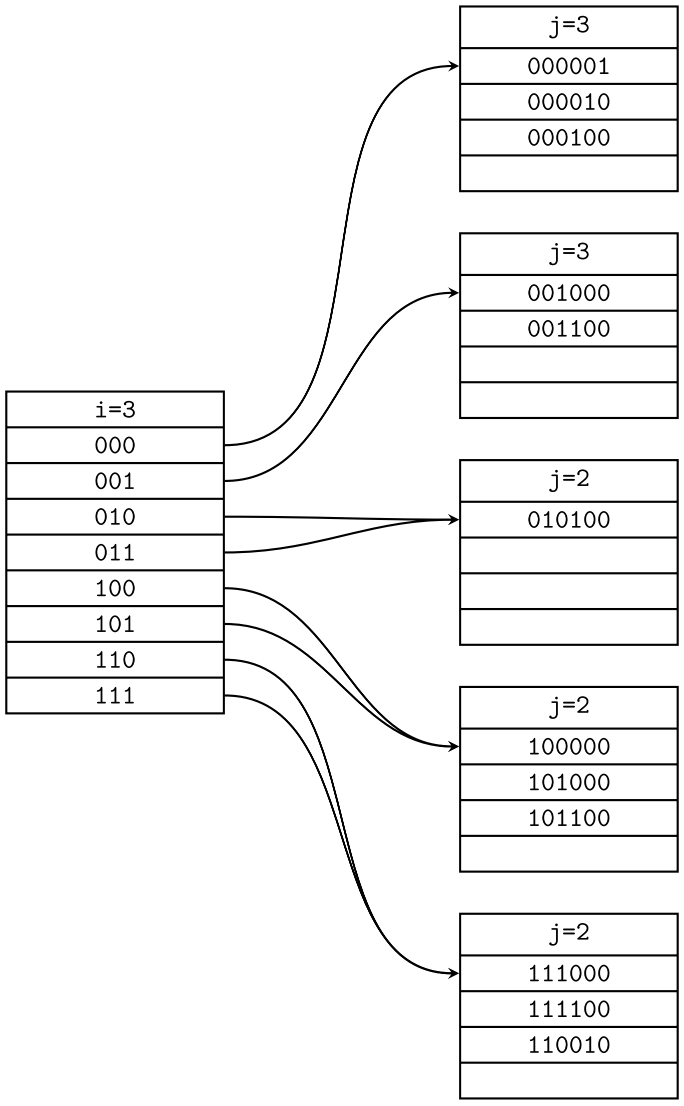
insert(258 [000010])



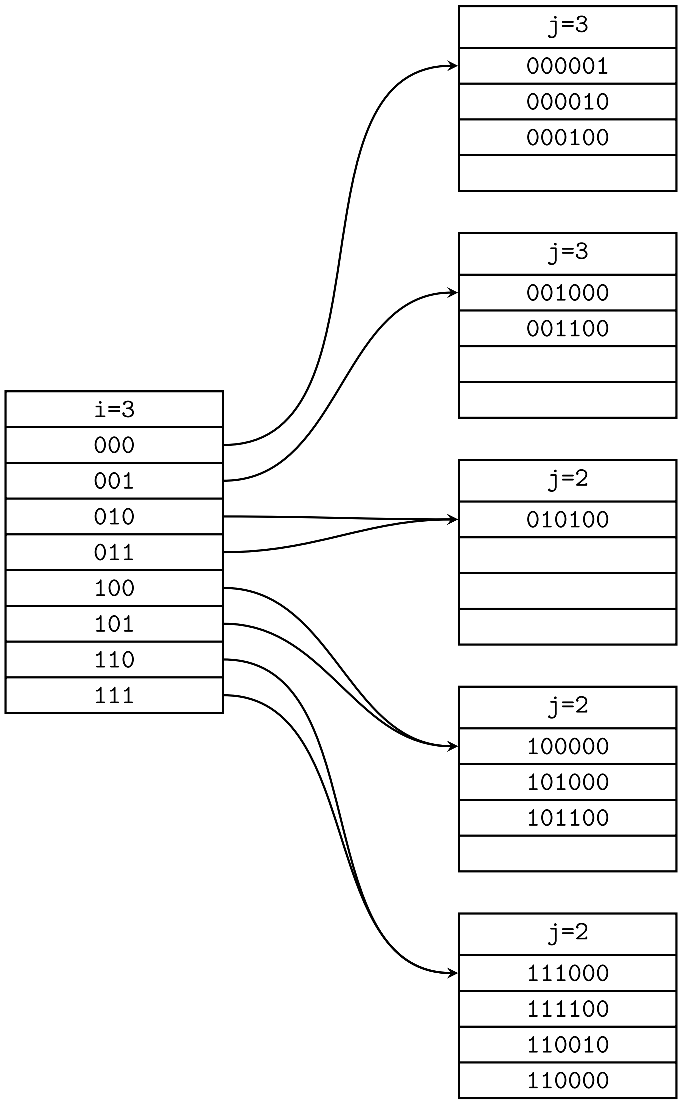
insert(1586 [110010]), insert(44 [101100])



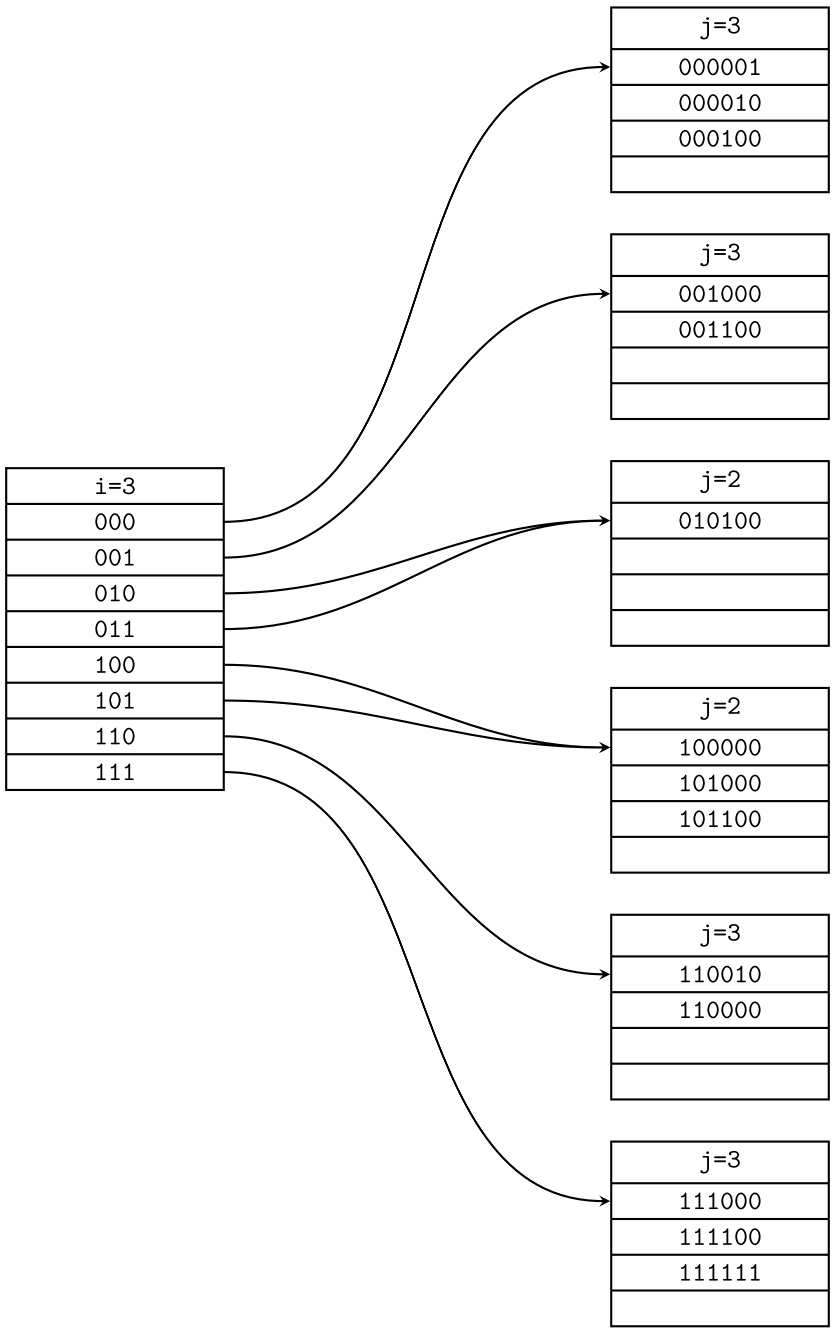
insert(68 [000100])



insert(48 [110000])



insert(575 [111111])



This page left blank intentionally. There are no more questions.

This page left blank intentionally. There are no more questions.