Name

CWID

Exam
1

March 6th, 2023

CS525 - Midterm Exam
Solutions

Please leave this empty! 1 2 3 4 Sum

Instructions

- Things that you are **not** allowed to use
  - Personal notes
  - Textbook
  - Printed lecture notes
  - Phone
- The exam is **75** minutes long
- Multiple choice questions are graded in the following way: You get points for correct answers and points subtracted for wrong answers. The minimum points for each questions is **0**. For example, assume there is a multiple choice question with 6 answers - each may be correct or incorrect - and each answer gives 1 point. If you answer 3 questions correct and 3 incorrect you get 0 points. If you answer 4 questions correct and 2 incorrect you get 2 points. ...
- For your convenience the number of points for each part and questions are shown in parenthesis.
- There are 4 parts in this exam
  1. SQL
  2. Relational Algebra
  3. Index Structures
  4. Result Size Estimation

Part 1 SQL (Total: 32 Points)

Consider the following zoo database schema and instance:

animal

nickname	species	weightlb	age	gender
Dumbo	Elephant	6300	45	m
Pum	Elephant	7900	15	f
Flippy	Dolphin	5000	23	m
Birdy	Owl	5	4	m
Lea	Lion	240	7	f

cage

id	size_sq	allowed_weight	aquatic
1	30000	50000	false
2	500	100	true
3	20000	100	false

eats

species	food	amount	unitprice
Elephant	grass	20000	0.1
Dolphin	mackerels	20	5.0
Owl	mice	1	3.0
Lion	meat	200	10.0

livesin

animal	cage
Dumbo	1
Pum	1
Flippy	2
Birdy	1
Lea	3

Hints:

- When writing queries do only take the schema into account and **not** the example data given here. That is your queries should return correct results for all potential instances of this schema.
- Attributes with black background form the primary key of an relation. For example, **animal,cage** is the primary key of relation **livesin**.
- The attribute **animal** of relation **livesin** is a foreign key to relation **animal**.
- The attribute **cage** of relation **livesin** is a foreign key to relation **cage**.

Question 1.1 (6 Points)

Write an SQL query that returns the id of cages with at least one male elephant and one female Elephant.

Solution

```
WITH lsg AS (
  SELECT c.id, species, gender
    FROM cage c, livesin l, animal a
   WHERE c.id = l.cage AND l.animal = a.nickname)
SELECT id FROM lsg WHERE species = 'Elephant' and gender = 'm'
INTERSECT
SELECT id FROM lsg WHERE species = 'Elephant' and gender = 'f';
```

Question 1.2 (8 Points)

Write a query that returns species where the total weight of male members of this species is higher than the total weight of female members of this species

Solution

```
SELECT species
FROM (SELECT species,
             sum(CASE WHEN gender = 'm' THEN weightlb ELSE 0 END) AS wmales,
             sum(CASE WHEN gender = 'f' THEN weightlb ELSE 0 END) AS wfemales
      FROM animal
       GROUP BY species) s
WHERE wmales > wfemales;
```

Question 1.3 (9 Points)

Write a query that computes the number of cages with a certain number of inhabitants. From this set we then want to return the number of inhabitants where there are the most cages with this many inhabitants. For instance, in the example database there is 1 cage with 3 inhabitants and there are 2 cages with 1 inhabitants. Thus, the later should be returned.

Solution

```
WITH numcages AS (
   SELECT inh, count(*) AS numcages
   FROM (SELECT cage, count(*) AS inh
         FROM livesin l
         GROUP BY cage) numin
   GROUP BY inh)
SELECT inh, numcages
FROM numcages
WHERE numcages = (SELECT max(numcages) FROM numcages);
```

Question 1.4 (9 Points)

Write an SQL query that returns species where all members of the species live in cages which host only members of this one species. For instance, in the example databases the query would return *dolphins* and *lions* as the only members of these species live in a cage by themselves.

Solution

```
WITH speciescage AS (
  SELECT species, cage
    FROM animal a, livesin l
   WHERE a.nickname = l.animal)

SELECT species
  FROM speciescage c
 WHERE NOT EXISTS (SELECT *
                     FROM speciescage c1,
                          speciescage c2
                    WHERE c1.cage = c2. cage
                      AND c1.species != c2. species
                      AND c1.species = c.species);
```

Part 2 Relational Algebra (Total: 26 Points)

Question 2.1 Relational Algebra (6 Points)

Write a relational algebra expression over the schema from the SQL part that returns the ids of empty cages (cages with no animals living in these cages). Use the **bag semantics** version of relational algebra.

Solution

$$Q = \pi_{id}(cages) - \pi_{cage}(cages)$$

Question 2.2 Relational Algebra (8 Points)

Write a relational algebra expression over the schema from the SQL part that returns the `nickname` and `weightlb` of animals that weight more than the average weight for their species. Use the **bag semantics** version of relational algebra.

Solution

$$Q_{avg} = \gamma_{avg(weightlb);species}(animal)$$
$$Q = \pi_{nickname,weightlb}(\sigma_{weightlb>avg(weightlb)}(animal \bowtie Q_{avg}))$$

Question 2.3 Relational Algebra (12 Points)

Write a relational algebra expression over the schema from the SQL part that returns the name of the animal species that the zoo is spending the most money on to feed all members of this species. The cost of food for a single animal of a species is the *amount* of food they need times the *unitprice* for this type of food (this information is stored in the `eats` table). Use the **bag semantics** version of relational algebra.

Solution

$$
\begin{aligned}
Q_{percost} &= \pi_{species, amount \cdot unitprice \to cost}(animal \bowtie eats) \\
Q_{perspecies} &= \gamma_{sum(cost) \to ttl; species}(Q_{percost}) \\
Q_{max} &= \gamma_{max(ttl) \to mx;}(Q_{perspecies}) \\
Q &= \pi_{species}(Q_{perspecies} \bowtie_{ttl=mx} Q_{max})
\end{aligned}
$$

Part 3 Index Structures (Total: 24 Points)

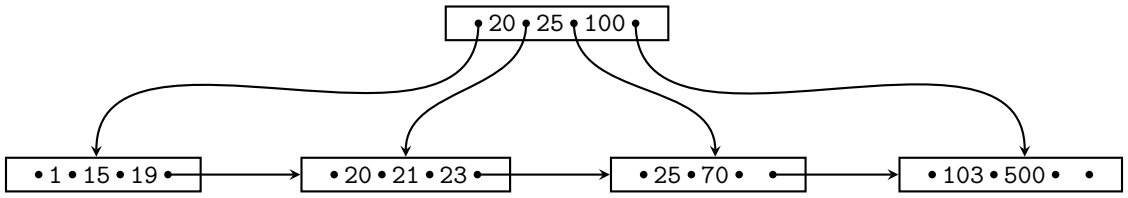
Question 3.1 B+-tree Operations (24 Points)

Given is the B+-tree shown below ($n = 3$). Execute the following operations and write down the resulting B+-tree after each step:

insert(88),insert(2),insert(24),delete(103)

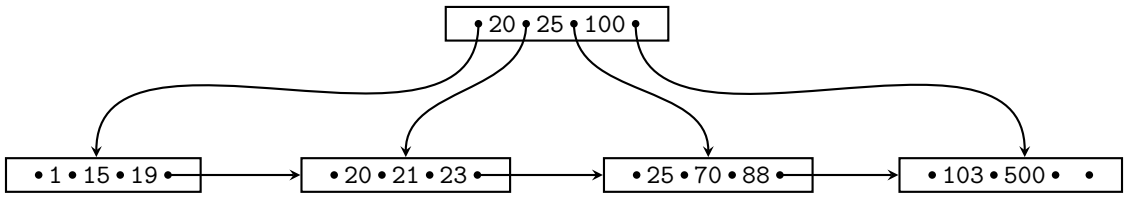
When splitting or merging nodes follow these conventions:

- **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.

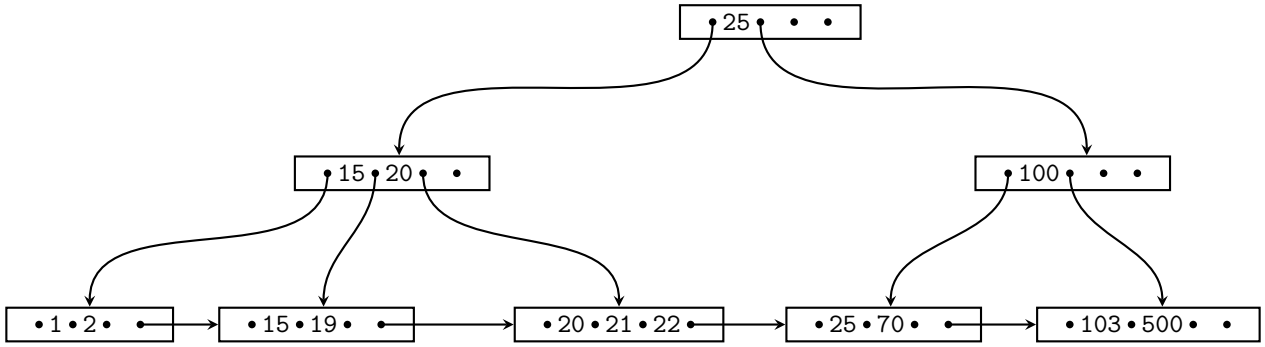


Solution

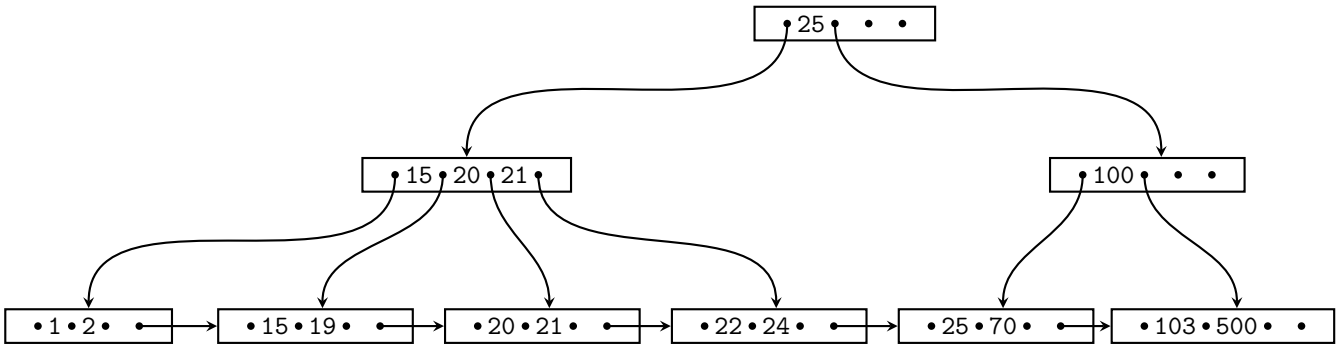
insert(88)



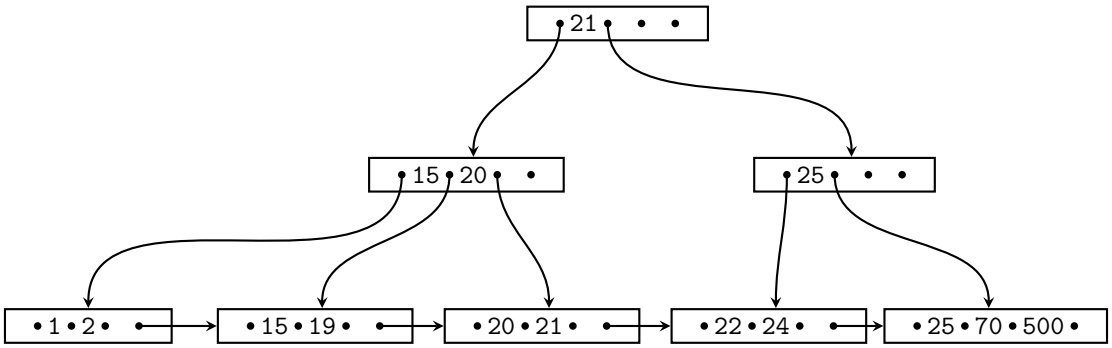
insert(2)



insert(24)



delete(103)



Part 4 Extensible Hashing (Total: 18 Points)

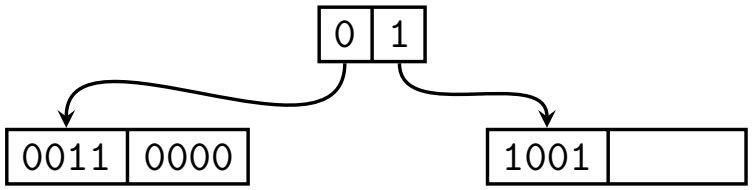
Question 4.1 Extensible Hashing (18 Points)

Consider the extensible Hash index shown below that is the result of inserting values 4,2,7. Each page holds two keys. Execute the following operations

`insert(6),insert(8),insert(5),insert(12)`

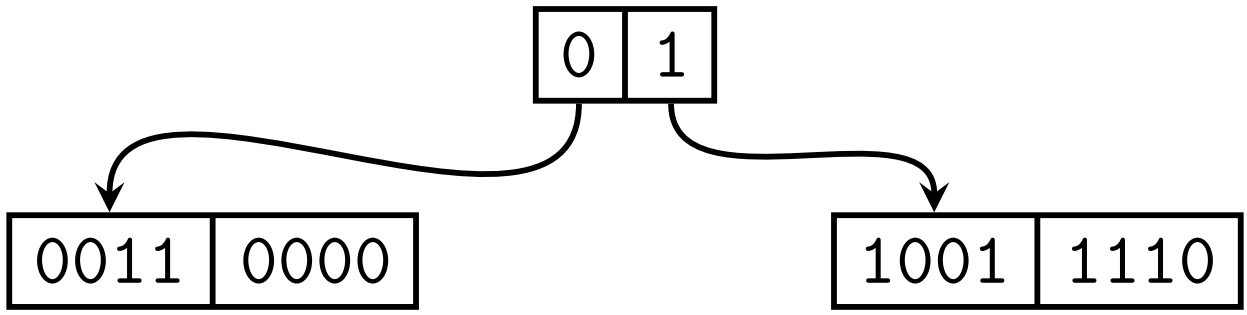
and write down the resulting index after each operation. Assume the hash function is defined as:

x	h(x)
0	1000
1	0101
2	0000
3	0101
4	0011
5	1110
6	1110
7	1001
8	1100
9	1001
10	1111
11	0010
12	0101

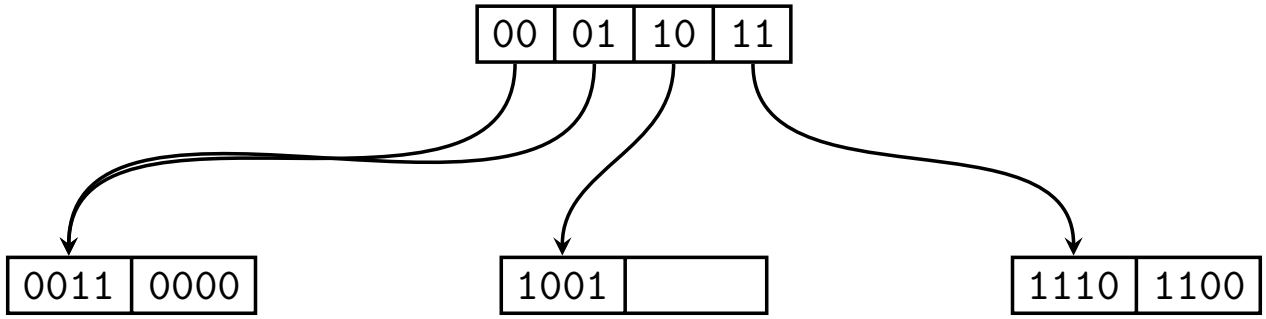


Solution

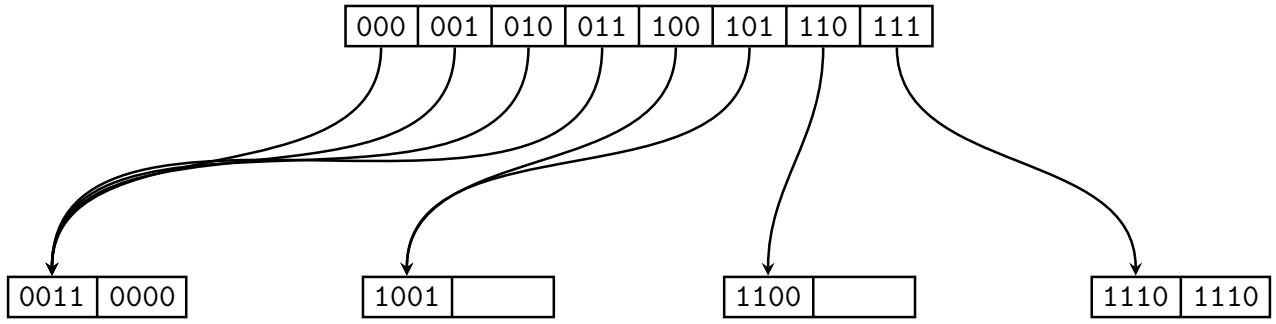
insert(6)



insert(8)



insert(5)



insert(12)