>>

Week 5 Exercises

❖ Things to Note

- More Exercises now available
- Assignment 1 due before 9pm Friday
- Assignment 2 available next week
- Next week is Flexibility Week
  - no new Videos+Slides
  - no online sessions
  - but Quiz 3 will run

<< ^ >>

❖ Exercise: Index Storage Overheads

Consider a relation with the following storage parameters:

- *B = 8192, R = 128, r = 100000*
- header in data pages: 256 bytes
- key is integer, data file is sorted on key
- index entries (keyVal,tupleID): 8 bytes
- header in index pages: 32 bytes

How many pages are needed to hold a dense index?

How many pages are needed to hold a sparse index?

❖ Exercise: One vs Multiple Indices

Consider a relation with r = 100,000, B = 4096, defined as:

```
create table Students (
    id       integer primary key,
    name     char(10), -- simplified
    gender   char(1),  -- 'm','f','?'
    birthday char(5)   -- 'MM-DD'
);
```

Assumptions:

- data file is not ordered on any attribute
- has a dense B-tree index on each attribute
- 96 bytes of header in each data/index page

<< ∧ >>

❖ Exercise: One vs Multiple Indices (cont)

For **`Students(id,name,gender,birthday)`** ...

- calculate the size of the data file and each index
- describe the selectivity of each attribute

Now consider a query on this relation:

```
select * from Students
where  name='John' and birthday='04-01'
```

- estimate the cost of answering using **`name`** index
- estimate the cost of answering using **`birthday`** index
- estimate the cost of answering using both indices

<< ^ >>

❖ Bitmap Indexes

Alternative index structure, focussing on sets of tuples:

Data File

	Part#	Colour	Price
[0]	P7	red	$2.50
[1]	P1	green	$3.50
[2]	P9	blue	$4.10
[3]	P4	blue	$7.00
[4]	P5	red	$5.20
[5]	P5	red	$2.50

.....

Colour Index

red	100011...
blue	001100...
green	010000...

Price Index

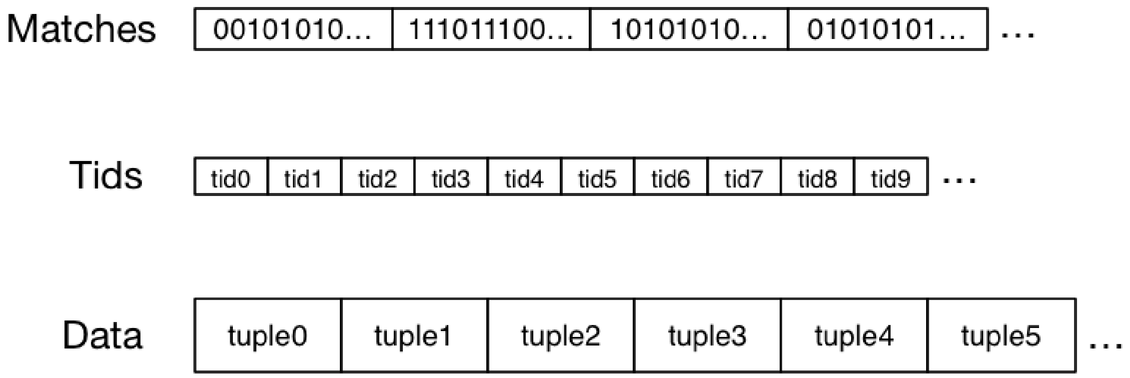
< $4.00	110001...
>= $4.00	001110...

Index contains bit-strings of *r* bits, one for each value/range

<< ^ >>

❖ Bitmap Indexes (cont)

Also useful to have a file of **`tid`**s, one for each tuple:



COMP9315 21T1 ♢ Week 5 Exercises ♢ [6/19]

<< ^ >>

❖ Bitmap Indexes (cont)

Answering queries using bitmap index:

```
Matches = AllOnes(r)
foreach attribute A with index {
   // select ith bit-string for attribute A
   // based on value associated with A in WHERE
   Matches = Matches & Bitmaps[A][i]
}
// Matches contains 1-bit for each matching tuple
foreach i in 0..r-1 {
   if (Matches[i] == 0) continue;
   Pages = Pages ∪ {pageOf(Tids[i])}
}
foreach pid in Pages {
   P = getPage(pid)
   extract matching tuples from P
}
```

<< ^ >>

❖ Exercise: Bitmap Index

Using the following file structure:

Data File

	Part#	Colour	Price
[0]	P7	red	$2.50
[1]	P1	green	$3.50
[2]	P9	blue	$4.10
[3]	P4	blue	$7.00
[4]	P5	red	$5.20
[5]	P5	red	$2.50

.....

Colour Index

red	100011...
blue	001100...
green	010000...

Price Index

< $4.00	110001...
>= $4.00	001110...

Show how the following queries would be answered:

```
select * from Parts
where colour='red' and price < 4.00

select * from Parts
where colour='green' or colour ='blue'
```

<< ^ >>

❖ Exercise: Bitmap Index (cont)

Storage costs for bitmap indexes:

- one bitmap for each value/range for each indexed attribute
- each bitmap has length *ceil(r/8)* bytes
- e.g. with 50K records and 8KB pages, bitmap fits in one page

Query execution costs for bitmap indexes:

- read one bitmap for each indexed attribute in query
- perform bitwise AND on bitmaps (in memory)
- read pages containing matching tuples

Note: bitmaps could index pages rather than tuples (shorter bitmaps)

❖ Exercise: Insertion into B-trees

Assumptions

- B-tree is initially empty
- each node in the B-tree is a page with $c_i = 5$
- key values are unique (i.e. primary key)
- each leaf node has a pointer to its right sibling
- tuples are added to the data file in order of insertion

Show how the tree grows as we add the following key values

- 100, 50, 80, 200, 20, 65, 150, 120, 110, 75, 10, 180, ...

<< ^ >>

❖ Exercise: Multi-attribute Hashing

Compute the hash value for the tuple

```
('John Smith','BSc(CompSci)',1990,99.5)
```

where $d=6$, $d_1=3$, $d_2=2$, $d_3=1$, and

- cv = <(1,0), (1,1), (2,0), (3,0), (1,2), (2,1), (3,1), (1,3), ...>
- $hash_1$(**'John Smith'**) = **...0101010110110100**
- $hash_2$(**'BSc(CompSci)'**) = **...1011111101101111**
- $hash_3$(**1990**) = **...0001001011000000**

❖ Exercise: Partial hash values in MAH

Given the following:

- d=6, b=2^6, CV = <(0,0),(0,1),(1,0),(2,0),(1,1),(0,2), ...>
- hash (a) = ...00101101001101
- hash (b) = ...00101101001101
- hash (c) = ...00101101001101

What are the query hashes for each of the following:

- (a,b,c), (?,b,c), (a,?,?), (?,?,?)

<< ^ >>

❖ Exercise: Representing Stars

Our hash values are bit-strings (e.g. **`100101110101`**)

MA.Hashing introduces a third value (`*` = unknown)

How could we represent "bit"-strings like **`1011*1*0**010`**?

❖ Exercise: MA.Hashing Query Cost

Consider **`R(x,y,z)`** using multi-attribute hashing where

$d = 9 \quad d_x = 5 \quad d_y = 3 \quad d_z = 1$

How many buckets are accessed in answering each query?

1. **`select * from R where x = 4 and y = 2 and z = 1`**
2. **`select * from R where x = 5 and y = 3`**
3. **`select * from R where y = 99`**
4. **`select * from R where z = 23`**
5. **`select * from R where x > 5`**

<< ^ >>

❖ Exercise: MA.Hashing Design

Consider relation **`Person(name,gender,age)`** ...

p_Q	Query Type Q
0.6	**`select age,gender from Person where name=X`**
0.3	**`select count(*) from Person where gender=X`**
0.1	**`select name from Person where gender=X and age=Y`**

Assume that all other query types have $p_Q=0$.

Design a choice vector to minimise average selection cost.

<< ^ >>

❖ Exercise: Query Types and Tuple Space

Which part of the tuple-space does each query represent?

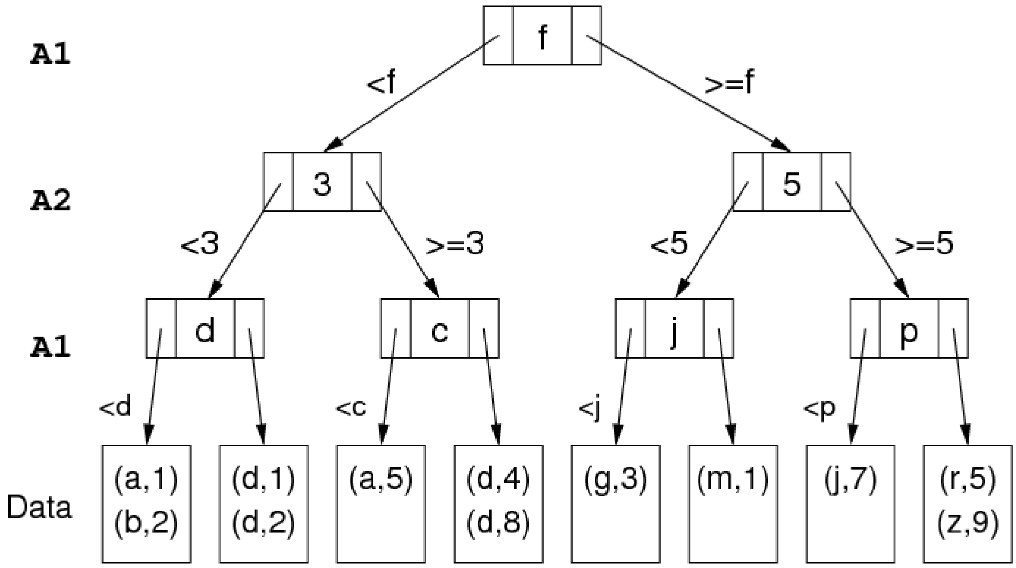
```
Q1: select * from Rel where X = 'd' and Y = 4
Q2: select * from Rel where 'j' < X ≤ 'r'
Q3: select * from Rel where X > 'm' and Y > 4
Q4: select * from Rel where 'k' ≤ X ≤ 'p' and 3 ≤ Y ≤ 6
```

```
   a b c d e f g h i j k l m n o p q r s t u v w x y z
 0
 1 *     *                 *
 2   *   *
 3             *
 4       *
 5 *                                 *
 6
 7                   *
 8       *
 9                                                   *
```

<< ∧ >>

❖ Exercise: Searching in kd-Trees

Using the following kd-tree index

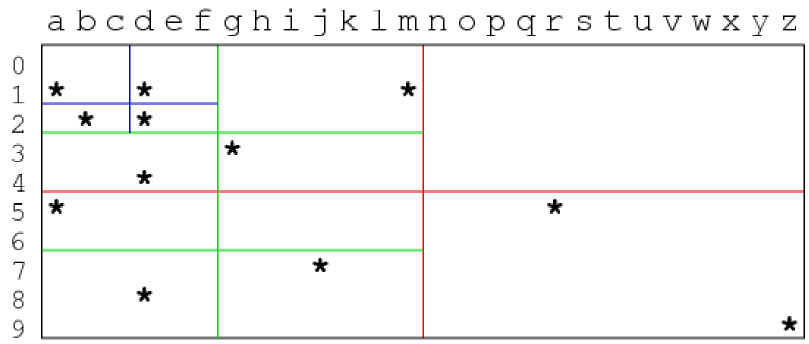


Answer the queries **`(m,1)`**, **`(a,?)`**, **`(?,1)`**, **`(?,?)`**

<< ^ >>

❖ Exercise: Searching in Quad-trees

Using the following quad-tree index

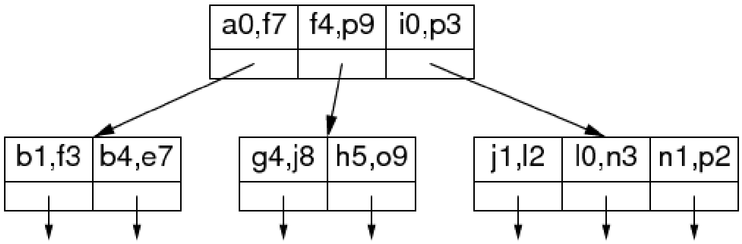


Answer the queries **`(m,1)`**, **`(a,?)`**, **`(?,1)`**, **`(?,?)`**

<< ^

❖ Exercise: Query with R-trees

Using the following R-tree:



Show how the following queries would be answered:

```
Q1: select * from Rel where X='a' and Y=4
Q2: select * from Rel where X='i' and Y=6
Q3: select * from Rel where 'c'≤X≤'j' and Y=5
Q4: select * from Rel where X='c'
```

Note: can view unknown value **X=?** as range *min(***X***) ≤* **X** *≤ max(***X***)*

Produced: 16 Mar 2021