COL362 / COL632 Major

Divyanshu Agarwal

TOTAL POINTS

33.5 / 45

QUESTION 1

A 30 pts

1.1 A.1 **0 / 2**

+ 2 pts Correct answer

- 1 pts Incorrect

✓ **+ 0 pts** *Not Attempted*

1.2 A.2 **0 / 2**

+ 2 pts Correct

- 1 pts Incorrect

✓ **+ 0 pts** *Not Attempted*

1.3 A.3 **2 / 2**

✓ **+ 2 pts** *Correct*

- 1 pts Incorrect

+ 0 pts Not Attempted

1.4 A.4 **2 / 2**

✓ **+ 2 pts** *Correct*

- 1 pts Incorrect

+ 0 pts Not Attempted

1.5 A.5 **2 / 2**

✓ **+ 2 pts** *Correct*

- 1 pts Incorrect

+ 0 pts Not Attempted

1.6 A.6 **2 / 2**

✓ **+ 2 pts** *Correct*

- 1 pts Incorrect

+ 0 pts Not Attempted

1.7 A.7 **0 / 2**

+ 2 pts Correct

- 1 pts Incorrect

✓ **+ 0 pts** *Not Attempted*

1.8 A.8 **2 / 2**

✓ **+ 2 pts** *Correct (A)*

- 1 pts Incorrect

+ 0 pts Not Attempted

1.9 A.9 **2 / 2**

✓ **+ 2 pts** *Correct*

1.10 A.10 **2 / 2**

✓ **+ 2 pts** *Correct*

+ 0 pts Not Attempted

- 1 pts Incorrect

1.11 A.11 **0 / 2**

+ 2 pts Correct

✓ **+ 0 pts** *Not Attempted*

- 1 pts Incorrect

1.12 A.12 **2 / 2**

✓ **+ 2 pts** *Correct*

+ **0 pts** Not Attempted

- **1 pts** Incorrect

1.13 A.13 **0 / 2**

+ **2 pts** Correct

✓ + **0 pts** *Not Attempted*

- **1 pts** Incorrect

1.14 A.14 **2 / 2**

✓ + **2 pts** *Correct*

+ **0 pts** Not Attempted

- **1 pts** Incorrect

1.15 A.15 **2 / 2**

✓ + **2 pts** *Correct*

+ **0 pts** Not Attempted

- **1 pts** Incorrect

QUESTION 2

2 B.16 **3 / 3**

✓ + **1 pts** *Self Join citetation table*

✓ + **2 pts** *Correct condition for join and final Group by*

- **0.5 pts** Minor syntax error in SQL

+ **0 pts** Incorrect/ Not Attempted

QUESTION 3

3 B.17 **2.5 / 3**

✓ + **1 pts** *Correct number of leaf nodes access (or it's IO access)*

✓ + **1 pts** *Block/pointer access to read records*

✓ + **1 pts** *Final answer*

✓ - **0.5 pts** *Minor calculation mistake*

+ **0 pts** Incorrect/ Not Attempted

QUESTION 4

4 B.18 **3 / 3**

✓ + **1 pts** *Correct Key*

✓ + **2 pts** *Justification of answer*

+ **1 pts** Incorrect ans but partially correct steps

+ **0 pts** Incorrect/ Not Attempted

QUESTION 5

B.19 6 pts

5.1 19.A **3 / 3**

✓ + **1.5 pts** *Correct pseudo-code of Map*

✓ + **1.5 pts** *Correct pseudo-code of Reduce*

+ **0 pts** Incorrect/ Not Attempted

5.2 19.B **2 / 3**

✓ + **1 pts** *Correct map for list of names*

✓ + **1 pts** *Correct map for list of diseases*

+ **1 pts** Correct reduce to combine them

+ **0 pts** Incorrect/ Not Attempted

1 How will this reducer merge them?

COL 362 / COL 632 - Major Examination

06th May 2023

Exam time: 05:00 IST – 6:30 IST

Duration: 1.5 hours **Total Points: 45**

Answer all questions.

This is **NOT** an open-book exam and **NOT** a collaborative exam. You are expected not to seek help from any other person(s) directly or indirectly (through web, forums, emails, messages etc.) for answering these questions.

Honor Code

I acknowledge the IITD Honor Code and confirm that the answers I have written for this exam are entirely my own.

1. I have **not** used the help of any person, organization or discussions during the examination in answering.
2. I have **not** given or received assistance in either answering, or providing specimen answer/hint/diagram/model/code during the exam.

I am fully aware that if found to have used unfair and disallowed practices in the examination, I am liable for strict action, which could include receiving a failing grade in this course, as well as other actions as deemed fit by the institute disciplinary committee.

Name: DIVYANSHU AGARWAL

Entrynumber: 2020CS10343

Signature: Divyanshu

Date: 06 May 2023

Instructions

- **Please make sure you have read the honor code in the front page, and accepted it by filling all the details.**
- **Please make sure you have no reference material, mobile phone, laptop, or any other device that could be construed to help you to answer this exam**
- Questions are made as unambiguous as possible. In case you need, make appropriate and meaningful assumptions and state them clearly (only if required).
- **Answer only within the space given below each question.** If you need correct an answer, then you must neatly cross-out the incorrect answer you have filled, and write it separately at the end (write the question number correctly).

Questions

Part A: MCQs with Negative Marking

- Write your answers in the blank line given after each question. Just write the choices (e.g., A,B,C...etc.) of correct answers.
- All questions carry equal marks (2 marks each), and there will be no partial marking.
- If there are multiple correct choices, answer is considered correct only if all and only correct choices are selected.
- Incorrect answer gets -1 marks.

1. (2 points) If we take into consideration the interesting sort order in a query plan, the number of left-deep plans to consider be equal to linear to the number of relations in the query - i.e., $O(n)$. Is this statement

1. _____

2. (2 points) Which of the following is **true** in the context of pipelined plan implementation in database systems:

   A. Demand-driven pipelining requires an iterator to remember the state between calls to the same operator

   B. In producer-driven pipelined execution implementation all operators are executing concurrently

   C. In demand-driven pipelined execution operators can execute in parallel

   D. Detailed state information needs to be maintained between function calls in producer-driven pipelining

   E. None of the above

2. _____

3. (2 points) Given 4 transactions T_1, T_2, T_3, T_4, how many serial schedules are possible?

   A. 4

   B. 16

   C. 24

   D. 8

3. C

4. (2 points) Assuming that R and S are sets and $|R| = r$ and $|S| = s$, which of the following statements is *true*?

   A. $\sigma_C(R \bowtie S)$ has a minimum of r tuples if condition C applies only to R.

   B. $R \bowtie_{R.A=S.A} S$ has exactly r tuples if R.A contains r unique values.

   C. $R \bowtie S$ has a maximum of rs tuples.

   D. $\pi_{R.A}(R \bowtie_{R.A=S.A} S)$ and $\pi_{S.A}(R \bowtie_{R.A=S.A} S)$ have exactly the same number of tuples.

4. CD

5. (2 points) Consider the following schedule with 4 transactions T_1,T_2,T_3 and T_4 such that each transaction T_i has timestamp i ($1 \leq i \leq 4$). The concurrency manager is using Thomas' write rule for ensuring isolation.

T_1	T_2	T_3	T_4
read(A)			
write(B)			
		read(A)	
	read(A)		
			read(B)
			write(C)
	write(A)		
read(C)			
write(C)			

T2 → rollback

A 13 W
B 1 4
C

Which of the following transaction(s) will be rolled back?

A. Only T_1.

B. Only T_1 and T_2.

C. Only T_2.

D. Only T_3 and T_1 X

5. B

6. (2 points) Suppose we have the following table definition (along with indexes, constraints, etc.):

```
       Table "public.yagofacts"
 Column   |       Type        | Modifiers
----------+-------------------+-----------
 id       | character varying |
 subject  | character varying |
 predicate | character varying |
 object   | character varying |
 value    | double precision  |
Indexes:
   "yagoindexid" btree (id)
   "yagoindexobject" btree (object)
   "yagoindexobjectpredicate" btree (object, predicate)
   "yagoindexpredicate" btree (predicate)
   "yagoindexsubject" btree (subject)
   "yagoindexsubjectpredicate" btree (subject, predicate)
   "yagoindexvalue" btree (value)
   "yagoindexvaluepredicate" btree (value, predicate)
```

Suppose we have the following three query plans.

QUERY PLAN 1

```
Aggregate  (cost=12037698.98..12037698.99 rows=1 width=8)
  -> GroupAggregate  (cost=11750486.83..12036482.16 rows=97346 width=41)
       -> Sort  (cost=11750486.83..11845494.12 rows=38002915 width=41)
            Sort Key: yagofacts.subject
            -> Seq Scan on yagofacts  (cost=0.00..1120819.56 rows=38002915 width=41)
                 Filter: (((subject)::text <> (object)::text) AND ((predicate)::text = '<linksTo>'::text))
```

QUERY PLAN 2

```
Aggregate  (cost=12871054.15..12871054.16 rows=1 width=8)
  ->  GroupAggregate  (cost=12583842.00..12869837.33 rows=97346 width=41)
        ->  Sort  (cost=12583842.00..12678849.29 rows=38002915 width=41)
              Sort Key: yagofacts.subject
              ->  Bitmap Heap Scan on yagofacts  (cost=917778.45..1954174.73 rows=38002915 width=41)
                    Recheck Cond: ((predicate)::text = '<linksTo>'::text)
                    Filter: ((subject)::text <> (object)::text)
                    ->  Bitmap Index Scan on yagoindexpredicate  (cost=0.00..908277.72 rows=38193885 width=0)
                          Index Cond: ((predicate)::text = '<linksTo>'::text)
```

QUERY PLAN 3

```
Aggregate  (cost=151643045.94..151643045.95 rows=1 width=8)
  ->  GroupAggregate  (cost=0.00..151641829.12 rows=97346 width=41)
        ->  Index Scan using yagoindexsubjectpredicate on yagofacts  (cost=0.00..151450841.08 rows=38002915 width=41)
              Index Cond: ((predicate)::text = '<linksTo>'::text)
              Filter: ((subject)::text <> (object)::text)
```

With respect to the following two queries, which of the options is true?

Query 1

```
SELECT avg (tab.cnt) FROM (
        SELECT count(subject) AS cnt FROM yagofacts WHERE
            subject != object AND predicate = '<linksTo>' GROUP BY object
) AS tab;
```

Query 2

```
SELECT avg (num\_out.c) FROM (
        SELECT count(object) AS c FROM yagoFacts WHERE
            predicate = '<linksTo>' and subject <> object GROUP BY subject
) AS num\_out;
```

A. All three query plans are execution strategies for query 1.

B. All three query plans are execution strategies for query 2.

C. All three query plans could be execution strategies for either query 1 or query 2.

D. Query plans 1 and 3 are execution strategies for query 1, query plan 2 is an execution strategy for query 2.

E. Query plans 1 and 3 are execution strategies for query 2, query plan 2 is an execution strategy for query 1.

6. B

7. (2 points) Which of the following is *true* with respect to bulk loading of a large set of tuples into a table:

A. During bulk loading, individual record insertions are logged to prevent the database going into inconsistent state if the machine crashes in the middle of bulk loading.

B. Bulk loading is faster than individual INSERT statements for inserting tuples.

C. Bulk loading is simply a convenience given to SQL programmers since database management system can easily optimize a long sequence of INSERT statements such that they all are inserted efficiently at once.

D. During bulk loading individual record inserts are not logged.

E. If there is a system crash in the middle of the bulk loading operation, the database will be in an inconsistent state.

7. _____

8. (2 points) Consider the following schedule with 2 transactions:

T_1	T_2
read(B)	
	read(A)
	read(B)
write(B)	
	write(A)
abort;	

Which of the following transaction(s) will be rolled back?

A. Only T_1.

B. Only T_2.

C. Both of them.

D. None of them.

8. A

9. (2 points) Which of the following statements are correct?

1. Atomicity of a partially committed transaction does not need to be ensured. ✓
2. A conflict serializable schedule is also recoverable. ✗
3. A committed transaction needs to be rolled back if the corresponding `COMMIT` entry is missing from the log. ✓
4. During recovery, the recovery manager can write compensation entries in the log only during the REDO phase.

A. Only 1 and 3.

B. Only 3.

C. Only 4.

D. All of them.

9. A

10. (2 points) Consider the following Log state:

$\langle T_0, \text{Start} \rangle$
$\langle T_0, B, 2000, 2050 \rangle$
$\langle T_1, \text{Start} \rangle$
$\langle T_2, \text{Start} \rangle$
$\langle \textbf{checkpoint}, \{T_0, T_1, T_2\} \rangle$
$\langle T_1, C, 700, 600 \rangle$
$\langle T_1, \text{Abort} \rangle$
$\langle T_2, A, 500, 400 \rangle$
$\langle T_0, B, 2000 \rangle$
$\langle T_0, \text{Abort} \rangle$
crash

$T_1 \rightarrow$ ~~undo~~ undo
$T_2 \rightarrow T_0 \rightarrow$ undo
$T_2 \rightarrow$ abort

During the recovery phase after the crash, which of the transaction(s) will be redone and undone?

A. Redo = $\{T_0\}$; Undo = $\{T_1 T_2\}$.

B. Redo = $\{T_0, T_1\}$; Undo = $\{T_2\}$.

C. Redo = NONE; Undo = $\{T_1, T_2, T_0\}$.

D. Redo = NONE; Undo = $\{T_2\}$.

10. B

11. (2 points) Which of the following statements is true:

A. In a MapReduce program, all intermediate results are stored on disk.

B. In a Spark program, all intermediate results are stored on disk.

C. If a worker fails during the execution of a MapReduce program, then the entire program needs to be restarted.

D. If a worker fails during the execution of a Spark program, then the entire program needs to be restarted.

11. _____

12. (2 points) Consider the following dataflow program in Spark:

```
// numbers is of type RDD[Int]

val numbers = sc.parallelize(Array(15, 21, 24, 30, 49))
val x = numbers.flatMap( n => {
        for(p <- primeDivisors(n))
        yield (p,n)
})

// Here primeDivisors(n:Int): Set[Int] is a function that returns
// a set of prime divisors for a number n. For example 11
// primeDivisor(12) = Set(2,3).

val y = x.reduceByKey((a,b)=>(a+b))
y.collect()
```

Which one of the below pairs will be the output of y.collect()

A. (7, 48)

B. (7, 70)

C. (6, 54)

D. (6, 48)

12. B

13. (2 points) Consider the following tables in a relational schema: `Prof (name, area)`, `Teaches (prof, course, quarter)`, `Registered(student, course, quarter)`, `Course (title, number)`, `Advises (prof, student)` where `Prof` lists the name of the professor and her area, `Teaches` tabulates the course that a professor teaches in a quarter, `Registered` tabulates students and the course

they have registered for in a quarter, `Course` lists the title and course number of all courses and `Advises` lists which student is advised by which professor.

Consider the following views on this schema:

V1:

```
CREATE VIEW V1 AS
        SELECT Registered.student, Teaches.prof, Registered.quarter
        FROM Registered, Teaches
        WHERE Registered.course=Teaches.course AND
          Registered.quarter=Teaches.quarter AND
              Registered.quarter > "winter97".
```

V2:

```
CREATE VIEW V2 AS
        SELECT Registered.student, Registered.quarter
        FROM Registered, Teaches
        WHERE Registered.course = Teaches.course AND
                  Registered.quarter = Teaches.quarter AND
                  Registered.quarter >= ''winter98''
```

V3:

```
CREATE VIEW V3 AS
        SELECT Registered.student, Teaches.prof, Registered.quarter
        FROM Registered, Teaches
        WHERE Registered.course=Teaches.course AND
        Registered.quarter >= ''winter98''
```

Suppose the following query is issued:

```
SELECT Advises.prof, Advises.student, Registered.quarter
FROM Registered, Teaches, Advises
WHERE Registered.course=Teaches.course AND
          Registered.quarter=Teaches.quarter AND
          Advises.prof=Teaches.prof AND
          Advises.student=Registered.student AND
          Registered.quarter >= ''winter98''
```

Which of the following is *true*?

A. All 3 views can be used to answer the query.

B. Only `V1` can be used to answer the query.

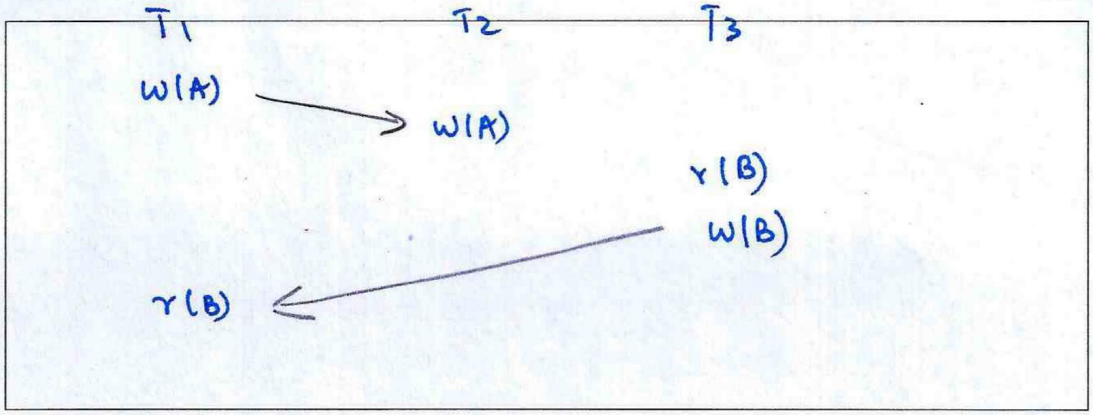
C. Only `V1` and `V2` can be used to answer the query.

D. Only `V1` and `V3` can be used to answer the query.

E. Only `V2` and `V3` can be used to answer the query.

13. _____

14. (2 points) Draw the precedence graph for the schedule w1(A) w2(A) r3(B) w3(B) r1(B).



15. (2 points) Write at least two most useful things learnt in this course.

- I wanted to learnt about how google's Page Rank is implemented. I learnt that ~~in the~~ it ~~uses~~ ~~BF BFF~~ has an ~~a~~sync. implemnt ation on Pregel, there are asynch. implementations aswell

- I wanted to learn how files are organised on Hard disk ~~and fi~~. I learnt that files are stored in the fom of pages. Pages map to blocks that are stored on the disk. Each file has Header meta data, each block has header meta-data and we read records from.

- External – Merge Sort was also interesting shows a new paradigm of algorithms where all ~~me~~ data ~~can~~ cannot be fit into working memory.

Part B: Long Answers (starting in next page)

- Write your answers only within the blank space given after each question.
- There is no negative marking, and partial marks may be awarded.

16. (3 points) Suppose there is a citation database with the following relations:
`article (id, title, year, journal)`, `citations (id1, id2)`
where `article` stores the id of the article, title of the article, the year it was published and the journal in which it was published, and `citations` consists of pairs of ids, id1 and id2 where the article with id1 cites article with id2.

Write a SQL query to find pairs of articles which have at least 5 citations in common. Return article 1, article 2 and the number of citations they have in common.

```
with
  (select  cit1.id1 as i1, cit2.id1 as i2 , 1 as num
   from  citations as cit1, citations as cit2
   where  (cit1.id1 ≠ cit2.id1) AND (cit1.id2 = cit2.id2))
as common-citations,

(select  cc.i1, cc.i2 , sum(num) as tot
   from common-citations as cc
   GROUP BY (cc.i1, cc.i2)) as total-sum

select  ts.i1, ts.i2, ts.tot
From  total-sum as ts
  where  ts.tot >= 5  ;
```

17. (3 points) Consider a geo-spatial database setting where we store millions of (x,y) coordinates of map locations in a table Punkte(x,y) (both x and y are real numbers with high precision). Assume that we have two B+-Trees indexes in the system – one on Punkte.x attribute, and the other on Punkte.y attribute.

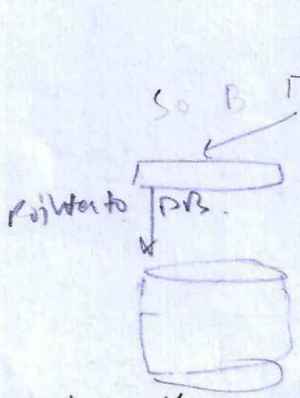
Assume we have a toy database where Punkte consists of 10,00,000 points distributed uniformly randomly in a space in which both x- and y-coordinates range between [0,1000] (closed interval from 0 to 1000). Suppose 100 records fit in a block, and an average B+-Tree leaf has about 200 key-pointer pairs.

Now, estimate the total disk I/O incurred by the following (Make appropriate assumptions (if required) compatible with the statistics already given above). A range query that asks for square in the middle that is $n \times n$ for some $1 \le n \le 1000$.

records = 10^6 # blocks for table = $\frac{10^6}{100} = 10^4$ blocks

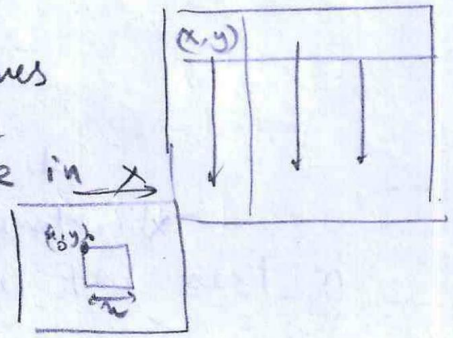
levels of B-tree → 3 levels

1 level can have 200 pointers
2 levels can have $(200)^2$ pointers
3 levels can have $(200)^3$ pointers



So Time to retrieve a file block is 4 block seeks & transfers

I am assuming the query gives x_0, y_0 n as input to denote location of the block in the grid



$(4)(t_T + t_s) + (10n \times t_T) + \left(\frac{n^2}{100}\right) \times (t_T + t_s)$

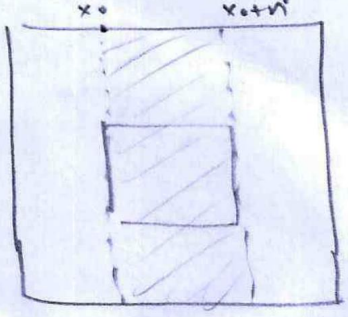
$\frac{n^2}{100}$ # output blocks

$(t_T + t_s)$ Seek & transfer time for each output block

We fetch all blocks from the B tree in the range $x_0 \cdots x_0 + n$, we reject point if $y < y_0$ or $y_0 + n < y$ else we accept it.

time to get first record for $x_0 \rightarrow (3+1)(t_T + t_s)$

for remaining blocks we would only need transfer time. Note that we would get a lot of extra blocks:



The number of points in region: $\frac{\text{Area}}{10^6} \times \frac{10^6}{\text{Total point}} = \text{Area}$

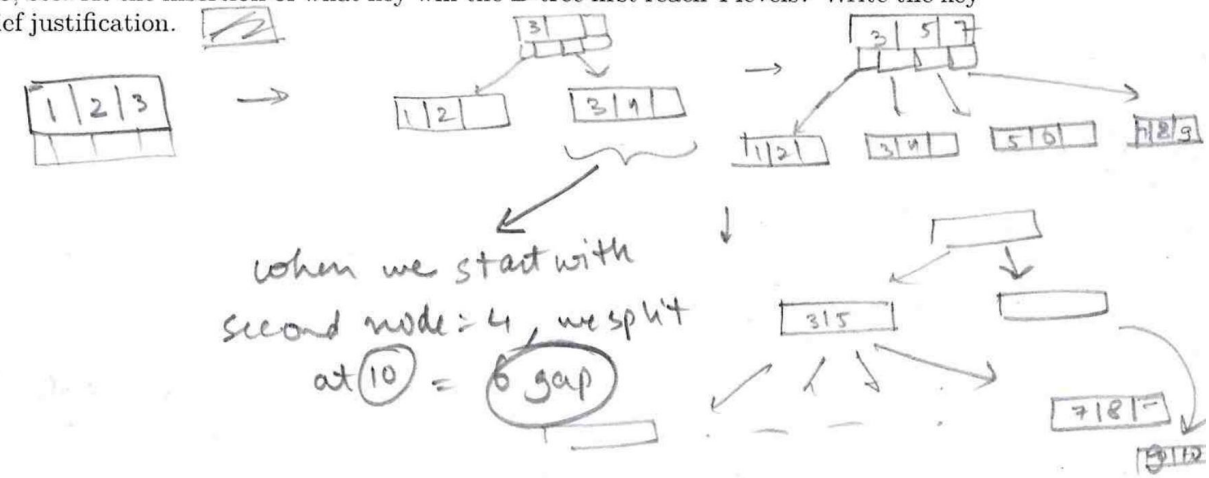
Area of grid

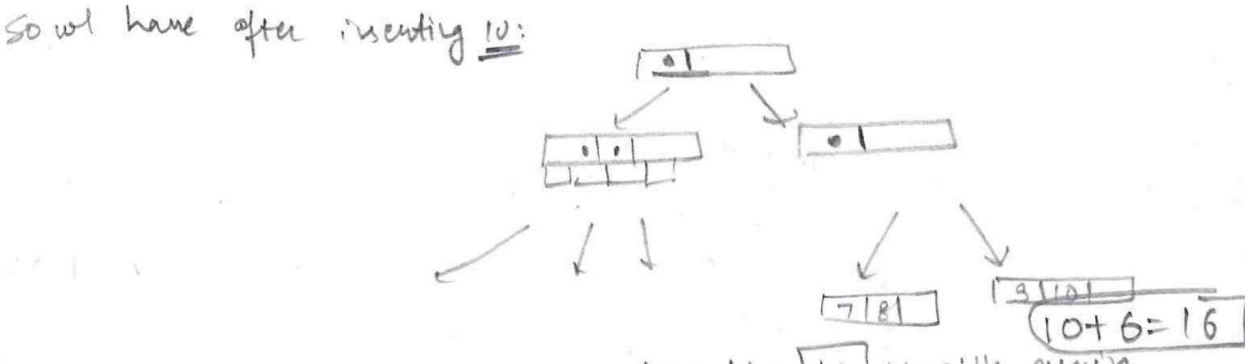
these points would be present on $\frac{10^3 n}{100} = 10n$ blocks $= 10^3 \times n$

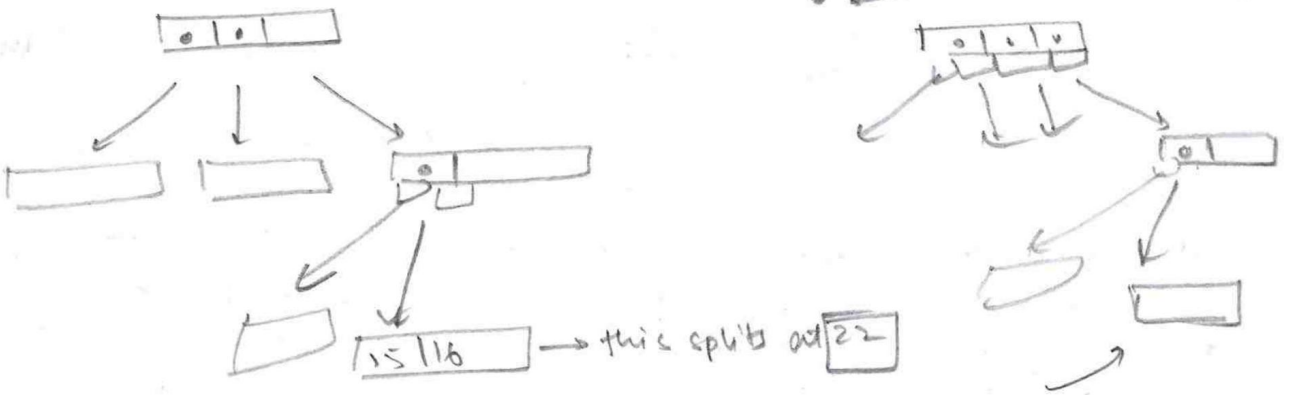
Now we also need to output n^2 blocks

y takes $\frac{n^2}{100}$ output blocks. So total I/O incurred is

18. (3 points) Suppose we have a B+-tree of order 4 (that is, maximum 3 keys). On insertion, the splitting policy is as follows: i) if a leaf node is split, the pointers are divided as 2 pointers to the left, 2 to the right (note that these are pointers to data), and, ii) if an internal node is split, the pointers are divided as 3 pointers to the left, 2 to the right (these are pointers to child nodes). Suppose we start with a leaf node containing keys 1,2,3. We then add records with keys 4,5,6, etc. At the insertion of what key will the B-tree first reach 4 levels? Write the key and a brief justification.

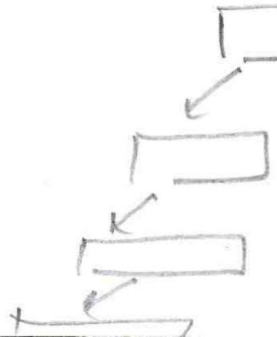






so on inserting 28

we get split at top level = 4 Levels



19. (6 points) For each of the following problems describe how you would solve it using map-reduce – i.e., how input is mapped (or computed) to (key, value) pairs by the map stage; how the (key, value) pairs produced by the map stage are processed by the reduce stage to get the final answer(s).

It is possible that a single map-reduce pass may not be sufficient in some cases. Then describe how it can be structured into two (or more) map-reduce jobs with the output of the first job becoming the input to the next one(s).

Note that you are required only to write the pseudo-code of map-reduce functions. You need not write the implementation in Java or any other language.

A. The input is a list of student data where each input record contains information about a single student: `(id, name, department, grade, course)`. The output should be the average course grade of students in each department. You may assume grades are integers between 1-8.

I am assuming average of grades over all courses

Map(Lid, name, dept, grade, Cours)) :

return < department, (grade, 1)>

key — value (to maintain count)

Reduce (< department, (grade, 1)> :

start with init=0 & Reduce on grade: using addition

start with init=0 & Reduce on 1 : using addition

output < department, (gradesum, nums)>

(eg. Reduce(<CS, (7,1)>, <CS, (8,1)>, <CS, (3,1)>) = <CS, (18,3)>)

Map (< department, (gradesum, nums)> :

Return < department, gradesum/nums>

→ No need to Reduce, (can use a redundant Reduce if necessary)

output tuples will contain department as key, average course grade of students in that dept. as value.

B. The input consists of two lists – one list containing voter info (`voterid, name, age, pincode`), and the other containing disease data (`pincode, age, disease`). For each unqiue pair of age and pincode values, the output would be a list of names and a list of diseases for people in that pincode and age. If a particular age/pincode pair appears in one input list but not the other, then that age/pincode pair can appear in the output with an empty list of names or diseases, or you can omit it from the output entirely, depending on which is easier.

Map (<voterid, name, age, pincode>)

= return

< (pincode, age) , name >

Map (< pincode, age, disease>)

= return

< (pin, age) , disease>

Reduce (<(pincode, age>, name>

= return (<pincode, age>, [name])

aggregate names into a table using append as reducer

Starting []

Reduce (<pin, age>, disease)

= return (<pin, age>, [disease])

~~Map~~ Reduce (<pin, age>, list)

~~tab~~ take pairwise concat of list start with [(,)]

ONLY FOR ROUGH WORK - WILL NOT BE EVALUATED