Is my friend correct?
Can you show me the evidence by building an index for the second letter.

Here are some parameters you may or may not need
1. Each index page can record 200 words
2. there are 12,000 words that has 5 letters, taking 50 pages
3. the selection factor is 0.05

How about I make 5 copies of the list and sort them on specific position? (so list 1 is sorted on the first letter, list 2 is sorted on the second letter, etc). Is indexing still better? Which option requires less space? And which option is more efficient?

B tree index dense

12000 <sk, pnt> pairs

60 data entry nodes

1 directory node

If sk is just the second letter

query result ≥ 600 words

index read root + 3 data entry = 4 pgs.

data read 600 pgs

not good.

If sk is the entire word

index read root + 3 data entry = 4 pgs

no need to read data

5 sorted file = 50×5 = 250 pg.

5 dense index = 61×5 = 305 pg

if use index, still need the original data 50pg.

search on sorted file total 355pg.

$\log_2(50)+3$

We have the following relations

Students

snum	name	gender
1001	Randy	M
1005	Nicole	F

Departments

dcode	Dpt_name	college
403	Chemical Engineering	Engineering
402	Mathematics	LAS
404	Landscape Architect	Design
401	Computer Science	LAS

Degree

name	level	dcode
Computer Science	BS	401
Computer Science	MS	401
Computer Science	PhD	401
Software Engineering	BS	401
Landscape Architect	BS	404
Chemical Engineering	BS	403
Applied Mathematics	MS	402

Major

snum	name	level
1005	Computer Science	MS
1001	Software Engineering	BS

Minor

snum	name	level
1001	Computer Science	BS
1005	Applied Mathematics	MS

Courses

cnum	cname	description	cdt	level	dcode
113	Spreadsheet	Microsoft Excel and Access	3	Undergraduate	401
311	Algorithm	Design and Analysis	3	Undergraduate	401
531	Theory of Computation	Theorem and Probability	3	Graduate	401
363	Database	Design Principle	3	Undergraduate	401
412	Water Management	Water Management	3	Undergraduate	404
228	Special Topics	Interesting Topics about CE	3	Undergraduate	403
114	Calculus	Limit and Derivative	4	Undergraduate	402

Registers

snum	cnum	dcode	semester	grade
1001	228	403	Spring2015	4
1005	114	402	Spring2015	4
1005	113	401	Spring2015	4
1001	363	401	Fall2015	3.8

(a) optimize relational algebra expressions for the following queries

i. Find the names and levels of degrees offered by LAS

$\Pi_{name, level}(\sigma_{college=LAS}(Dept) \bowtie Deg)$

ii. Find the department names that offer both graduate and undergraduate courses

$\Pi_{name}(\sigma_{level=grade}(coures) \bowtie Dept \cap \sigma_{level=undergrad}(courses) \bowtie Dept)$

You are given four schemas R(a, b, c), S(b, d), T(b, e), U(b, f).

(b) Consider the following SQL query

```
SELECT   *
FROM     R, S, T, U
WHERE    R.b = S.b
         AND S.b = T.b
         AND T.b = U.b
```

i. Write the relational algebra for this query. Draw a left-deep plan for the query that has selection pushed down.

$R \bowtie S \bowtie T \bowtie U$

ii. How many left-deep plans will be considered by the IBM System R (which we discussed in our lectures)? You need to explain why in order to receive points.

4!

Derive the I/O costs of different join algorithms of relations R and S given the following variables, which you may or may not use all of them. Ignore the CPU time costs and the cost of writing the results. Write down steps for partial credits

|R|=10: Number of tuples per page in R
|S|=20: Number of tuples per page in S
M=200: Number of pages in R
N=40: Number of pages in S
B=10: Number of available memory in pages

c). What is the procedure Sort-Merge Join? What is the minimal I/O cost?

sort R and S if they are not sorted

merge R and S

sort R cost: pass 1 200/10 = 20 sorted subfiles
pass 2 merge 9 files into 1. 3 sorted files
pass 3 merge 3 files into 1
2×3×M = 1200

Sort S cost: 2×2×40 = 160

merge R, S: M+N = 240

total cost = 240 + cost of sorting (R) + cost of sorting (S)

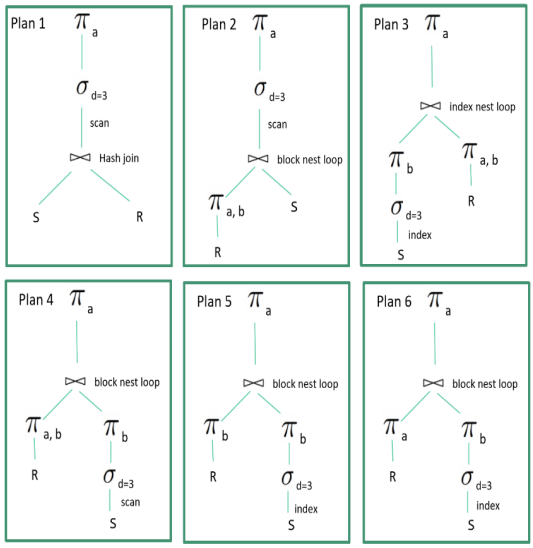
points) If the number of available memory in pages B increase to 20, and assumptions remain the same

1) the minimal I/O cost of block nested loop join will
increase[ ] decrease[✓] do not change[ ]
2) the minimal I/O cost of simple nested loop join will
increase[ ] decrease[✓] do not change[ ]
3) the minimal I/O cost of grace hash join will
increase[ ] decrease[ ] do not change[✓]
4) the minimal I/O cost of Sort-Merge join will
increase[ ] decrease[ ] do not change[ ]

if no need to sort R: do not change
if need to sorted R: decrease

(a) Consider the following SQL query and evaluation plans

```
SELECT  R.a
FROM    R, S
WHERE   R.b = S.b
        AND S.d = 3
```



i. Which plan(s) is not correct (i.e., producing wrong results)? You need to explain why in order to receive points. 5 & 6

5. Π_b removed field a before join

6. Π_a Π_b cannot join

ii. Which plan is likely to be the most efficient? You need to explain why in order to receive points. Can you estimate the I/O cost for the plans?

3 better than 4 better than 1 or 2

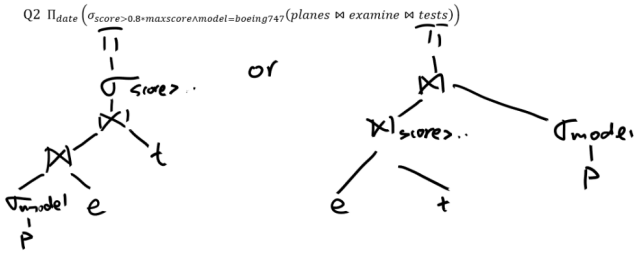
3. use index to select d=3. d is primary key, so selection result is a single record
R has index on b, and b is unique for R. so just need to look up for a single record using index.

2. (50 points) Consider the following relations: Technicians(SSN, tech_name, address, phone_number), Tests(FAAid, test_name, max_score), Planes(Pid, model), and Examine(SSN, FAAid, Pid, date, score), and the following queries:
   - Q1: Find the names and phone_numbers of the technicians who examine a plane from 10/27/2021 to 10/28/2022;
   - Q2: Find the dates that at least one Boeing 747 plane got higher than 80% of the max scores in its tests. (Hint: Boeing 747 is a model, not a Pid);
   - Q3: Find the name and ssn of the technicians who have not conducted any test on any Boeing 747 plane.

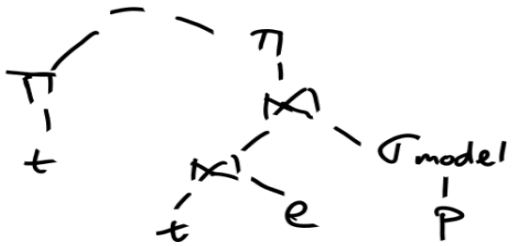
   a) (12 pts) For each of the queries, write a relational algebraic expression. (4 pts each)
   b) (30 pts) For each of the queries, draw their expression trees with selection and projection conducted as early as possible. Use left-deep joins whenever joins are needed. (10 pts each)
   c) (8 pts) If I want to join all four tables, how many left-deep plans without cross product are there? Write down all these plans by drawing their expression trees or writing their relational algebraic expressions with proper parenthesis. (Hint: if two tables do not have a common attribute, then natural join is defined as cross product, and thus should be excluded).

Q1 $\Pi_{technname,phone}\left(\sigma_{date>10/27/2021 \wedge date<10/28/2022}\left(technicians \bowtie examine\right)\right)$

Q2 $\Pi_{date}\left(\sigma_{score>0.8*maxscore \wedge model=boeing747}(planes \bowtie examine \bowtie tests)\right)$



Q3 $\Pi_{name,ssn}(tech) - \Pi_{name,ssn}\left(\sigma_{model=boeing747}(tech \bowtie examine \bowtie plane)\right)$



c. 3*2*1*2=12 plans. Examine needs to joint with other tables first.

$((E \bowtie TB1) \bowtie TB2) \bowtie TB3$

$((TB1 \bowtie E) \bowtie TB2) \bowtie TB3$

S ⋈ R

Nested loop join
For every record in S:
    Scan R to find the matching record

N + N*P_s*M

Nested block loop join
For every B-2 pages in S:
    Scan R to find the matching record

N + ceil(N/(B-2))*M

Indexed nested loop join (assume R has the corresponding index)
For every record in S:
    Look up matching record in R using index

N + N*P_s*(cost of finding matching record in R using index)

- Consider two relations
  - Sailors: 1000 pages, 100 records/page
  - Reserves: 500 pages, 80 records/page
- Estimate the cost of join the two relations with Sailors being the outer relations, with these approaches (ignoring the cost of writing the results)
- Buffer size = 102
- Index look up = 4 pages per records

S ⋈ R

- Simple nested loop join

for every record in S:
    scan R

1000*100*500+1000 I/O cost

- Block nested loop join, assuming the total memory is 102 block

- Index nested loop join (assuming the inner relation is indexed by a sparse and clustered B^+-tree, and the cost of searching a record takes 4 pages)

for every record in S:
    search for the corresponding record in R

1000+1000*100*4

- Grace hash join (assuming the total memory is 102 blocks)

$B > \sqrt{M}$?
102>sqrt(1000) ?
3*(M+N)
3*(1000+500)

assumption: memory size to 5
Every relation, 1 page for the relation, 4 pages for the partition.
For each partition of S, 1000/4=250 h1
Continue partition:
Apply another hash function h2
250/4=63