CS3223 Tutorial 6

Gary Lim

Admin

- How was the midterms? :p

Admin

- How was the midterms? :p
- **Project**
  - Submit report and code by **18 Mar 2022**, Week 9 Friday
  - Project demo in Week 10 (book a slot on the Google docs)
  - Support partition-based join, aggregates (SUM, COUNT, AVG, MIN, MAX), GROUP BY, DISTINCT
  - Experiments and discussions

Chapter Review

- Query Optimization
  - Plan Space
  - Search algorithm: Systems R

Query Optimization

- Find the **best** query evaluation plan (QEP) for a given query

Efficient, less I/O

these query evaluation plans will return the same answers

Query Optimization

- Find the **best** query evaluation plan (QEP) for a given query
- Consists of the following:
  - **Plan space** (set of all semantically equivalent plans)
  - **Enumeration / Search algorithm** (efficient search, able to find a ‘decent’ plan)
  - **Cost model** (basis for comparison between plans)

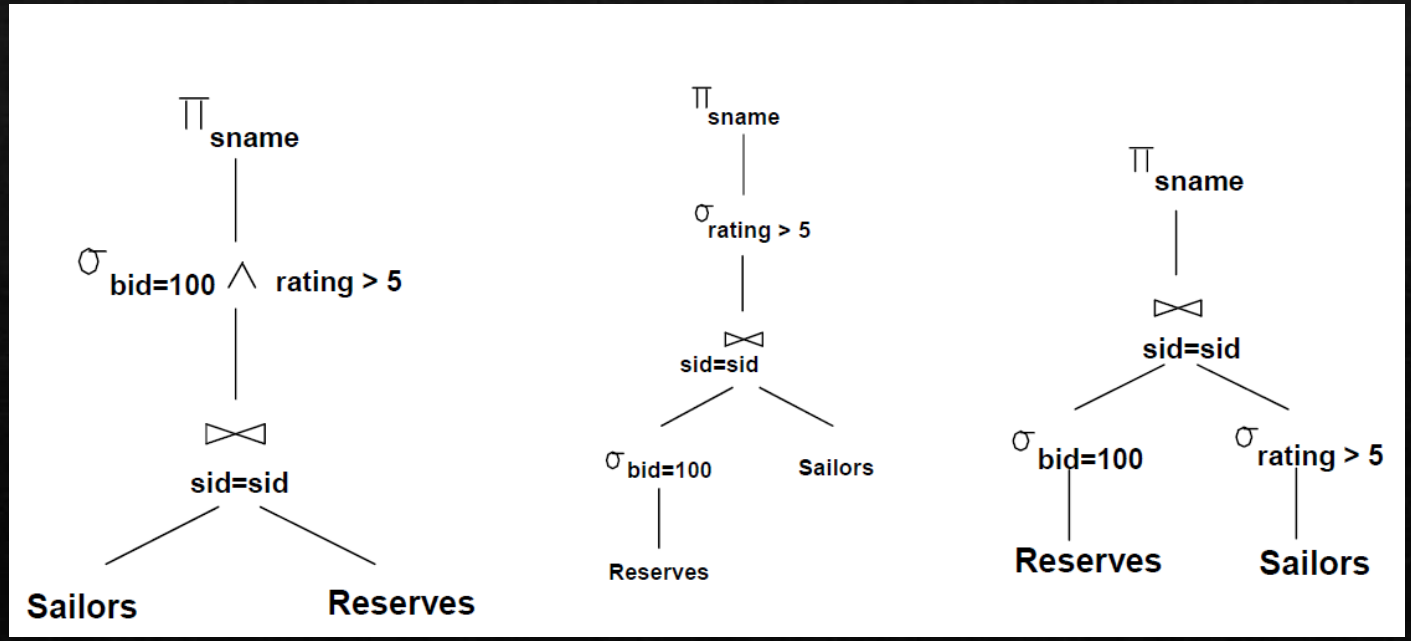
Query Optimization

- Common Heuristics?

Plan Space

- Common Heuristics
  - Push down **selections** and **projections**
  - Smallest relation first
  - Avoid cross products
  - Focus on a certain type of plans (e.g. left deep trees)

Selection pushed down

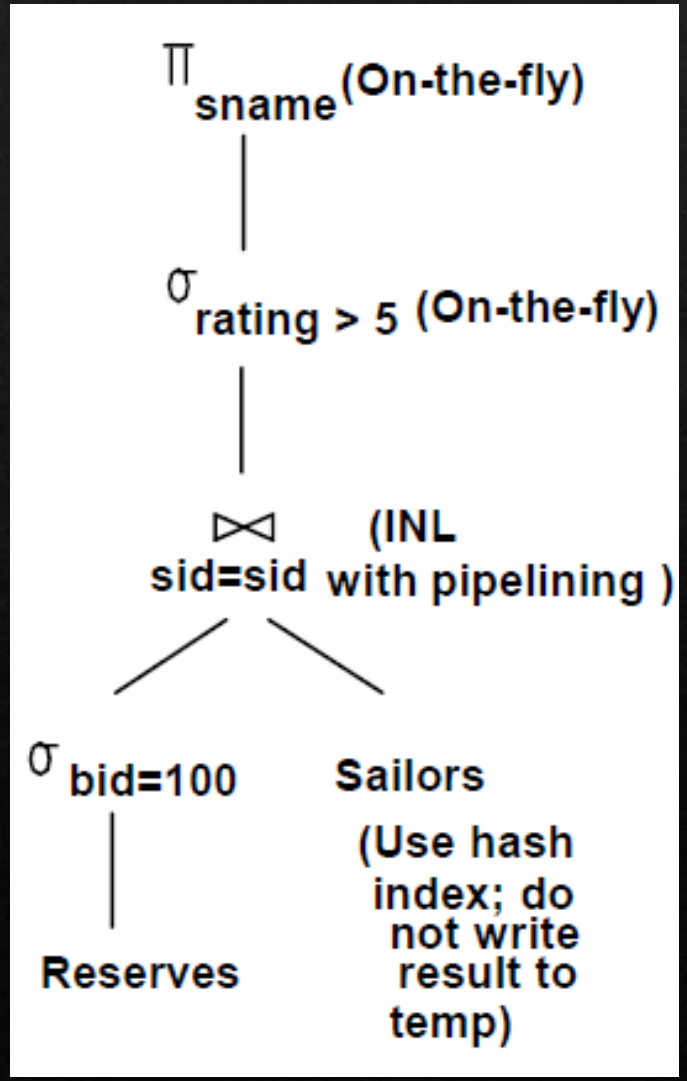


Plan Space

- Common Heuristics
  - Push down **selections** and **projections**
  - Smallest relation first
  - Avoid cross products
  - Focus on a certain type of plans (e.g. left deep trees)

Not always ideal to push down selection

Why?



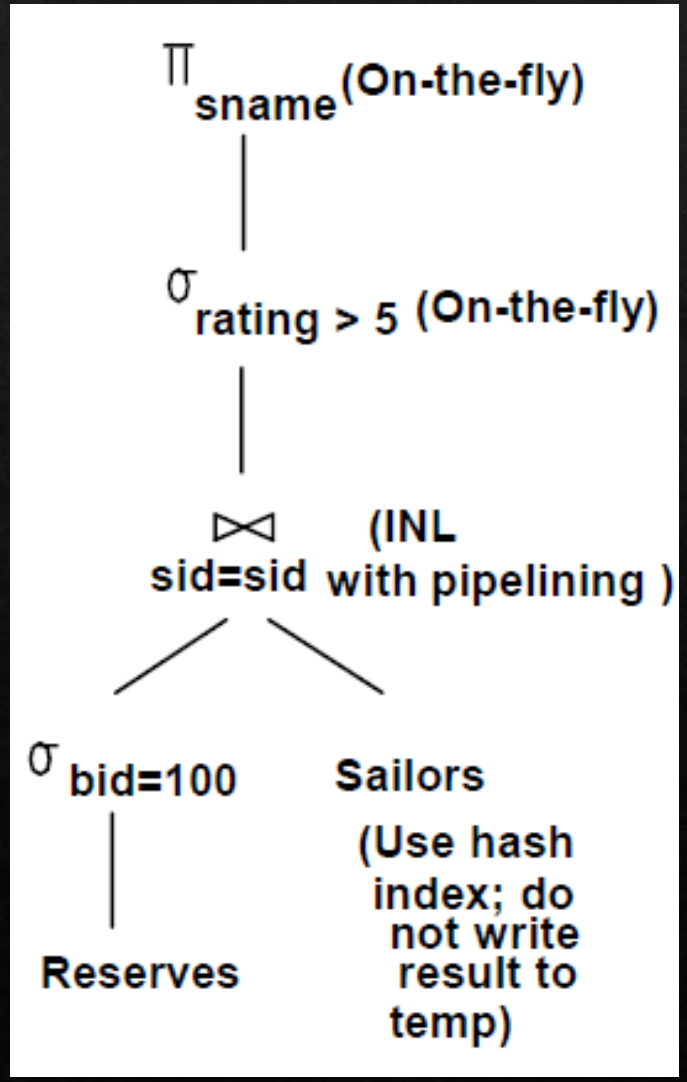
Plan Space

- Common Heuristics
  - Push down **selections** and **projections**
  - Smallest relation first
  - Avoid cross products
  - Focus on a certain type of plans (e.g. left deep trees)

Not always ideal to push down selection

Why?

If we push down $\sigma_{rating>5}$ we need to perform a full index scan on the hash index for sailors, and write out the intermediate result, and cannot use index nested loop join anymore



Plan Space

Left-deep tree

- Right child has to be a base table

Right-deep Tree

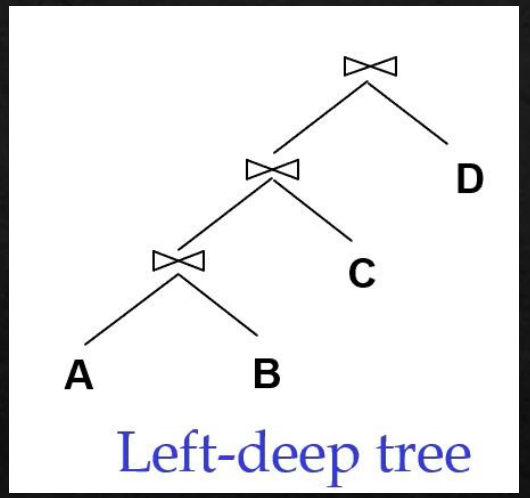
- Left child has to be a base table

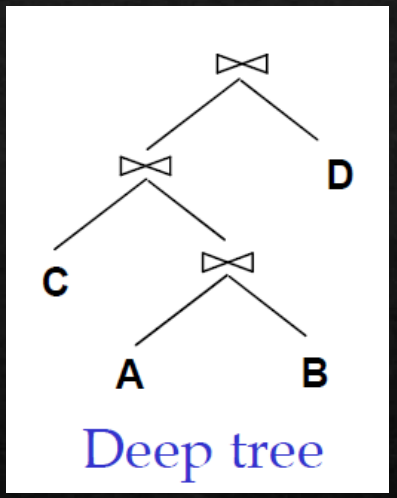
Deep Tree

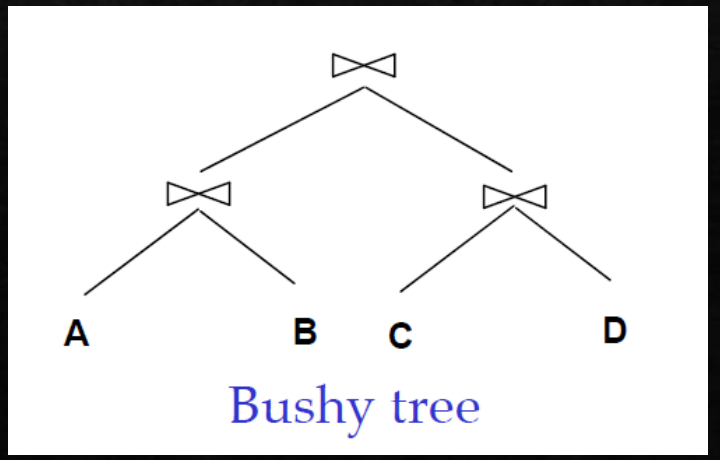
- One of the children is a base table

Bushy Tree

- No restriction







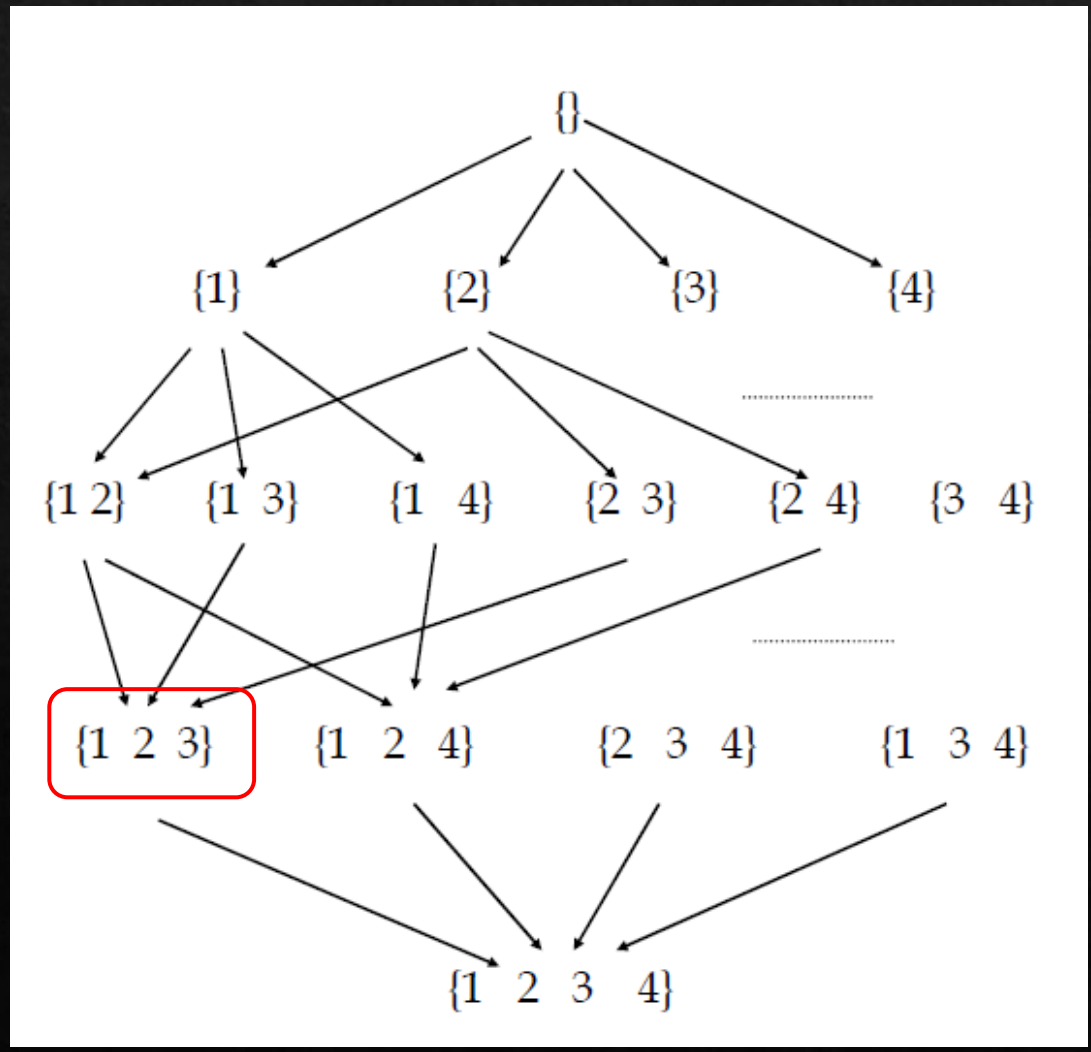
Search Algorithms

Search Algorithm	Search Space	Plan Quality	Optimization Overhead
Exhaustive	Complete	✓✓	✗✗
Greedy	Small	✓/✗	✓ Polynomial
Randomized	Depends	✓/✗	Depends on how long it is ran
Systems R	Almost complete	✓	✗ $O(2^k)$

Systems R

- Only left-deep trees
- **Principal of Optimality**
  - If two plans differ only in a subplan, then the plan with the better subplan is also the better plan (allows you to solve with DP due to overlapping subproblems)
- Number of subproblems: 2^k where k is number of relations

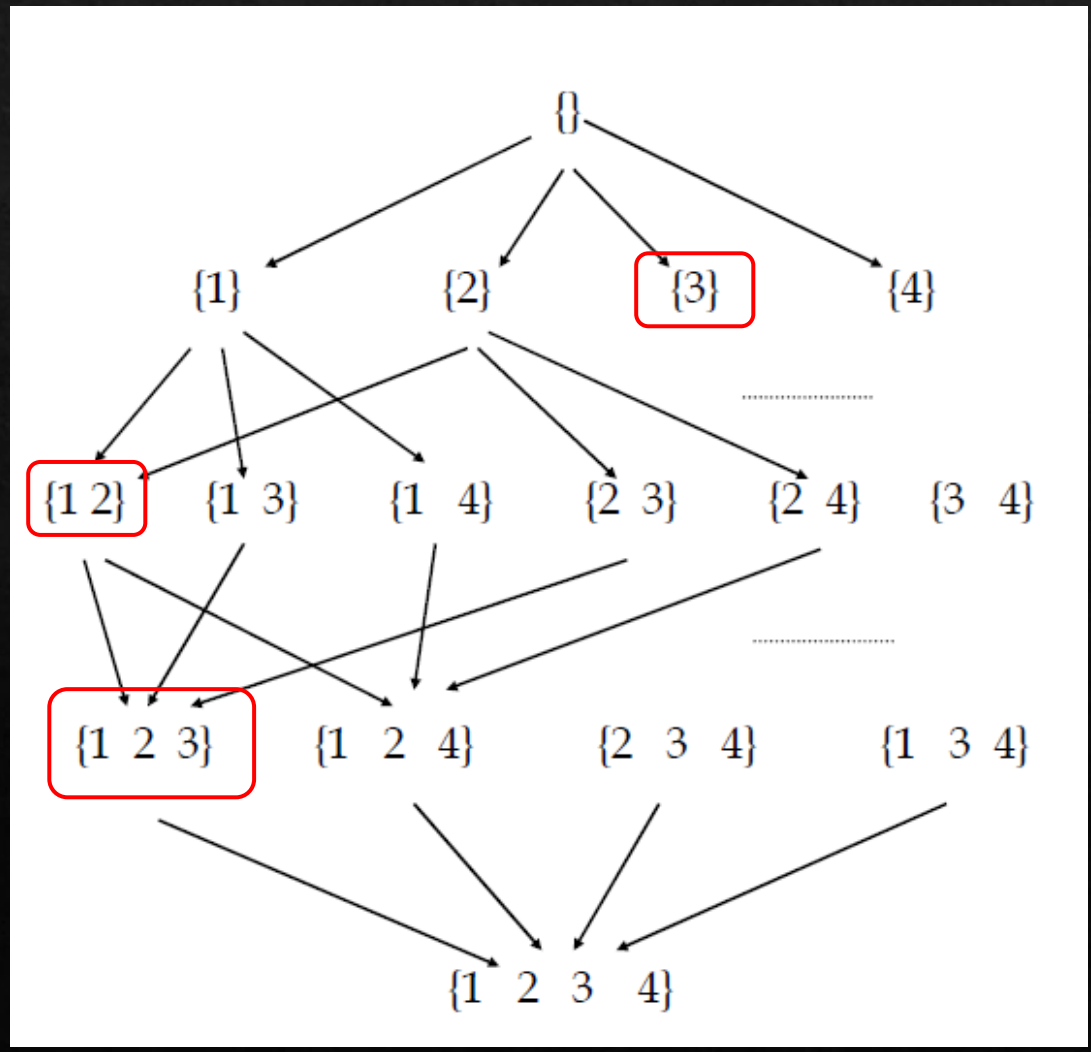
$$\sum_{k=0}^{n} \binom{n}{k} = 2^n$$



Systems R

- Only left-deep trees
- **Principal of Optimality**
  - If two plans differ only in a subplan, then the plan with the better subplan is also the better plan (allows you to solve with DP due to overlapping subproblems)
- Number of subproblems: 2^k where k is number of relations

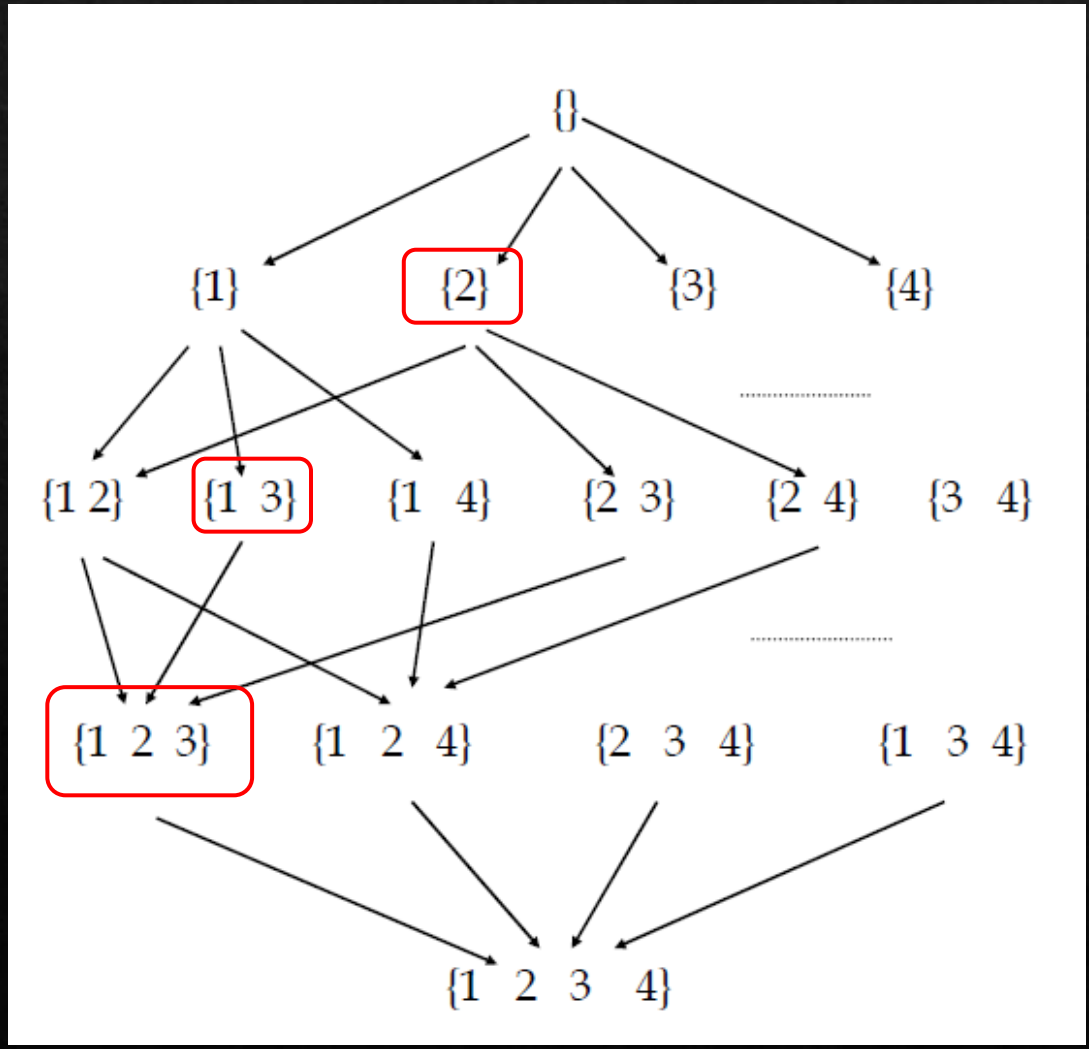
$$\sum_{k=0}^{n} \binom{n}{k} = 2^n$$



Systems R

- Only left-deep trees
- **Principal of Optimality**
  - If two plans differ only in a subplan, then the plan with the better subplan is also the better plan (allows you to solve with DP due to overlapping subproblems)
- Number of subproblems: 2^k where k is number of relations

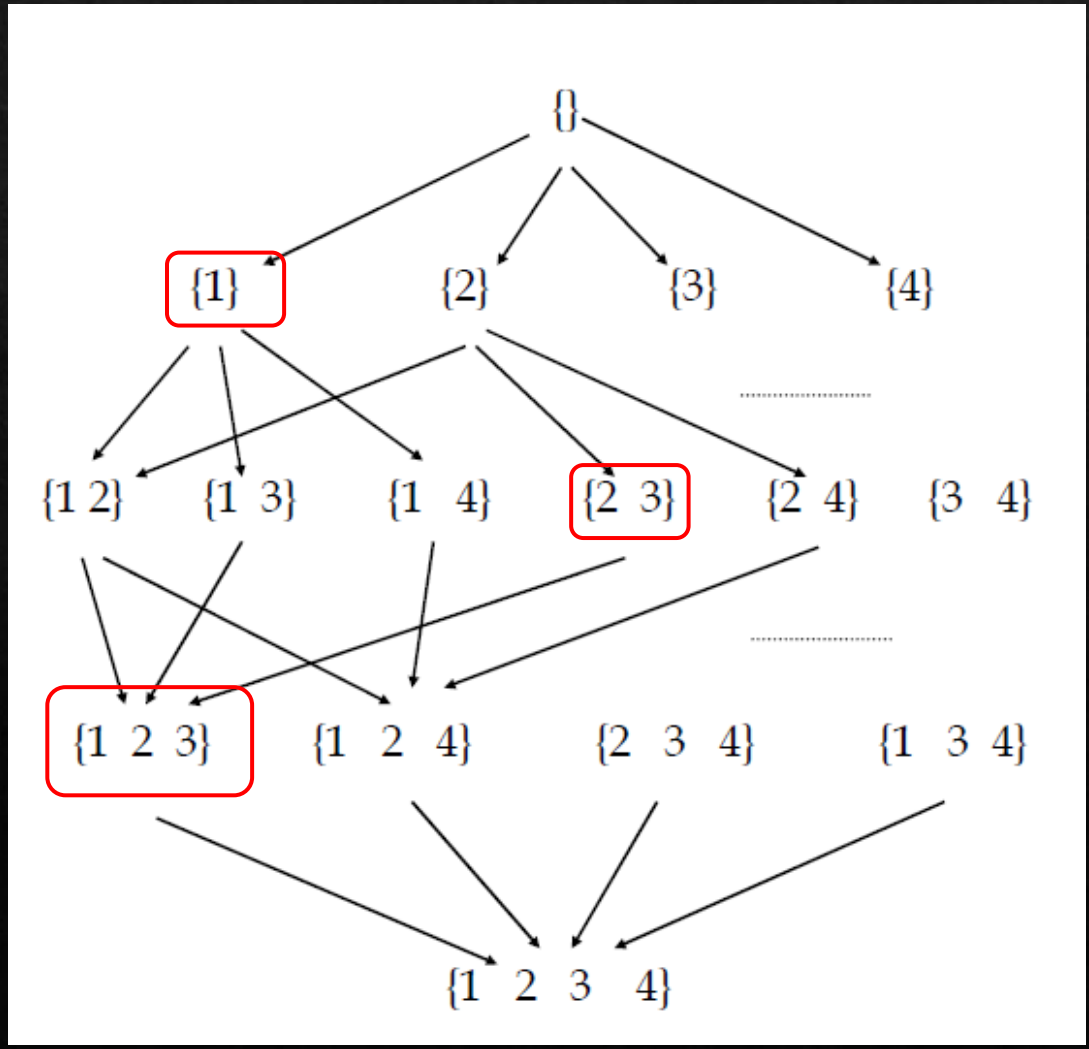
$$\sum_{k=0}^{n} \binom{n}{k} = 2^n$$



Systems R

- Only left-deep trees
- **Principal of Optimality**
  - If two plans differ only in a subplan, then the plan with the better subplan is also the better plan (allows you to solve with DP due to overlapping subproblems)
- Number of subproblems: 2^k where k is number of relations

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n$$



Tutorial Questions

Q1

- R1(**A**, B, C) – 10,000 tuples
- R2(**C**, D, E) – 15000 tuples
- R3(**E**, F) – 7500 tuples

- 1 page ← 10 tuples of R1

Q1a

- R1(**A**, B, C) – 10,000 tuples
- R2(**C**, D, E) – 15000 tuples
- R3(**E**, F) – 7500 tuples

- 1 page ← 10 tuples of R1

Estimate the result size of R1 ⋈ R2 ⋈ R3

Q1a

- R1(**A**, B, **C**) – 10,000 tuples
- R2(**C**, D, **E**) – 15,000 tuples
- R3(**E**, F) – 7,500 tuples

- Since A, C and E are **primary keys**, there are
  - 10,000 distinct values of R1.A,
  - 15,000 distinct values of R2.C
  - 7,500 distinct values of R3.E
- Assuming **uniform distribution**,
  - all 10,000 values of R1.C are evenly distributed
  - all 15,000 values of R2.E are evenly distributed

Estimate the result size of R1 ⋈ R2 ⋈ R3

Q1a

- R1(**A**, B, **C**) – 10,000 tuples
- R2(**C**, D, **E**) – 15,000 tuples
- R3(**E**, F) – 7,500 tuples

- Since A, C and E are **primary keys**, there are
  - 10,000 distinct values of R1.A,
  - 15,000 distinct values of R2.C
  - 7,500 distinct values of R3.E
- Assuming **uniform distribution**,
  - **all 10,000 values of R1.C are evenly distributed**
  - all 15,000 values of R2.E are evenly distributed

Estimate the result size of R1 ⋈ R2 ⋈ R3

- R1 ⋈ R2
  - Every R1 tuple has a matching R2 tuple (**10,000** tuples)

Q1a

- R1(**A**, B, **C**) – 10,000 tuples
- R2(**C**, D, **E**) – 15,000 tuples
- R3(**E**, F) – 7,500 tuples

- Since A, C and E are **primary keys**, there are
  - 10,000 distinct values of R1.A,
  - 15,000 distinct values of R2.C
  - 7,500 distinct values of R3.E
- Assuming **uniform distribution**,
  - all 10,000 values of R1.C are evenly distributed
  - **all 15,000 values of R2.E are evenly distributed**

Estimate the result size of R1 ⋈ R2 ⋈ R3

- R1 ⋈ R2
  - Every R1 tuple has a matching R2 tuple (**10,000** tuples)
- R1 ⋈ R2 ⋈ R3
  - Every R3 tuple has 2 matching R2 tuples; (still **10,000** tuples)

Q1a

- R1(**A**, B, **C**) – 10,000 tuples
- R2(**C**, D, **E**) – 15,000 tuples
- R3(**E**, F) – 7,500 tuples

1
1
2
1

- Since A, C and E are **primary keys**, there are
  - 10,000 distinct values of R1.A,
  - 15,000 distinct values of R2.C
  - 7,500 distinct values of R3.E
- Assuming **uniform distribution**,
  - all 10,000 values of R1.C are evenly distributed
  - **all 15,000 values of R2.E are evenly distributed**

Estimate the result size of R1 ⋈ R2 ⋈ R3

- R1 ⋈ R2
  - Every R1 tuple has a matching R2 tuple (**10,000** tuples)
- R1 ⋈ R2 ⋈ R3
  - Every R3 tuple has 2 matching R2 tuples; (still **10,000** tuples)

Q1b

- R1(**A**, B, **C**) – 10,000 tuples
- R2(**C**, D, **E**) – 15,000 tuples
- R3(**E**, F) – 7,500 tuples

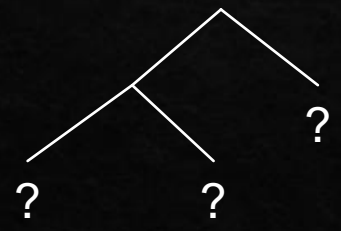
List all left-deep tree plans, excluding those with cross products.

Q1b

- R1(**A**, B, **C**) – 10,000 tuples
- R2(**C**, D, **E**) – 15,000 tuples
- R3(**E**, F) – 7,500 tuples

List all **left-deep tree** plans, excluding those with cross products.

Recall: For left-deep trees, right child must be base table



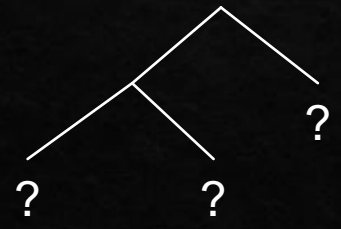
3! ways

Q1b

- R1(**A**, B, **C**) – 10,000 tuples
- R2(**C**, D, **E**) – 15,000 tuples
- R3(**E**, F) – 7,500 tuples

List all left-deep tree plans, **excluding those with cross products**.

Exclude R1 ⋈ R3



3! – 2 = 4

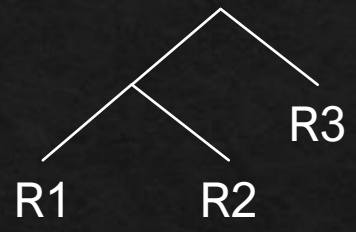
Q1b

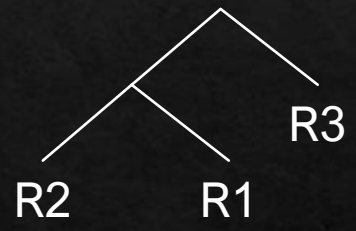
- R1(**A**, B, **C**) – 10,000 tuples
- R2(**C**, D, **E**) – 15,000 tuples
- R3(**E**, F) – 7,500 tuples

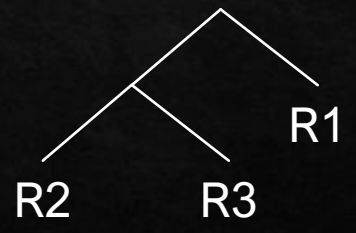
List all left-deep tree plans, excluding those with cross products.

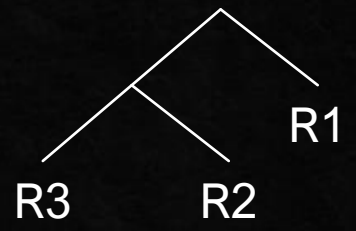
1. (R1 ⋈ R2) ⋈ R3
2. (R2 ⋈ R1) ⋈ R3
3. (R2 ⋈ R3) ⋈ R1
4. (R3 ⋈ R2) ⋈ R1

Depending on join algorithm, R1 ⋈ R2 can have a different cost from R2 ⋈ R1 (e.g. block nested join, hash-join)









Q1c

- R1(**A**, B, **C**) – 10,000 tuples
- R2(**C**, D, **E**) – 15,000 tuples
- R3(**E**, F) – 7,500 tuples
- R1⋈R2(A, B, C, D, E) – 10,000 tuples
- R2⋈R3(C, D, E, F) – 15,000 tuples

- Using block-nested loop joins
- 100 buffer pages

main focus is that must calculate how many tuples there can be in a page
- since we know the size of an attribute
- then will need to know how many attributes will exist in the relation

Q1c

- R1(**A**, B, **C**) – 10,000 tuples
- R2(**C**, D, **E**) – 15,000 tuples
- R3(**E**, F) – 7,500 tuples
- R1⋈R2(A, B, C, D, E) – 10,000 tuples
- R2⋈R3(C, D, E, F) – 15,000 tuples

- Using block-nested loop joins
- 100 buffer pages
- 10 tuples of R1 per page (30 attributes per page)
- 10 tuples of R2 per page
- 15 tuples of R3 per page
- 6 tuples of R1 ⋈ R2 per page
- 7 tuples of R2 ⋈ R3 per page

Attributes have the same size

Cost of (R1 ⋈ R2) ⋈ R3

T: R1 ⋈ R2: $\underbrace{\frac{10000}{10} + \left\lceil \frac{\frac{10000}{10}}{98} \right\rceil \times \frac{15000}{10}}_{\text{read}} + \underbrace{\left\lceil \frac{10000}{6} \right\rceil}_{\text{write R1⋈R2}} = 19167$

T ⋈ R3 : $\underbrace{\left\lceil \frac{10000}{6} \right\rceil + \left\lceil \frac{\left\lceil \frac{10000}{6} \right\rceil}{98} \right\rceil \times \frac{7500}{15}}_{\text{read R1⋈R2 and R3}} = 10667$

29834 I/O

Q1c

- R1(**A**, B, **C**) – 10,000 tuples
- R2(**C**, D, **E**) – 15,000 tuples
- R3(**E**, F) – 7,500 tuples
- R1⋈R2(A, B, C, D, E) – 10,000 tuples
- R2⋈R3(C, D, E, F) – 15,000 tuples

- Using block-nested loop joins
- 100 buffer pages
- 10 tuples of R1 per page (30 attributes per page)
- 10 tuples of R2 per page
- 15 tuples of R3 per page
- 6 tuples of R1 ⋈ R2 per page
- 7 tuples of R2 ⋈ R3 per page

Attributes have the same size

Cost of (R2 ⋈ R1) ⋈ R3

T: R2 ⋈ R1: $\frac{15000}{10} + \left\lceil \frac{\frac{15000}{10}}{98} \right\rceil \times \frac{10000}{10} + \left\lceil \frac{10000}{6} \right\rceil = 19167$

read | write R1⋈R2

T ⋈ R3 : $\left\lceil \frac{10000}{6} \right\rceil + \left\lceil \frac{\left\lceil \frac{10000}{6} \right\rceil}{98} \right\rceil \times \frac{7500}{15} = 10667$

read R1⋈R2 and R3

29834 I/O

Q1c

- R1(**A**, B, **C**) – 10,000 tuples
- R2(**C**, D, **E**) – 15,000 tuples
- R3(**E**, F) – 7,500 tuples
- R1⋈R2(A, B, C, D, E) – 10,000 tuples
- R2⋈R3(C, D, E, F) – 15,000 tuples

- Using block-nested loop joins
- 100 buffer pages
- 10 tuples of R1 per page (30 attributes per page)
- 10 tuples of R2 per page
- 15 tuples of R3 per page
- 6 tuples of R1 ⋈ R2 per page
- 7 tuples of R2 ⋈ R3 per page

Attributes have the same size

Cost of (R2 ⋈ R3) ⋈ R1

T: R2 ⋈ R3: $\frac{15000}{10} + \left\lceil \frac{\frac{15000}{10}}{98} \right\rceil \times \frac{7500}{15} + \left\lceil \frac{15000}{7} \right\rceil = 11643$

read | write R2⋈R3

T ⋈ R1 : $\left\lceil \frac{15000}{7} \right\rceil + \left\lceil \frac{\left\lceil \frac{15000}{7} \right\rceil}{98} \right\rceil \times \frac{10000}{10} = 24143$

read R2⋈R3 and R1

35786 I/O

Q1c

- R1(**A**, B, **C**) – 10,000 tuples
- R2(**C**, D, **E**) – 15,000 tuples
- R3(**E**, F) – 7,500 tuples
- R1⋈R2(A, B, C, D, E) – 10,000 tuples
- R2⋈R3(C, D, E, F) – 15,000 tuples

- Using block-nested loop joins
- 100 buffer pages
- 10 tuples of R1 per page (30 attributes per page)
- 10 tuples of R2 per page
- 15 tuples of R3 per page
- 6 tuples of R1 ⋈ R2 per page
- 7 tuples of R2 ⋈ R3 per page

Attributes have the same size

Cost of (R3 ⋈ R2) ⋈ R1

T: R3 ⋈ R2: $\frac{7500}{15} + \left\lceil \frac{\frac{7500}{15}}{98} \right\rceil \times \frac{15000}{10} + \left\lceil \frac{15000}{7} \right\rceil = 11643$

read | write R2⋈R3

T ⋈ R1 : $\left\lceil \frac{15000}{7} \right\rceil + \left\lceil \frac{\left\lceil \frac{15000}{7} \right\rceil}{98} \right\rceil \times \frac{10000}{10} = 24143$

read R2⋈R3 and R1

35786 I/O

Q1

- R3 ⋈ (R1 ⋈ R2) actually has a lower cost but this is not within the search space of left deep trees
  - Left-deep trees search space does not guarantee optimal plan

Q2

- Discuss the motivation for the “avoid cross products” heuristics in multi-join query optimization.
- Does it always lead to better plans? Justify your answer.

Q2

- **Discuss the motivation for the “avoid cross products” heuristics in multi-join query optimization.**
- Does it always lead to better plans? Justify your answer.

- Avoid large intermediate results (**more efficient query plan**)
- Excluding cross-product plans also significantly **reduces the plan space**

+ join is one of the most expensive operations in the dbms

Q2

- Discuss the motivation for the “avoid cross products” heuristics in multi-join query optimization.
- **Does it always lead to better plans? Justify your answer.**

Q2

- Not necessarily. Consider this counter example:
  - R1(A, B): tuple size = 10 bytes, 10 tuples
  - R2(D, E): tuple size = 10 bytes, 10 tuples
  - R3(B, E, F, G, H, …): tuple size = 1000 bytes, 10000 tuples
- Query: R1 ⋈ R2 ⋈ R3
  - Non-cross product plans:
    - (R1 ⋈ R3) ⋈ R2
    - (R2 ⋈ R3) ⋈ R1
  - Cross product plans:
    - (R1 ⋈ R2) ⋈ R3

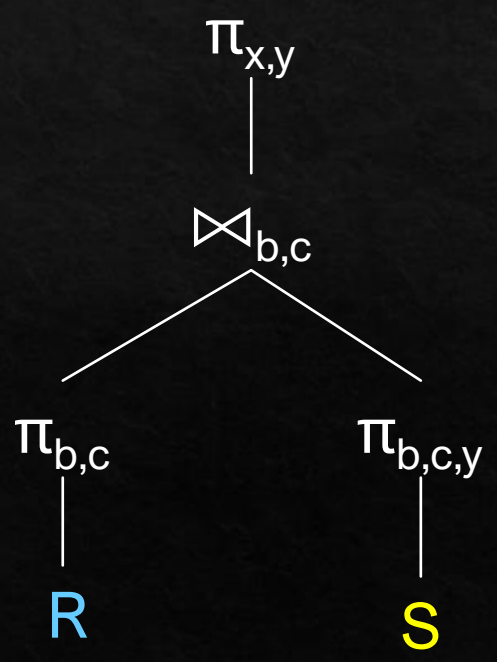
Large intermediate result!

Q3

- π_L(R(a, b, c) ⋈ S(b, c, d, e)) where L is
  - (x, y), x = b + c, y = c + d
  - (a, b, z), z = a + d
- Push down projection as far as possible

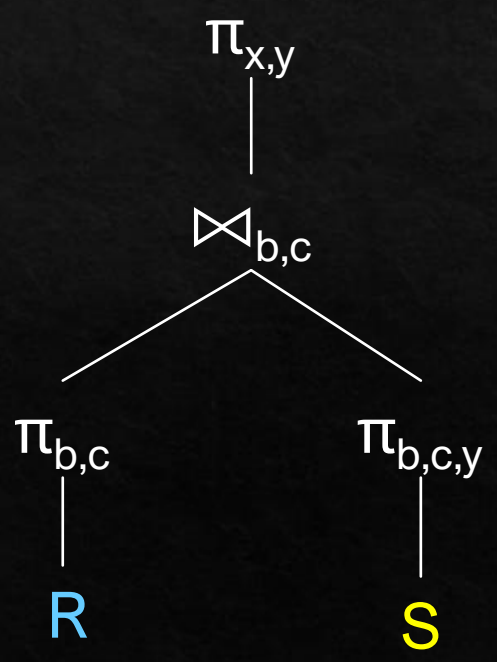
Q3a

- π_L(R(a, b, c) $\bowtie$ S(b, c, d, e)) where L is
  - (x, y), x = b + c, y = c + d



Q3a

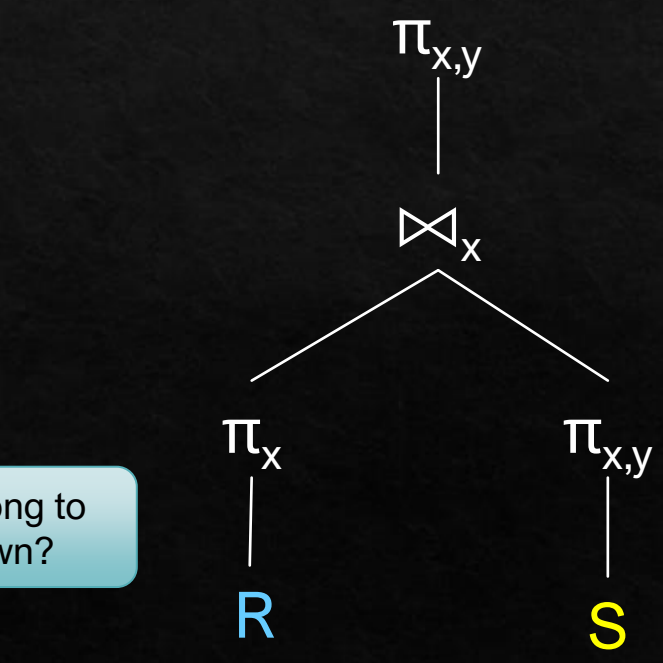
- π_L(R(a, b, c) $\bowtie$ S(b, c, d, e)) where L is
  - (x, y), x = b + c, y = c + d



Why is it wrong to push x down?

Q3a

- π_L(R(a, b, c) ⋈ S(b, c, d, e)) where L is
  - (x, y), x = b + c, y = c + d



Why is it wrong to push x down?

R

b	c
1	3

S

b	c	d
1	3	7
2	2	7

$\pi_x(R)$

x
4

$\pi_{x,y}(S)$

x	y
4	10
4	9

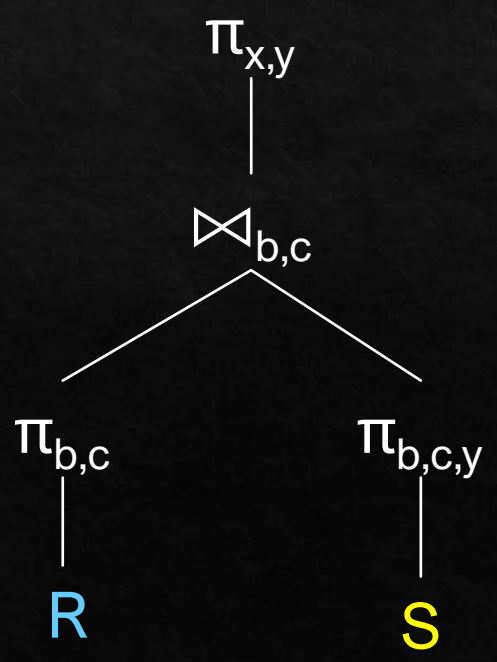
$\pi_{x,y}$

x	y
4	10
4	9

This is wrong! We need to keep b and c for the join

Q3a

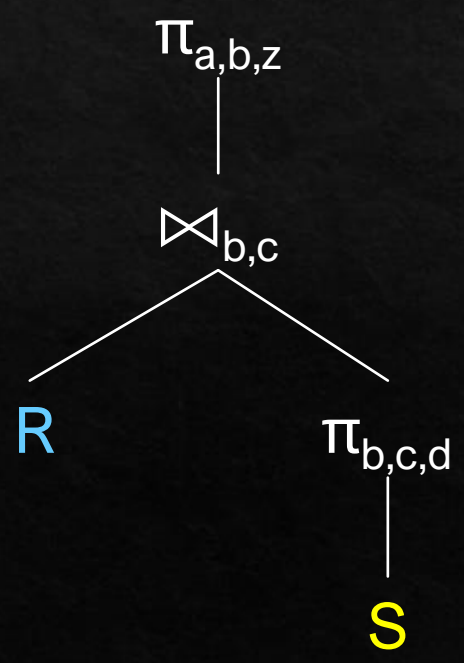
- π_L(R(a, b, c) $\bowtie$ S(b, c, d, e)) where L is
  - (x, y), x = b + c, y = c + d



Actually no cost savings in pushing down y, but question ask to push projection down as far as possible

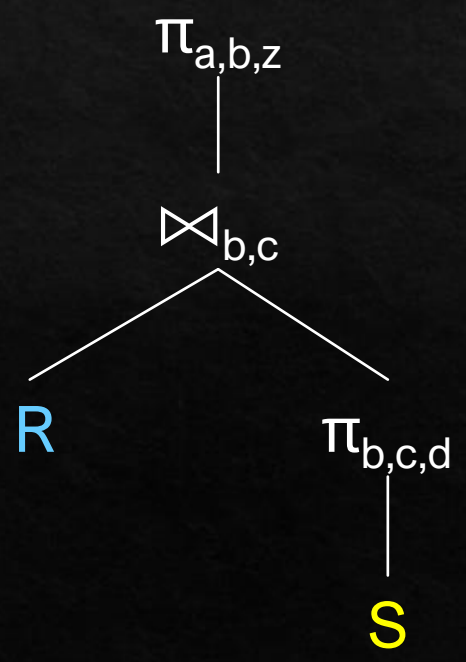
Q3b

- π_L(R(a, b, c) $\bowtie$ S(b, c, d, e)) where L is
  - (a, b, z), z = a + d



Q3b

- π_L(R(a, b, c) ⋈ S(b, c, d, e)) where L is
  - (a, b, z), z = a + d



no way to project z down since there is no 'a' in S

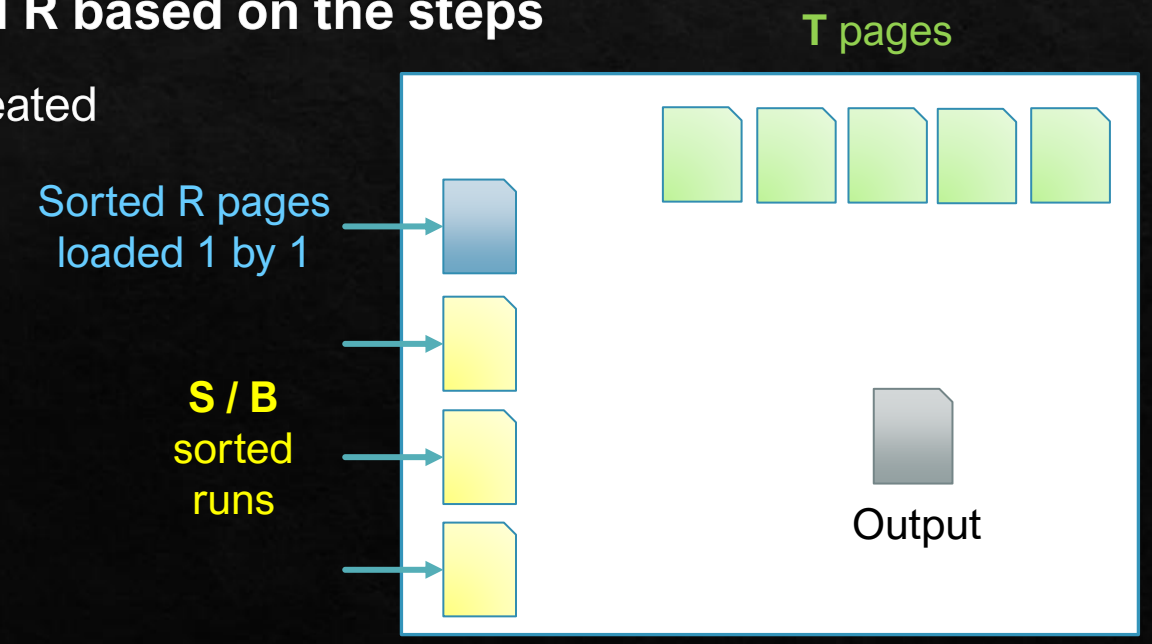
Keep d because you need it to compute z

Q4

- Wish to join R(a, b), S(b, c) and T(c, d) with R, S and T pages respectively
- B buffers (exclude output buffer)
- R is already sorted on b
- **Provide an inequality in terms of B, S, T and R based on the steps**

Q4

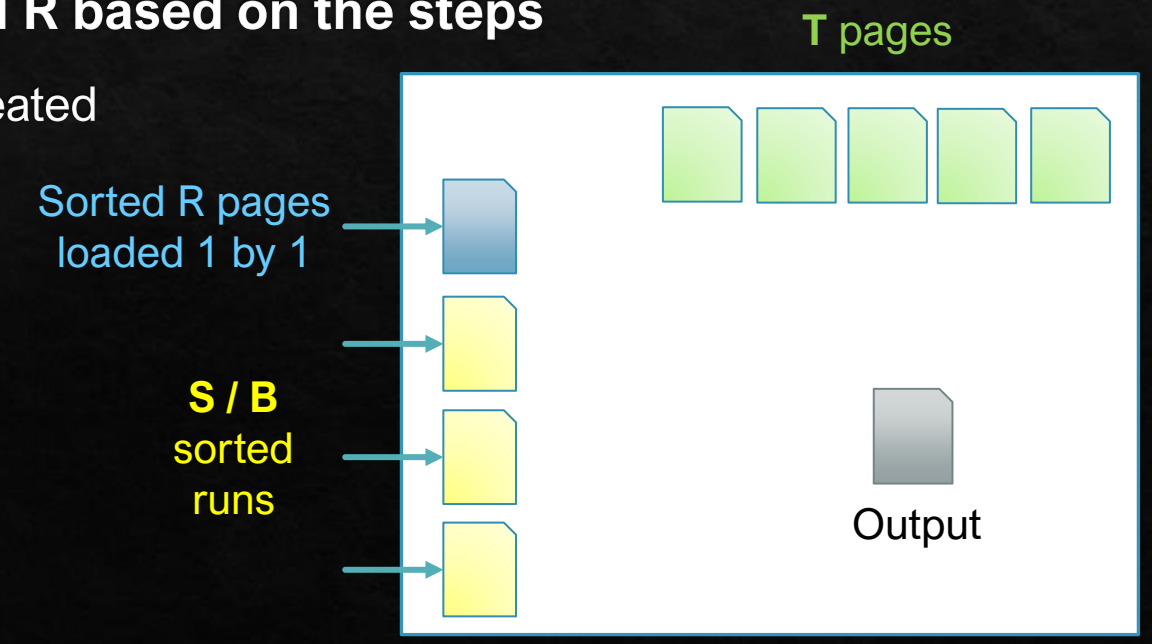
- Wish to join R(a, b), S(b, c) and T(c, d) with **R**, **S** and **T** pages respectively
- B buffers (exclude output buffer)
- R is already sorted on b
- **Provide an inequality in terms of B, S, T and R based on the steps**
  - Generate sorted runs for S: **S**/**B** sorted runs created
  - Load T into memory entirely: **T** buffers needed
  - Merge R and S and T
- $1 + T + \left\lceil \frac{S}{B} \right\rceil \leq B$



Q4

- Wish to join R(a, b), S(b, c) and T(c, d) with **R**, **S** and **T** pages respectively
- B buffers (exclude output buffer)
- R is already sorted on b
- **Provide an inequality in terms of B, S, T and R based on the steps**
  - Generate sorted runs for S: **S/B** sorted runs created
  - Load T into memory entirely: **T** buffers needed
  - Merge R and S and T
- $1 + T + \left\lceil \frac{S}{B} \right\rceil \leq B$
- $B^2 \geq S + BT$

Ignore the 1, multiply B on both sides



Extra

- Let δ denote duplicate elimination. Which of the following statements is TRUE?
  - $\delta(R \times S) = \delta(R) \times \delta(S)$
  - $\delta(R \bowtie S) = \delta(R) \bowtie \delta(S)$
  - $\delta(\sigma_s(R)) = \sigma_s(\delta(R))$

Extra

- All true.
- To prove equality, try to show that LHS is a subset of RHS, and RHS is a subset of LHS

End