

CompSci 516  
Data Intensive Computing Systems

Lecture 9

Query Optimization

Instructor: Sudeepa Roy

# Announcements

- Project proposal due today
  - use template on sakai
  - submit to sakai
- Practice problem set#1 posted on sakai
  - try yourself before looking at the solutions!
- Tomorrow make up lecture “only” for students going to CS grad retreat on Friday
  - Thursday, LSRC D309, 4:40 pm, room accommodates ~10 people
  - Regular class on Friday

# Reading Material

- [RG]
  - Query optimization: Chapter 15 (overview only)
- [GUW]
  - Chapter 16.2-16.7
- Original paper by Selinger et al. :
  - P. Selinger, M. Astrahan, D. Chamberlin, R. Lorie, and T. Price. *Access Path Selection in a Relational Database Management System*  
Proceedings of ACM SIGMOD, 1979. Pages 22-34
  - No need to understand the whole paper, but take a look at the example (link on the course webpage)

Acknowledgement:

Some of the following slides have been created by adapting slides by Profs. Shivnath Babu and Magda Balazinska

# Query Optimization

# Query Blocks: Units of Optimization

- **Query Block**
  - No nesting
  - One SELECT., one FROM
  - At most one WHERE, GROUP BY, HAVING
- SQL query
- => parsed into a collection of query blocks
- => the blocks are optimized one block at a time
- Express single-block it as a relational algebra (RA) expression

```
SELECT S.sname
FROM Sailors S
WHERE S.age IN
  (SELECT MAX (S2.age)
   FROM Sailors S2
   GROUP BY S2.rating)
```

*Outer block*

*Nested block*

# Cost Estimation

- For each plan considered, must estimate cost:
- Must **estimate cost** of each operation in plan tree.
  - Depends on input cardinalities
  - We've discussed how to estimate the cost of operations (sequential scan, index scan, joins, etc.)
- Must also **estimate size of result** for each operation in tree
  - gives input cardinality of next operators
- Also consider whether the output is sorted

# Relational Algebra Equivalences

- Allow us to choose different join orders and to ‘push’ selections and projections ahead of joins.

- Selections:  
$$\sigma_{c1 \wedge \dots \wedge cn}(R) \equiv \sigma_{c1}(\dots \sigma_{cn}(R)) \quad (\text{Cascade})$$
$$\sigma_{c1}(\sigma_{c2}(R)) \equiv \sigma_{c2}(\sigma_{c1}(R)) \quad (\text{Commute})$$
- ❖ Projections:  
$$\pi_{a1}(R) \equiv \pi_{a1}(\dots (\pi_{an}(R))) \quad (\text{Cascade})$$
- ❖ Joins:  
$$R \bowtie (S \bowtie T) \equiv (R \bowtie S) \bowtie T \quad (\text{Associative})$$
$$(R \bowtie S) \equiv (S \bowtie R) \quad (\text{Commute})$$

There are many more intuitive equivalences, see 15.3.4 for details

# Notation

- $T(R)$  : Number of tuples in  $R$
- $B(R)$  : Number of blocks in  $R$
- $V(R, A)$  : Number of distinct values of attribute  $A$  in  $R$



# Query Optimization Problem

Pick the best plan from the space of  
physical plans

# Cost-based Query Optimization

Pick the plan with least cost

Challenge:

- Do not want to execute more than one plans
- Need to estimate the cost without executing the plan

“heuristic-based” optimizer (e.g. push selections down) have limited power and not used much

# Cost-based Query Optimization

Pick the plan with least cost

Tasks:

1. Estimate the cost of individual operators  
done in Lecture 9
2. Estimate the size of output of individual operators  
today
3. Combine costs of different operators in a plan  
today
4. Efficiently search the space of plans  
today

# Task 1 and 2

## Estimating cost and size of different operators

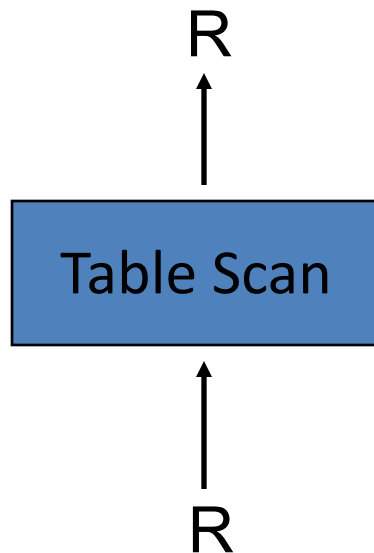
- Size = #tuples, NOT #pages
- Cost = #page I/O
  - but, need to consider whether the intermediate relation fits in memory, is written back to/read from disk (or on-the-fly goes to the next operator), etc.

# Desired Properties of Estimating Sizes of Intermediate Relations

Ideally,

- should give accurate estimates (as much as possible)
- should be easy to compute
- should be logically consistent
  - size estimate should be independent of how the relation is computed
  - e.g. which join algorithm/join order is used
- But, no “universally agreed upon” ways to meet these goals

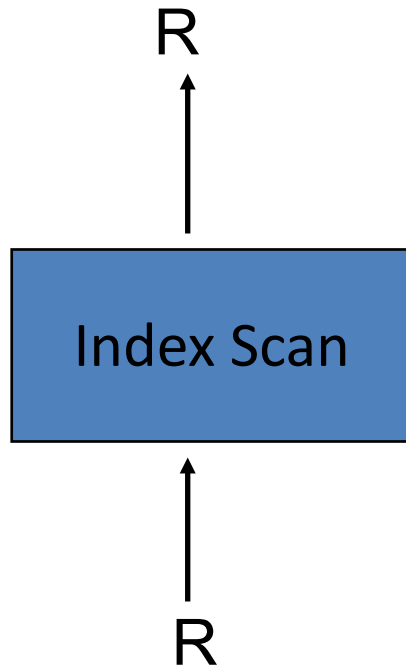
# Cost of Table Scan



Cost:  $B(R)$   
Size:  $T(R)$

$T(R)$  : Number of tuples in  $R$   
 $B(R)$  : Number of blocks in  $R$

# Cost of Index Scan



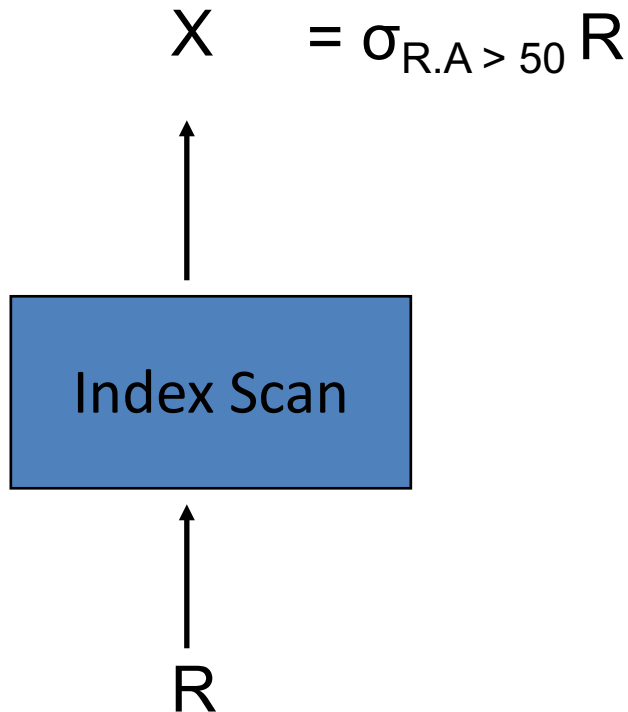
Cost:  $B(R)$  – if clustered  
 $T(R)$  – if unclustered

Size:  $T(R)$

$T(R)$  : Number of tuples in  $R$   
 $B(R)$  : Number of blocks in  $R$

Note: size is independent of the implementation of the scan/index

# Cost of Index Scan with Selection



Cost:  $B(R) * f$  – if clustered  
 $T(R) * f$  – if unclustered

Size:  $T(R) * f$

$T(R)$  : Number of tuples in  $R$   
 $B(R)$  : Number of blocks in  $R$

Reduction factor

$$f = (\text{Max}(R.A) - 50) / (\text{Max}(R.A) - \text{Min}(R.A))$$

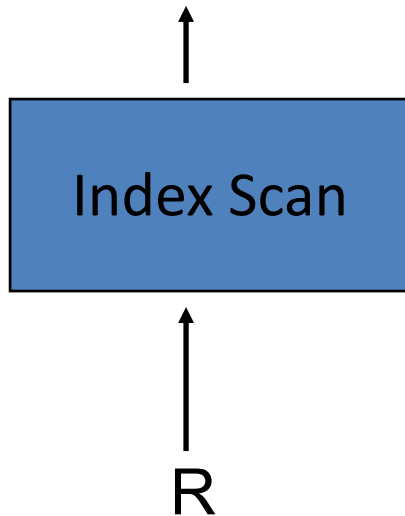
assumes uniform distribution



# Cost of Index Scan with Selection (and multiple conditions)

$$X = \sigma_{R.A > 50 \text{ and } R.B = C} R$$

assume index on  
(A, B)



What is  $f_1$  if the first condition is  $100 > R.1 > 50$ ?

Cost:  $B(R) * f$  – if clustered  
 $T(R) * f$  – if unclustered

Size:  $T(R) * f$

Reduction factors

range selection

$$f_1 = \text{Max}(R.A) - 50 / (\text{Max}(R.A) - \text{Min}(R.A))$$

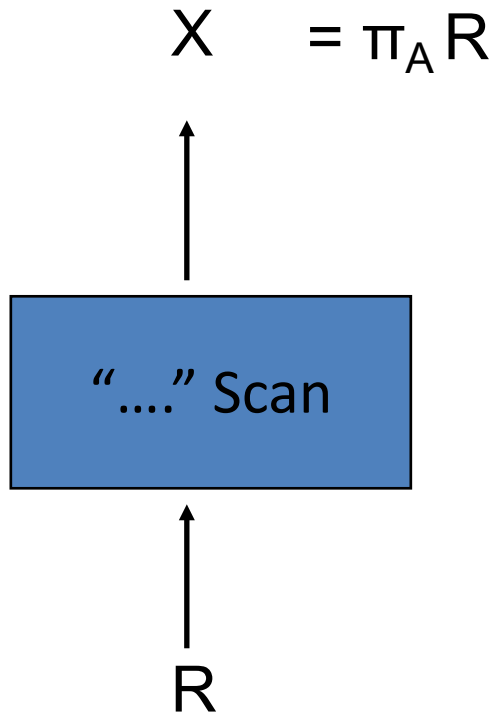
$$f_2 = 1 / V(R, B)$$

value selection

$$f = f_1 * f_2 \text{ (assumes independence and uniform distribution)}$$

$T(R)$  : Number of tuples in  $R$   
 $B(R)$  : Number of blocks in  $R$   
 $V(R, A)$  : Number of distinct values of attribute  $A$  in  $R$

# Cost of Projection



Cost: depends on the method of scanning  $R$

$B(R)$  for table scan or clustered index scan

Size:  $T(R)$

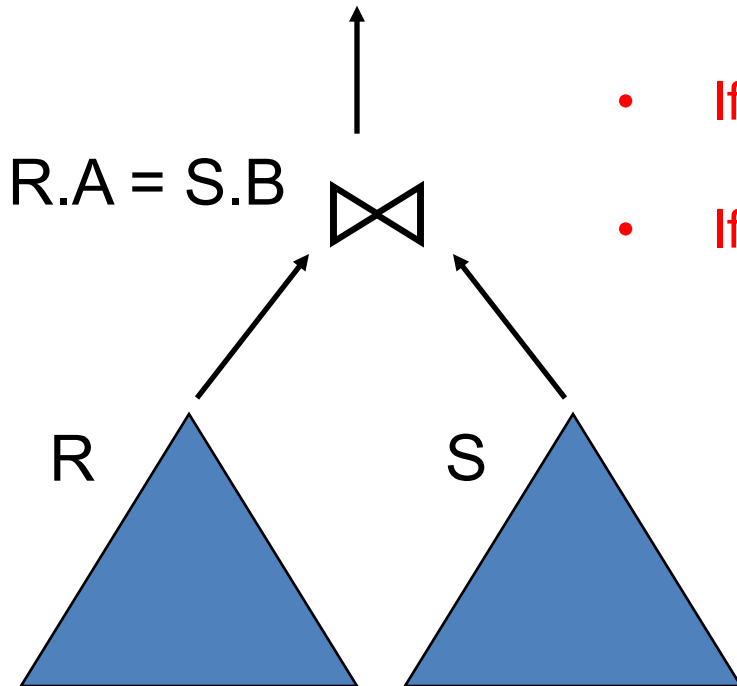
But tuples are smaller

If you have more information on the size of the smaller tuples, can estimate #I/O better

# Size of Join

Quite tricky

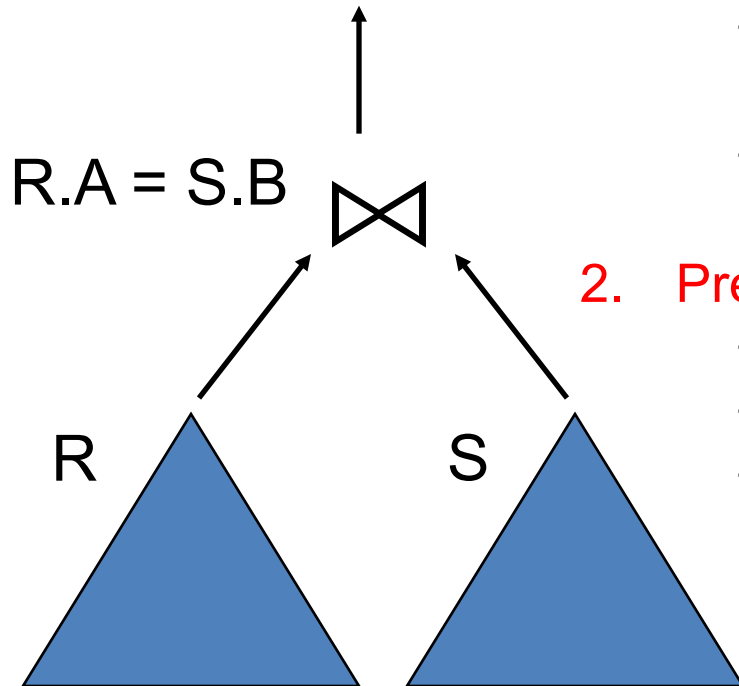
- If disjoint A and B values
  - then 0
- If A is key of R and B is foreign key of S
  - then  $T(S)$
- If all tuples have the same value of  $R.A = S.B = x$ 
  - then  $T(R) * T(S)$



$T(R)$  : Number of tuples in R  
 $B(R)$  : Number of blocks in R  
 $V(R, A)$  : Number of distinct values of attribute A in R

# Size of Join

## Two assumptions



### 1. Containment of value sets:

- if  $V(R, A) \leq V(S, B)$ , then all A-values of R are included in B-values of S
- e.g. satisfied when A is foreign key, B is key

### 2. Preservation of value sets:

- $V(R \bowtie S, A) = V(R, A)$
- $V(R \bowtie S, B) = V(S, B)$
- No value is lost in join

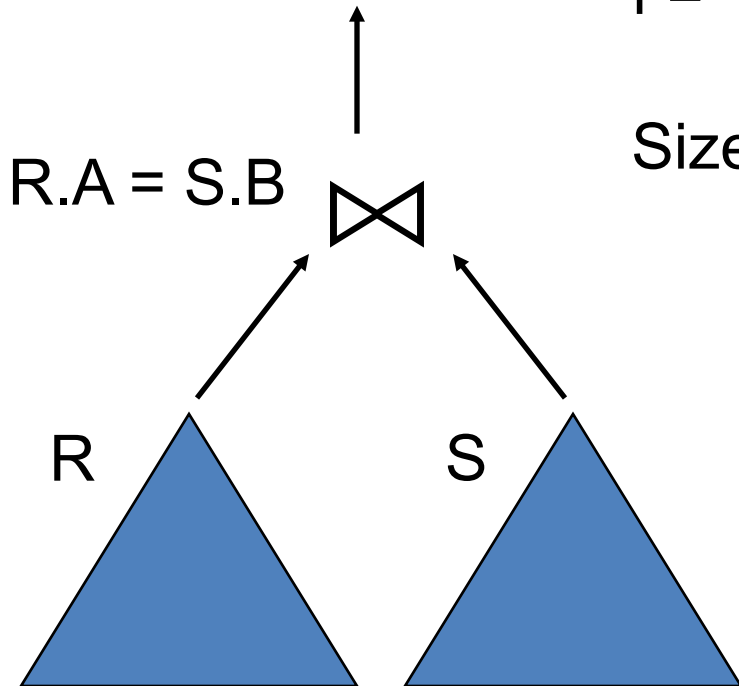
T (R) : Number of tuples in R  
B (R) : Number of blocks in R  
V(R, A) : Number of distinct values of attribute A in R

# Size of Join

Reduction factor

$$f = 1/\max(V(R, A), V(S, B))$$

$$\text{Size} = T(R) * T(S) * f$$



T (R) : Number of tuples in R  
B (R) : Number of blocks in R  
V(R, A) : Number of distinct values of attribute A in R

Assumes index on both A and B

if one index:  $1/V(\dots, \dots)$

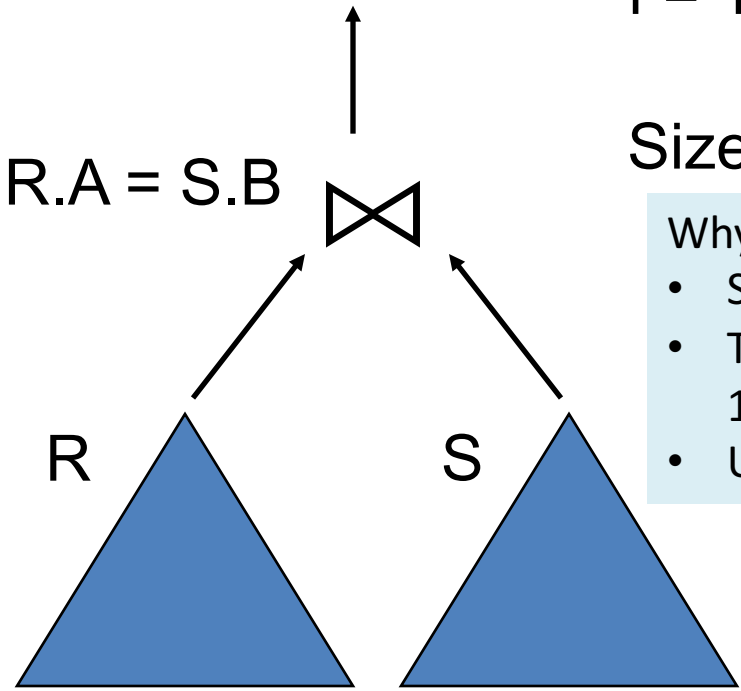
if no index: say  $1/10$

# Size of Join

## Reduction factor

$$f = 1/\max(V(R, A), V(S, B))$$

$$\text{Size} = T(R) * T(S) * f$$



### Why max?

- Suppose  $V(R, A) \leq V(S, B)$
- The probability of a A-value joining with a B-value is  $1/V(S, B) = \text{reduction factor}$
- Under the two assumptions stated earlier + uniformity

T (R) : Number of tuples in R  
 B (R) : Number of blocks in R  
 V(R, A) : Number of distinct values of attribute A in R

# Task 3: Combine cost of different operators in a plan

With Examples

“Given” the physical plan

- Size = #tuples, NOT #pages
- Cost = #page I/O
  - but, need to consider whether the intermediate relation fits in memory, is written back to disk (or on-the-fly goes to the next operator) etc.

# Example Query

Student (sid, name, age, address)

Book(bid, title, author)

Checkout(sid, bid, date)

Query:

```
SELECT S.name
FROM Student S, Book B, Checkout C
WHERE S.sid = C.sid
AND B.bid = C.bid
AND B.author = 'Olden Fames'
AND S.age > 12
AND S.age < 20
```



S(sid,name,age,addr)  
B(bid,title,author)  
C(sid,bid,date)

# Assumptions

- Student: S, Book: B, Checkout: C
- Sid, bid foreign key in C referencing S and B resp.
- There are 10,000 Student records stored on 1,000 pages.
- There are 50,000 Book records stored on 5,000 pages.
- There are 300,000 Checkout records stored on 15,000 pages.
- There are 500 different authors.
- Student ages range from 7 to 24.

Warning: a few dense slides next 😊

S(sid,name,age,addr)

T(S)=10,000

B(S)=1,000

V(B,author) = 500

B(bid,title,author)

T(B)=50,000

B(B)=5,000

7 <= age <= 24

C(sid,bid,date)

T(C)=300,000

B(C)=15,000

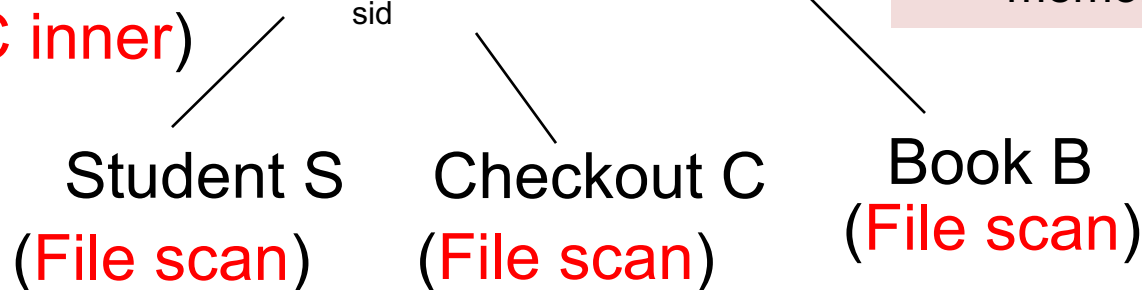
# Physical Query Plan – 1

(On the fly) (d)  $\Pi_{name}$

(On the fly) (c)  $\sigma_{12 < age < 20 \wedge author = 'Olden Fames'}$

(Tuple-based nested loop  
B inner)

(Page-oriented  
-nested loop,  
S outer, C inner)



## Q. Compute

1. the cost and cardinality in steps (a) to (d)
2. the total cost

## Assumptions:

- Data is not sorted on any attributes
- For both in (a) and (b), outer relations fit in memory

S(sid,name,age,addr)

T(S)=10,000

B(S)=1,000

V(B,author) = 500

B(bid,title,author)

T(B)=50,000

B(B)=5,000

7 <= age <= 24

C(sid,bid,date)

T(C)=300,000

B(C)=15,000

(a)

(On the fly) (d)  $\Pi_{name}$

(On the fly) (C)  $\sigma_{12 < age < 20 \wedge author = 'Olden Fames'}$

**Cost =**

$$B(S) + B(S) * B(C)$$

$$= 1000 + 1000 * 15000$$

$$= 15,001,000$$

**Cardinality =**

$$T(C) = 300,000$$

- foreign key join, output pipelined to next join
- Can apply the formula as well

$$T(S) * T(C) / \max(V(S, sid), V(C, sid))$$

$$= T(C)$$

since  $V(S, sid) \geq V(C, sid)$   
and

$$T(S) = V(S, sid)$$

(Tuple-based nested loop  
B inner)

(Page-oriented  
-nested loop,  
S outer, C inner)

Student S  
(File scan)

Checkout C  
(File scan)

Book B  
(File scan)

S(sid,name,age,addr)

T(S)=10,000

B(S)=1,000

V(B,author) = 500

B(bid,title,author)

T(B)=50,000

B(B)=5,000

7 <= age <= 24

C(sid,bid,date)

T(C)=300,000

B(C)=15,000

(b)

(On the fly) (d)  $\Pi_{name}$

(On the fly) (C)  $\sigma_{12 < \text{age} < 20 \wedge \text{author} = \text{'Olden Fames'}}$

**Cost =**

$T(S \bowtie C) * B(B)$

$= 300,000 * 5,000 = 15 * 10^8$

**Cardinality =**

$T(S \bowtie C) = 300,000$

- foreign key join, don't need scanning for outer relation

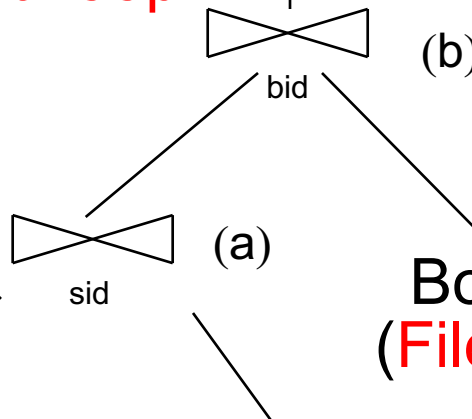
(Tuple-based nested loop  
B inner)

(Page-oriented  
-nested loop,  
S outer, C inner)

Student S  
(File scan)

Checkout C  
(File scan)

Book B  
(File scan)



S(sid,name,age,addr)

T(S)=10,000

B(S)=1,000

V(B,author) = 500

B(bid,title,author)

T(B)=50,000

B(B)=5,000

7 <= age <= 24

C(sid,bid,date)

T(C)=300,000

B(C)=15,000

(c, d)

(On the fly) (d)  $\Pi_{name}$

(On the fly) (C)  $\sigma_{12 < age < 20 \wedge author = 'Olden Fames'}$

**Cost =**

**0** (on the fly)

**Cardinality =**

$300,000 * 1/500 * 7/18$

$= 234$  (approx)

(assuming uniformity and independence)

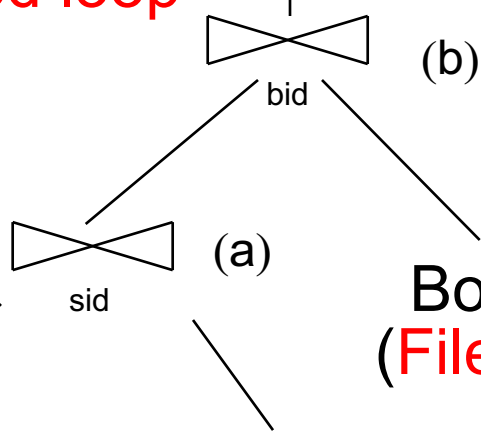
(Tuple-based nested loop  
B inner)

(Page-oriented  
-nested loop,  
S outer, C inner)

Student S  
(File scan)

Checkout C  
(File scan)

Book B  
(File scan)



S(sid,name,age,addr)

T(S)=10,000

B(S)=1,000

V(B,author) = 500

B(bid,title,author)

T(B)=50,000

B(B)=5,000

7 <= age <= 24

C(sid,bid,date)

T(C)=300,000

B(C)=15,000

(Total)

**Total cost =**

1,515,001,000

**Final cardinality =**

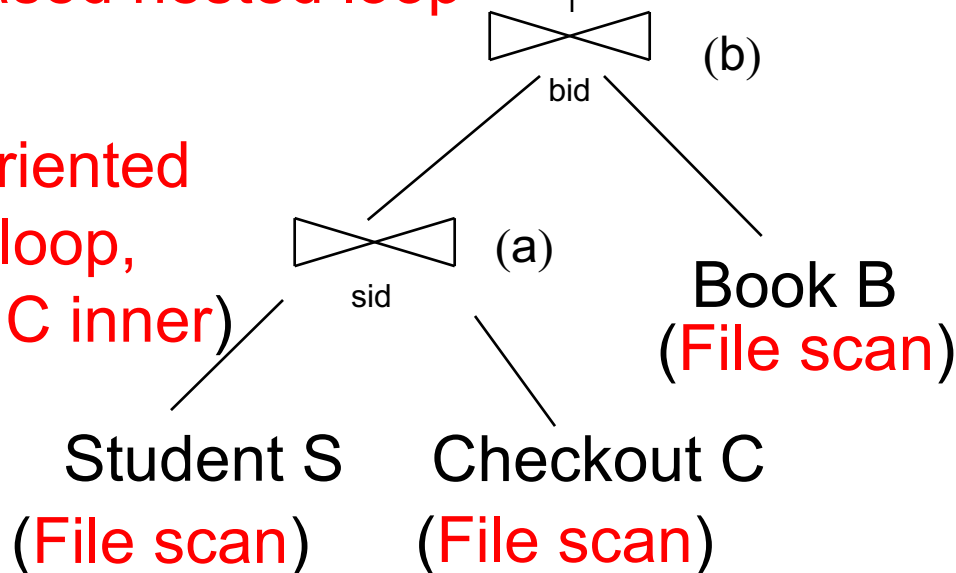
234 (approx)

(On the fly) (d)  $\Pi_{name}$

(On the fly) (c)  $\sigma_{12 < age < 20 \wedge author = 'Olden Fames'}$

(Tuple-based nested loop  
B inner)

(Page-oriented  
-nested loop,  
S outer, C inner)



S(sid,name,age,addr)

T(S)=10,000

B(S)=1,000

V(B,author) = 500

B(bid,title,author)

T(B)=50,000

B(B)=5,000

7 <= age <= 24

C(sid,bid,date)

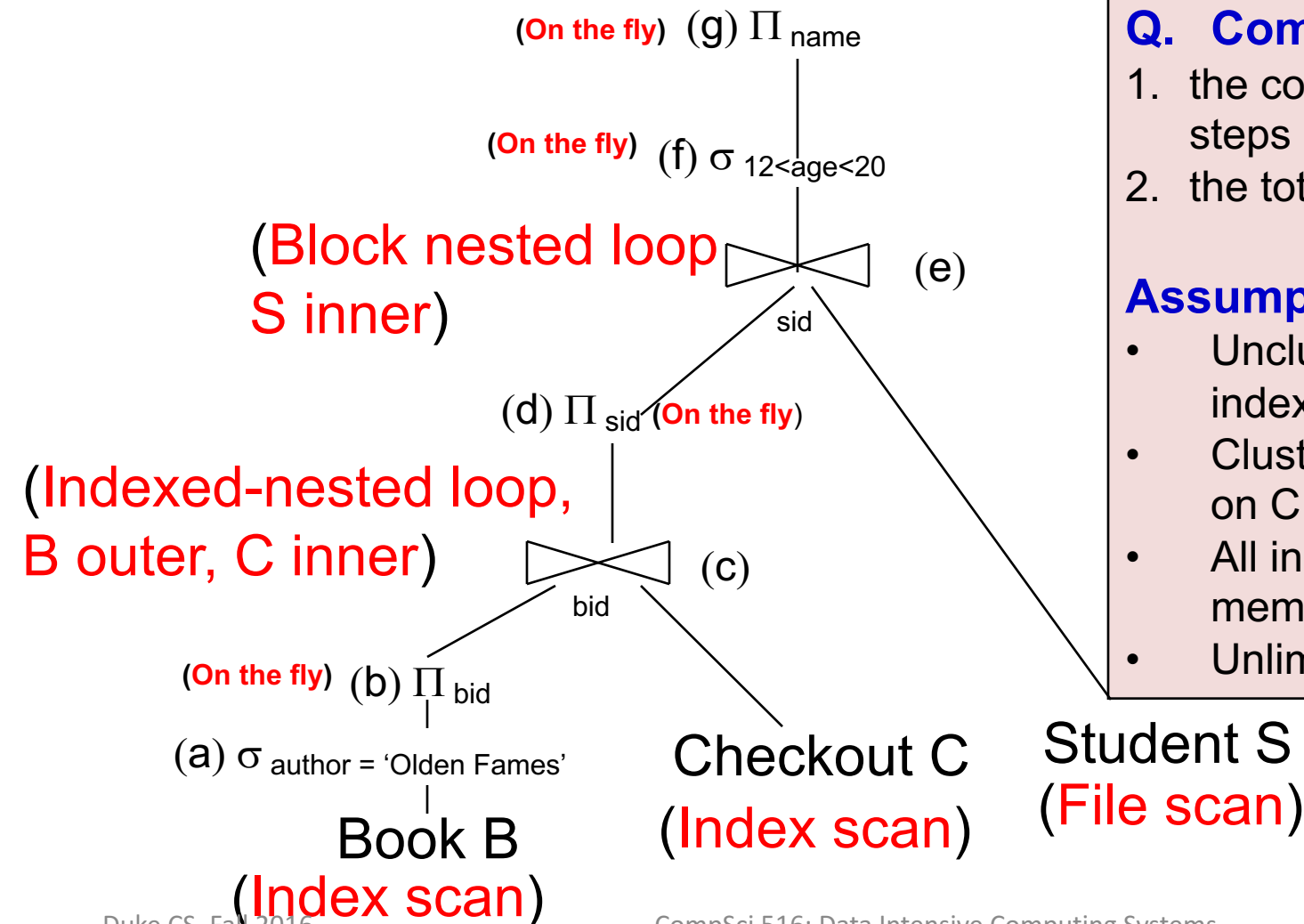
T(C)=300,000

B(C)=15,000

V(B,author) = 500

7 <= age <= 24

# Physical Query Plan – 2



## Q. Compute

1. the cost and cardinality in steps (a) to (g)
2. the total cost

## Assumptions:

- Unclustered B+tree index on B.author
- Clustered B+tree index on C.bid
- All index pages are in memory
- Unlimited memory

S(sid,name,age,addr)

T(S)=10,000

B(S)=1,000

V(B,author) = 500

B(bid,title,author): Un. B+ on author

T(B)=50,000

B(B)=5,000

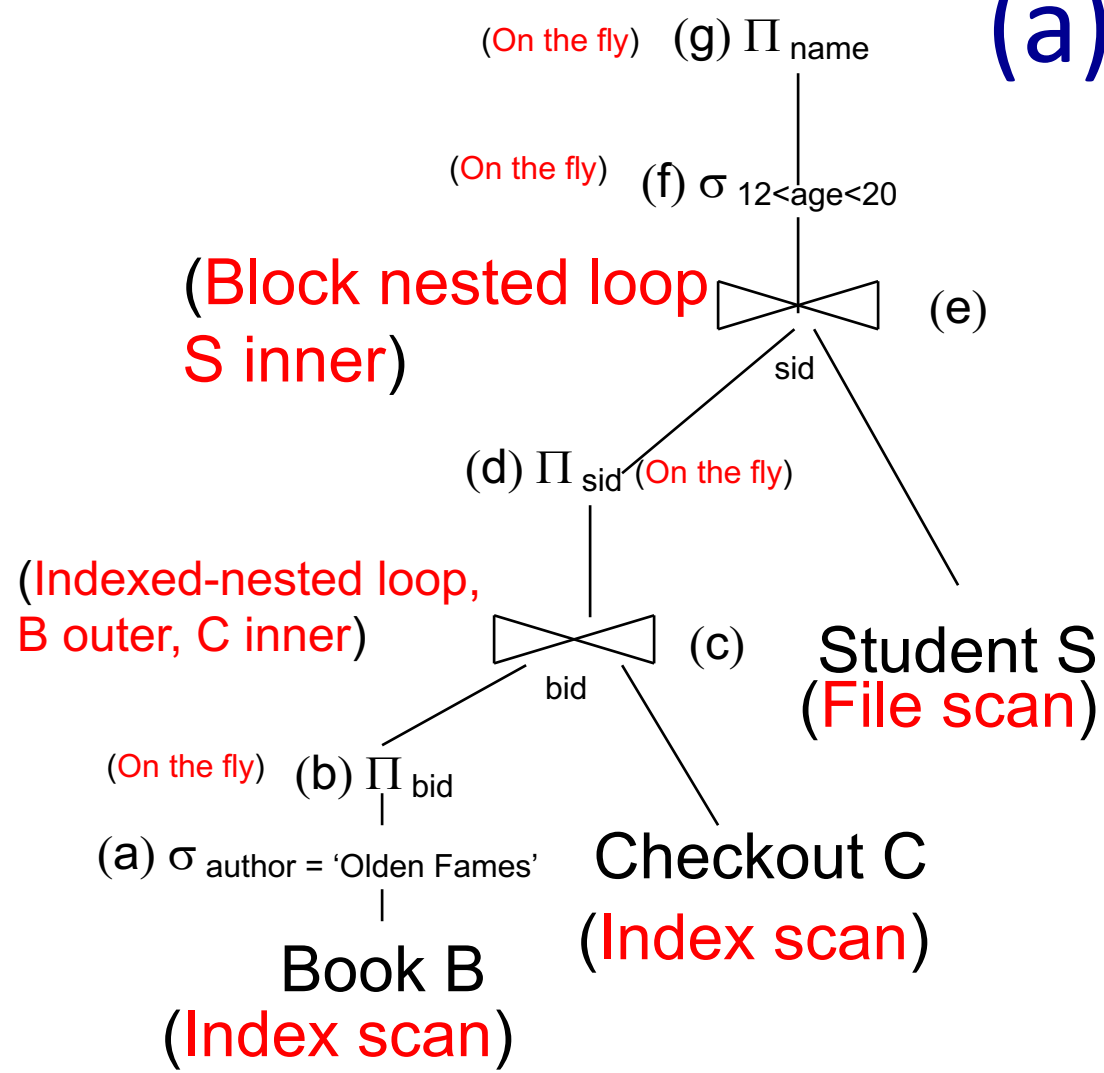
7 <= age <= 24

C(sid,bid,date): Cl. B+ on bid

T(C)=300,000

B(C)=15,000

(a)



**Cost =**  
 $T(B) / V(B, \text{author})$   
 $= 50,000 / 500$   
 $= 100$  (unclustered)

**Cardinality =**  
 100



S(sid,name,age,addr)

T(S)=10,000

B(S)=1,000

V(B,author) = 500

B(bid,title,author): Un. B+ on author

T(B)=50,000

B(B)=5,000

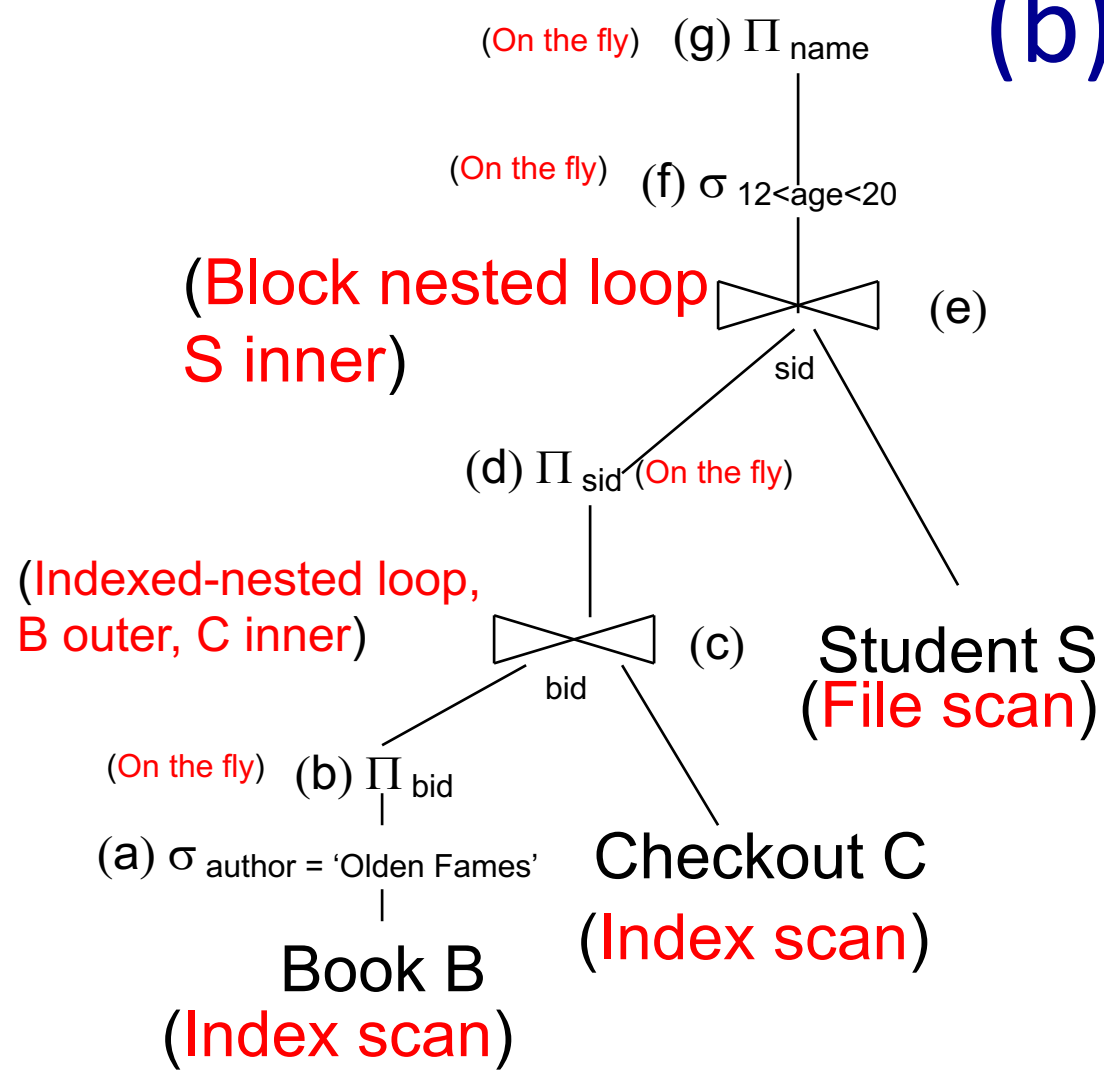
7 <= age <= 24

C(sid,bid,date): Cl. B+ on bid

T(C)=300,000

B(C)=15,000

(b)



**Cost =**  
 0 (on the fly)

**Cardinality =**  
 100

S(sid,name,age,addr)

T(S)=10,000

B(S)=1,000

V(B,author) = 500

B(bid,title,author): Un. B+ on author

T(B)=50,000

B(B)=5,000

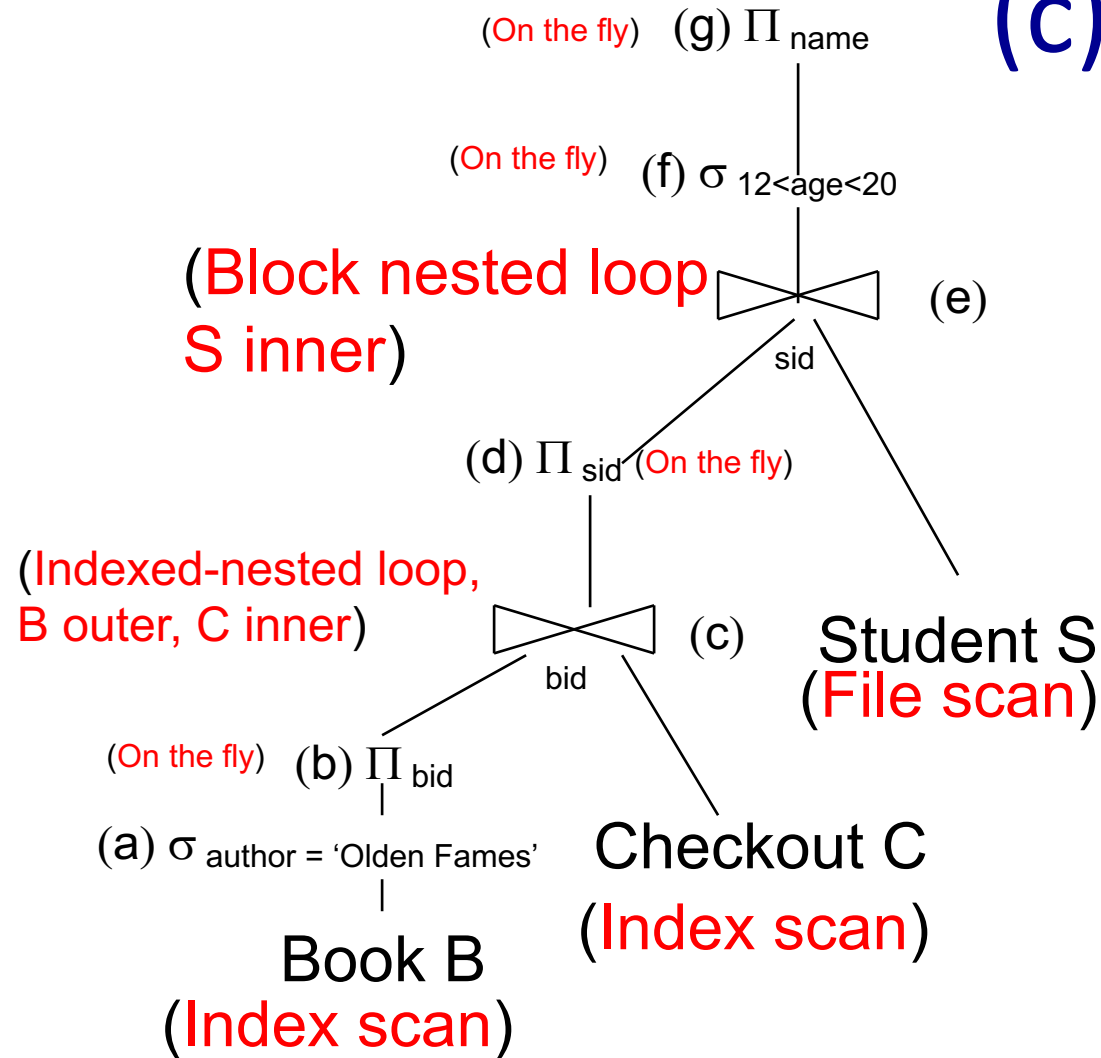
7 <= age <= 24

C(sid,bid,date): Cl. B+ on bid

T(C)=300,000

B(C)=15,000

(c)



- one index lookup per outer B tuple
- 1 book has  $T(C)/T(B) = 6$  checkouts (uniformity)
- # C tuples per page =  $T(C)/B(C) = 20$
- 6 tuples fit in at most 2 consecutive pages (clustered) could assume 1 page as well

**Cost <=**

$100 * 2 = 200$

**Cardinality =**

$100 * 6 = 600$

$= 100 * T(C) / \text{MAX}(100, V(C, bid))$   
 assuming  
 $V(C, bid) = V(B, bid) = T(B) = 50,000$

S(sid,name,age,addr)

T(S)=10,000

B(S)=1,000

V(B,author) = 500

B(bid,title,author): Un. B+ on author

T(B)=50,000

B(B)=5,000

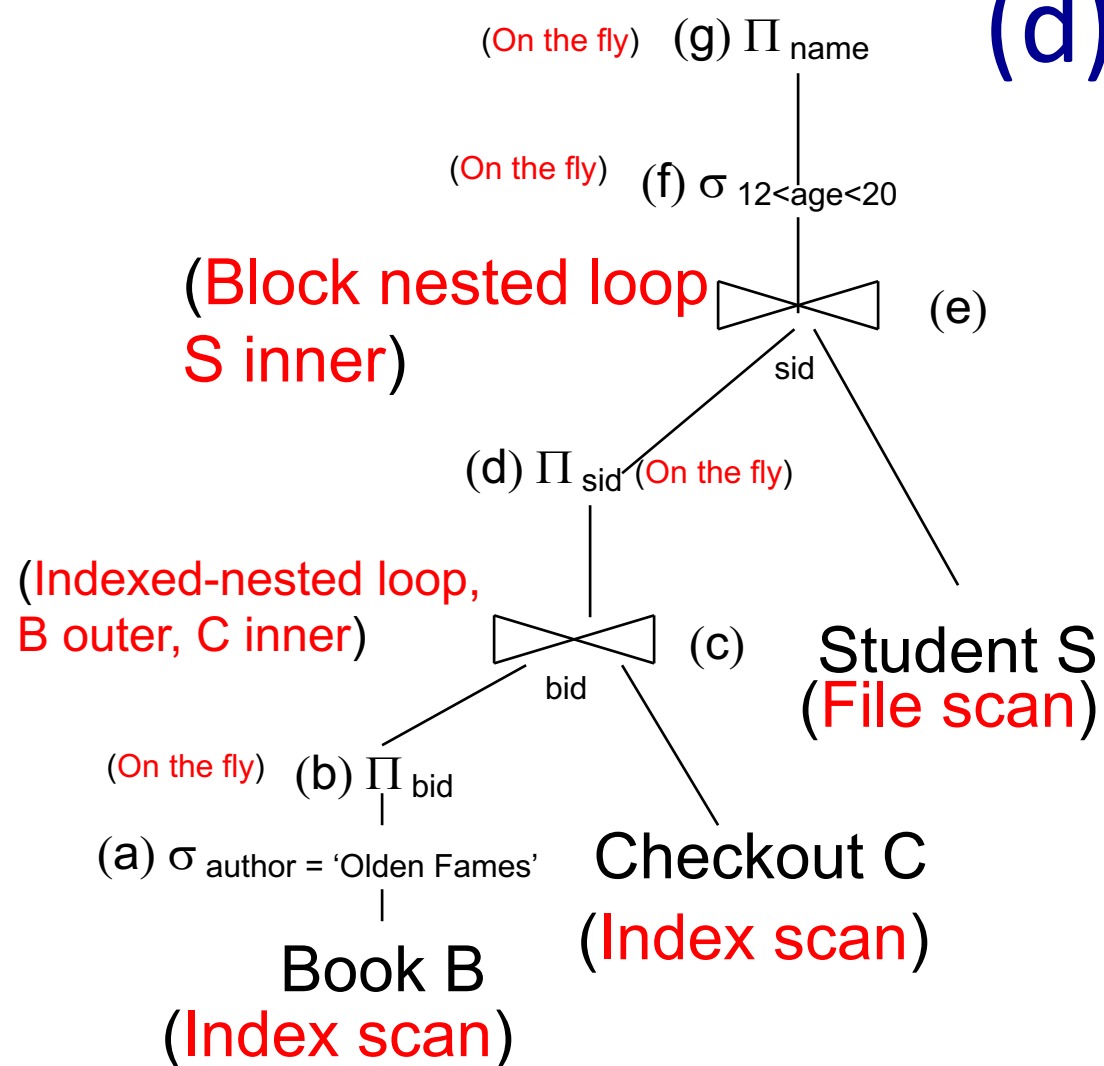
7 <= age <= 24

C(sid,bid,date): Cl. B+ on bid

T(C)=300,000

B(C)=15,000

(d)



**Cost =**  
**0 (on the fly)**

**Cardinality =**  
**600**

S(sid,name,age,addr)

T(S)=10,000

B(S)=1,000

V(B,author) = 500

B(bid,title,author): Un. B+ on author

T(B)=50,000

B(B)=5,000

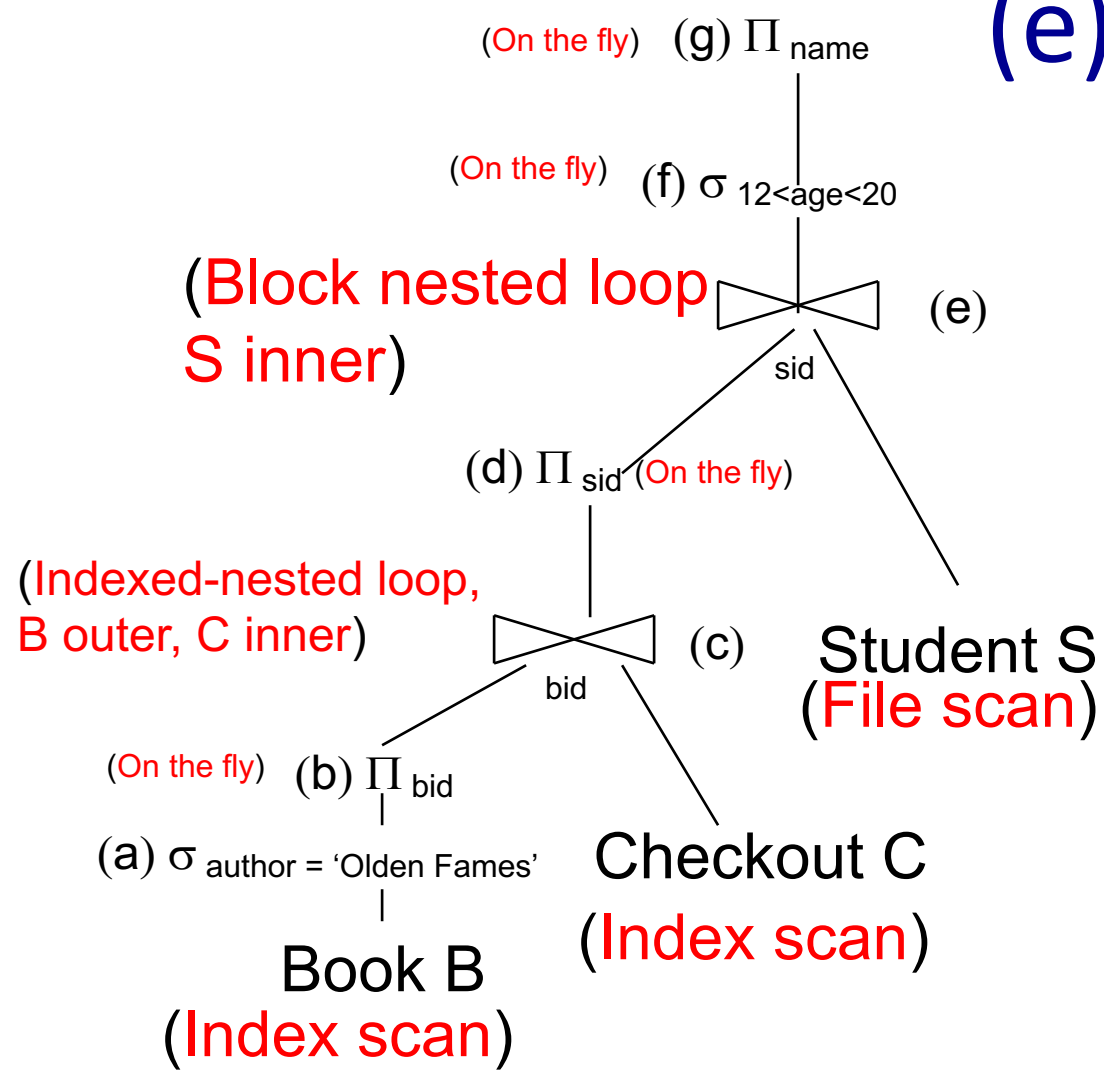
7 <= age <= 24

C(sid,bid,date): Cl. B+ on bid

T(C)=300,000

B(C)=15,000

(e)



Outer relation is already in (unlimited) memory need to scan S relation

**Cost =**  
 $B(S) = 1000$

**Cardinality =**  
 600

S(sid,name,age,addr)

T(S)=10,000

B(S)=1,000

V(B,author) = 500

B(bid,title,author): Un. B+ on author

T(B)=50,000

B(B)=5,000

7 <= age <= 24

C(sid,bid,date): Cl. B+ on bid

T(C)=300,000

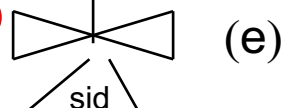
B(C)=15,000

(f)

(On the fly) (g)  $\Pi_{name}$

(On the fly) (f)  $\sigma_{12 < age < 20}$

(Block nested loop  
S inner)

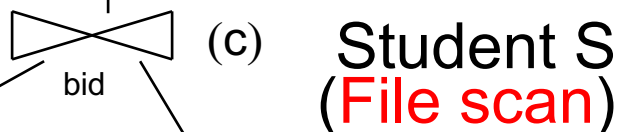


**Cost =**  
**0** (on the fly)

**Cardinality =**  
**600 \* 7/18 = 234** (approx)

(d)  $\Pi_{sid}$  (On the fly)

(Indexed-nested loop,  
B outer, C inner)



(On the fly) (b)  $\Pi_{bid}$

(a)  $\sigma_{author = 'Olden Fames'}$

Checkout C  
(Index scan)

Book B

(Index scan)

S(sid,name,age,addr)

T(S)=10,000

B(S)=1,000

V(B,author) = 500

B(bid,title,author): Un. B+ on author

T(B)=50,000

B(B)=5,000

7 <= age <= 24

C(sid,bid,date): Cl. B+ on bid

T(C)=300,000

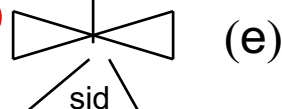
B(C)=15,000

(g)

(On the fly) (g)  $\Pi_{name}$

(On the fly) (f)  $\sigma_{12 < age < 20}$

(Block nested loop  
S inner)

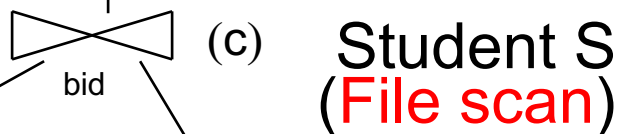


**Cost =**  
**0 (on the fly)**

**Cardinality =**  
**234**

(d)  $\Pi_{sid}$  (On the fly)

(Indexed-nested loop,  
B outer, C inner)



(On the fly) (b)  $\Pi_{bid}$

(a)  $\sigma_{author = 'Olden Fames'}$



Book B

(Index scan)

S(sid,name,age,addr)

T(S)=10,000

B(S)=1,000

V(B,author) = 500

B(bid,title,author): Un. B+ on author

T(B)=50,000

B(B)=5,000

7 <= age <= 24

C(sid,bid,date): Cl. B+ on bid

T(C)=300,000

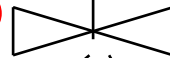
B(C)=15,000

(total)

(On the fly) (g)  $\Pi_{name}$

(On the fly) (f)  $\sigma_{12 < age < 20}$

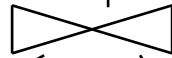
(Block nested loop  
S inner)



(e)

sid

(d)  $\Pi_{sid}$  (On the fly)



(c)

Student S  
(File scan)

bid

(On the fly) (b)  $\Pi_{bid}$

(a)  $\sigma_{author = 'Olden Fames'}$

Checkout C  
(Index scan)

Book B

(Index scan)

**Total cost =  
1300**

(compare with 1,515,001,000  
for plan 1!)

**Final cardinality =  
234 (approx)**  
(same as plan 1!)

# Task 4:

## Efficiently searching the plan space

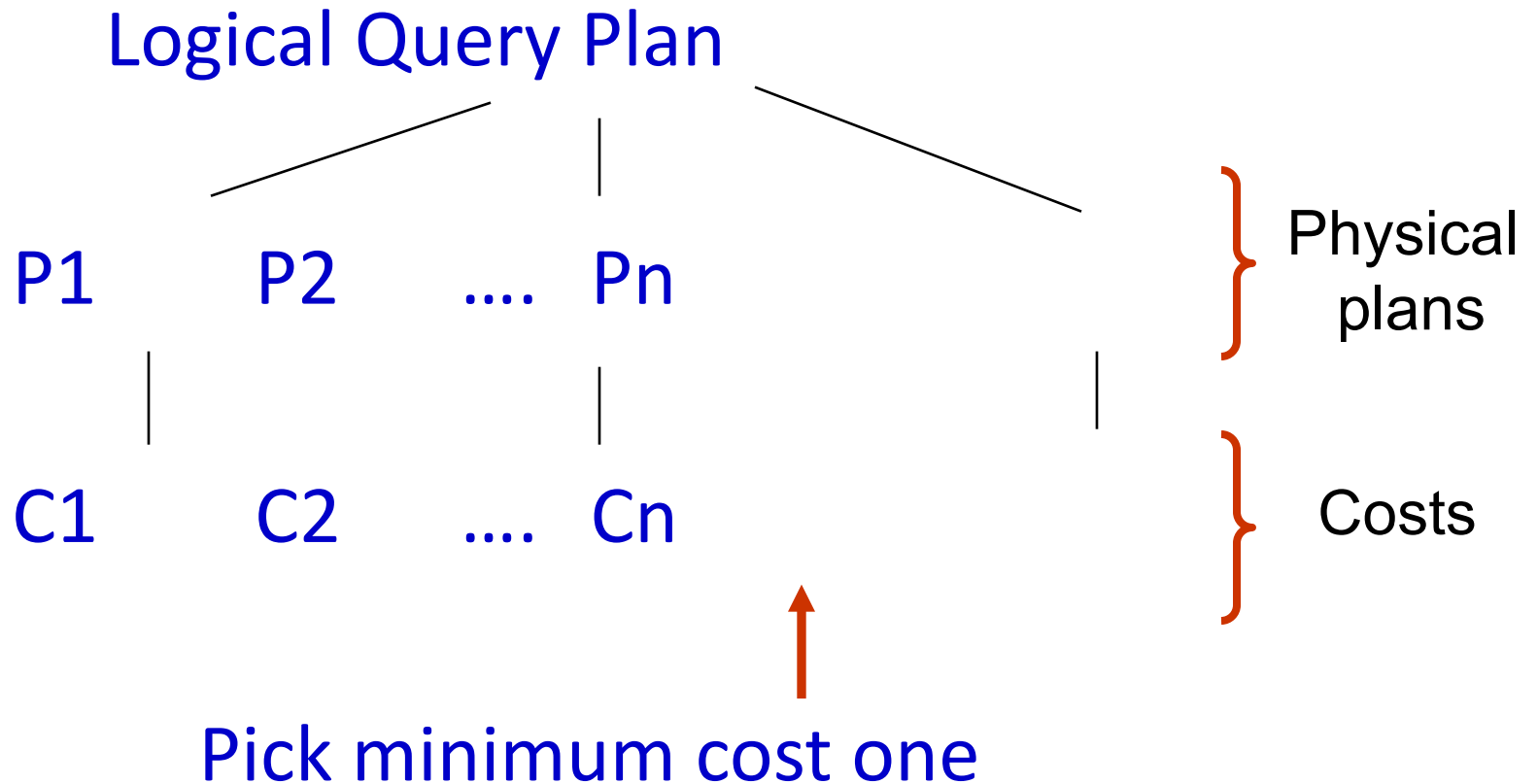
Use dynamic-programming based  
Selinger's algorithm



# Heuristics for pruning plan space

- Predicates as early as possible
- Avoid plans with cross products
- Only left-deep join trees

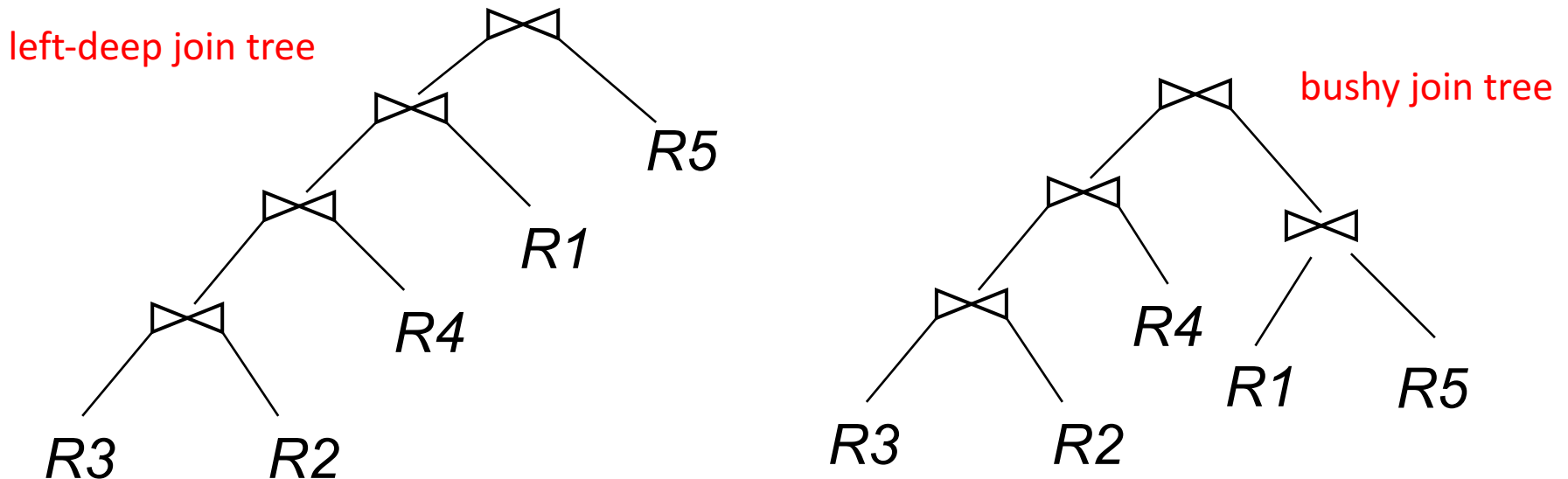
# Physical Plan Selection



# Join Trees

Query:  $R1 \bowtie R2 \bowtie R3 \bowtie R4 \bowtie R5$

---



(logical plan space)

- Several possible structure of the trees
- Each tree can have  $n!$  permutations of relations

(physical plan space)

- Different implementation and scanning of intermediate operators for each logical plan

# Selinger Algorithm

- **Dynamic Programming** based
- **Dynamic Programming:**
  - General algorithmic paradigm
  - Exploits “principle of optimality”
    - Useful reading: Chapter 16, Introduction to Algorithms, Cormen, Leiserson, Rivest
- **Considers the search space of left-deep join trees**
  - reduces search space (only one structure), still  $n!$  permutations
  - interacts well with join algos (esp. NLJ)
  - e.g. might not need to write tuples to disk if enough memory

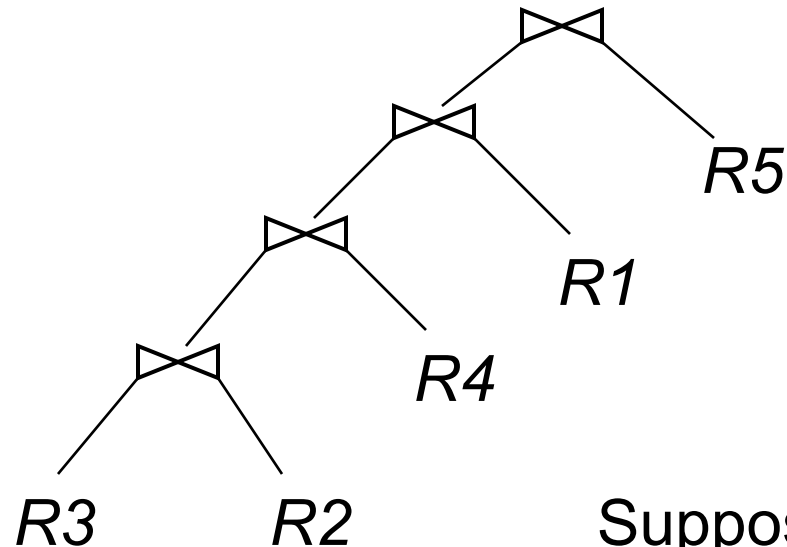
# Principle of Optimality

Optimal for “whole” made up from  
optimal for “parts”

# Principle of Optimality

Query:  $R1 \bowtie R2 \bowtie R3 \bowtie R4 \bowtie R5$

---

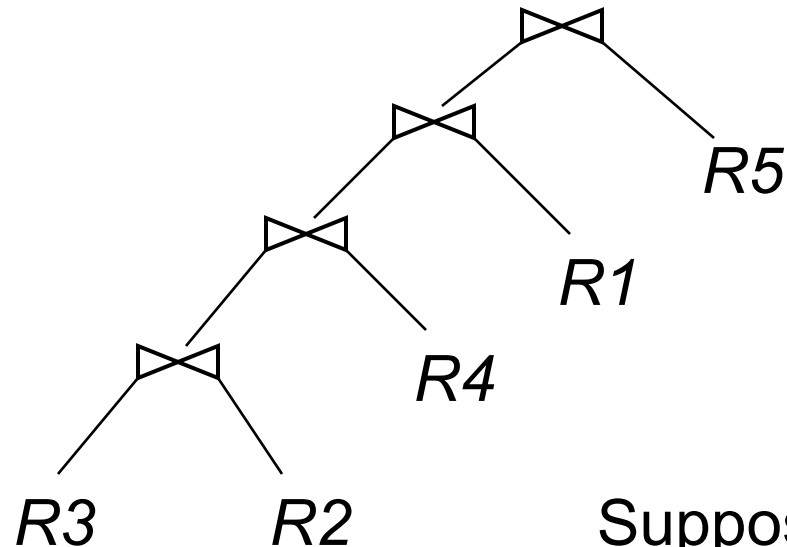


Suppose,  
this is an Optimal Plan  
for joining  $R1 \dots R5$ :

# Principle of Optimality

Query:  $R1 \bowtie R2 \bowtie R3 \bowtie R4 \bowtie R5$

---



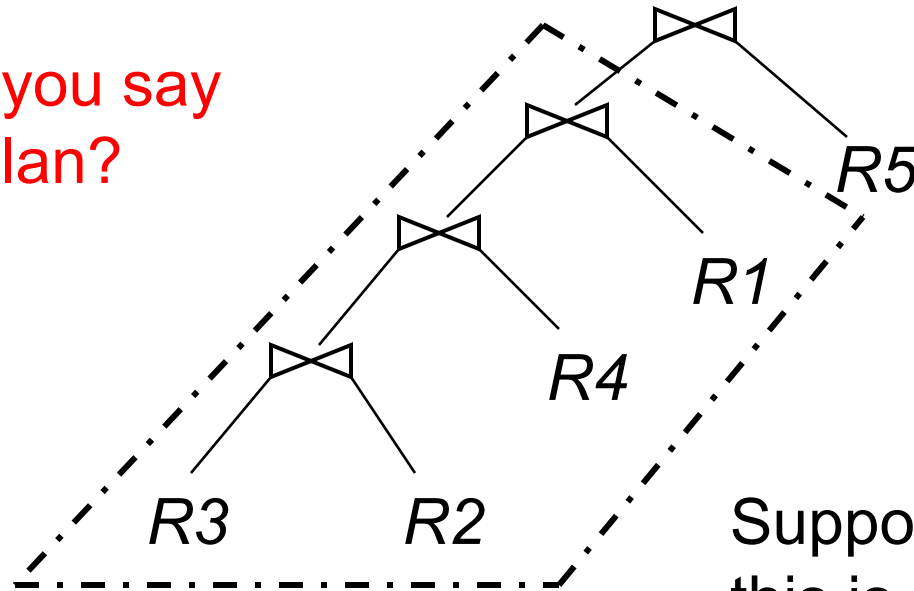
Suppose,  
this is an Optimal Plan  
for joining  $R1 \dots R5$ :

# Principle of Optimality

Query:  $R1 \bowtie R2 \bowtie R3 \bowtie R4 \bowtie R5$

---

Then, what can you say about this sub-plan?



This has to be the optimal plan for joining  $R3, R2, R4, R1$

Suppose, this is an Optimal Plan for joining  $R1 \dots R5$ :



# Principle of Optimality

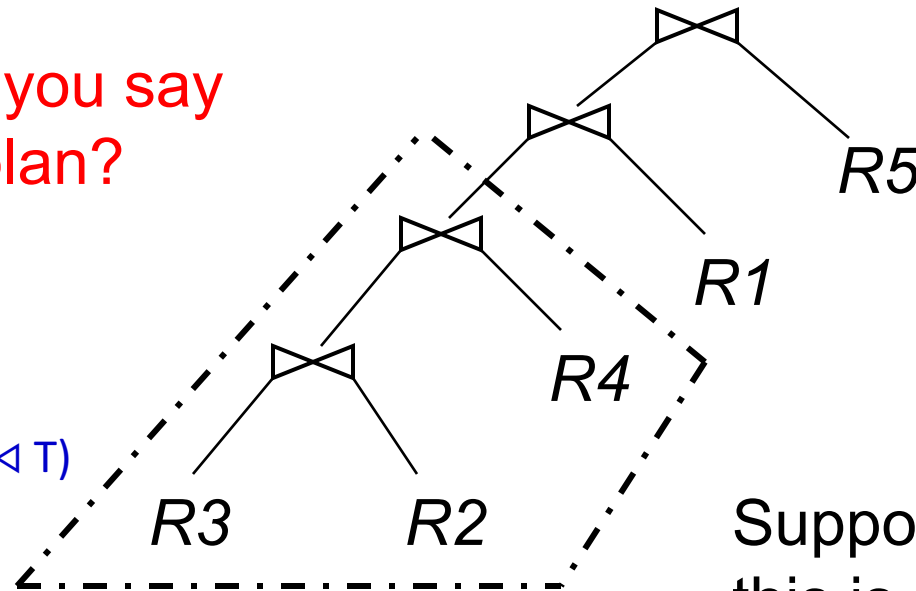
Query:  $R1 \bowtie R2 \bowtie R3 \bowtie R4 \bowtie R5$

Then, what can you say about this sub-plan?

We are using the associativity and commutativity of joins

$$(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$$

$$R \bowtie S = S \bowtie R$$



Suppose, this is an Optimal Plan for joining  $R1 \dots R5$ :

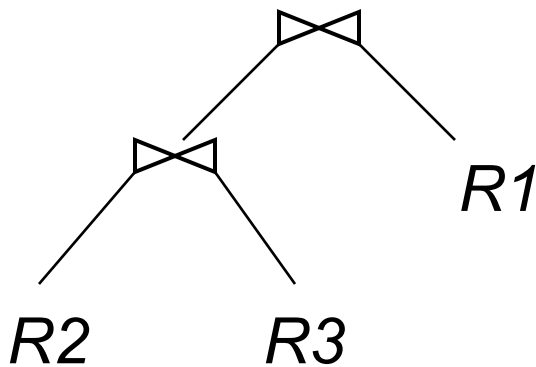
This has to be the optimal plan for joining  $R3, R2, R4$

# Exploiting Principle of Optimality

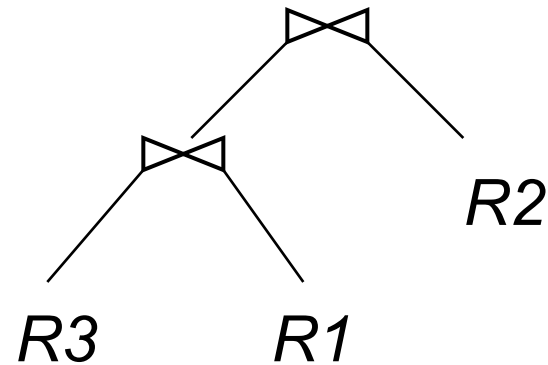
Query:  $R1 \bowtie R2 \bowtie \dots \bowtie Rn$

---

Both are giving the same result  
 $R2 \bowtie R3 \bowtie R1 = R3 \bowtie R1 \bowtie R2$



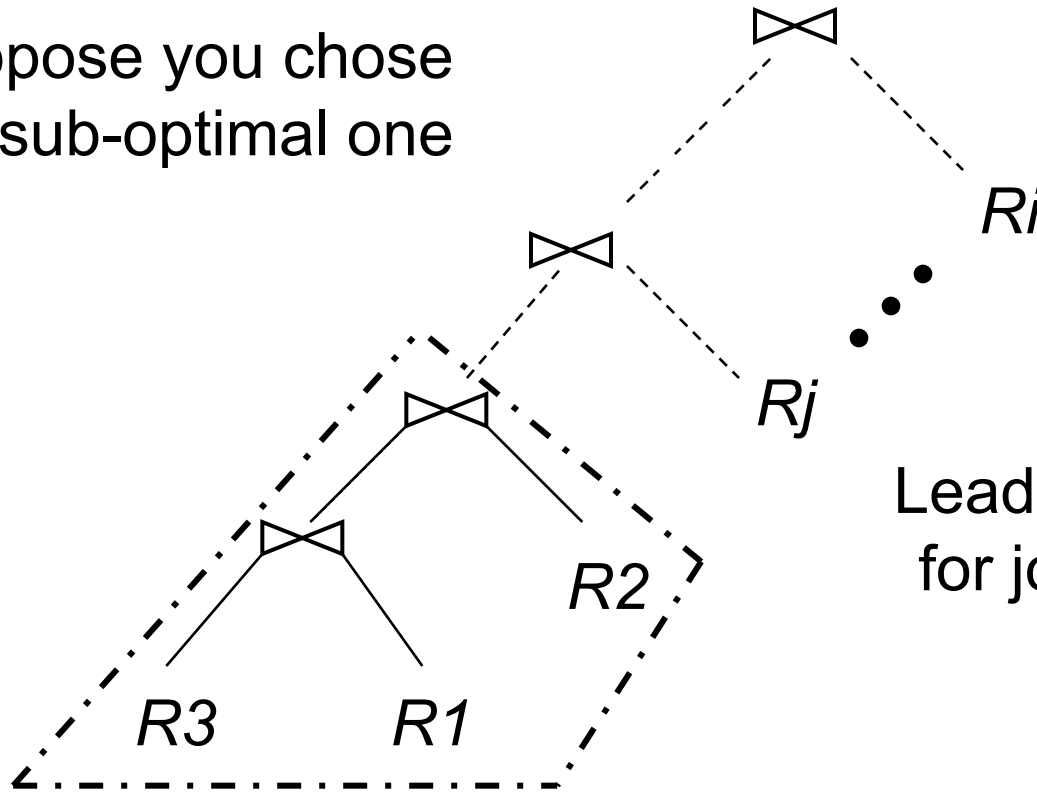
Optimal  
for joining  $R1, R2, R3$



Sub-Optimal  
for joining  $R1, R2, R3$

# Exploiting Principle of Optimality

Suppose you chose the sub-optimal one



Leads to sub-Optimal for joining  $R_1, \dots, R_n$

**A sub-optimal sub-plan cannot lead to an optimal plan**

# Notation

$\text{OPT} ( \{ R1, R2, R3 \} ):$

Cost of optimal plan to join  $R1, R2, R3$

$T ( \{ R1, R2, R3 \} ):$

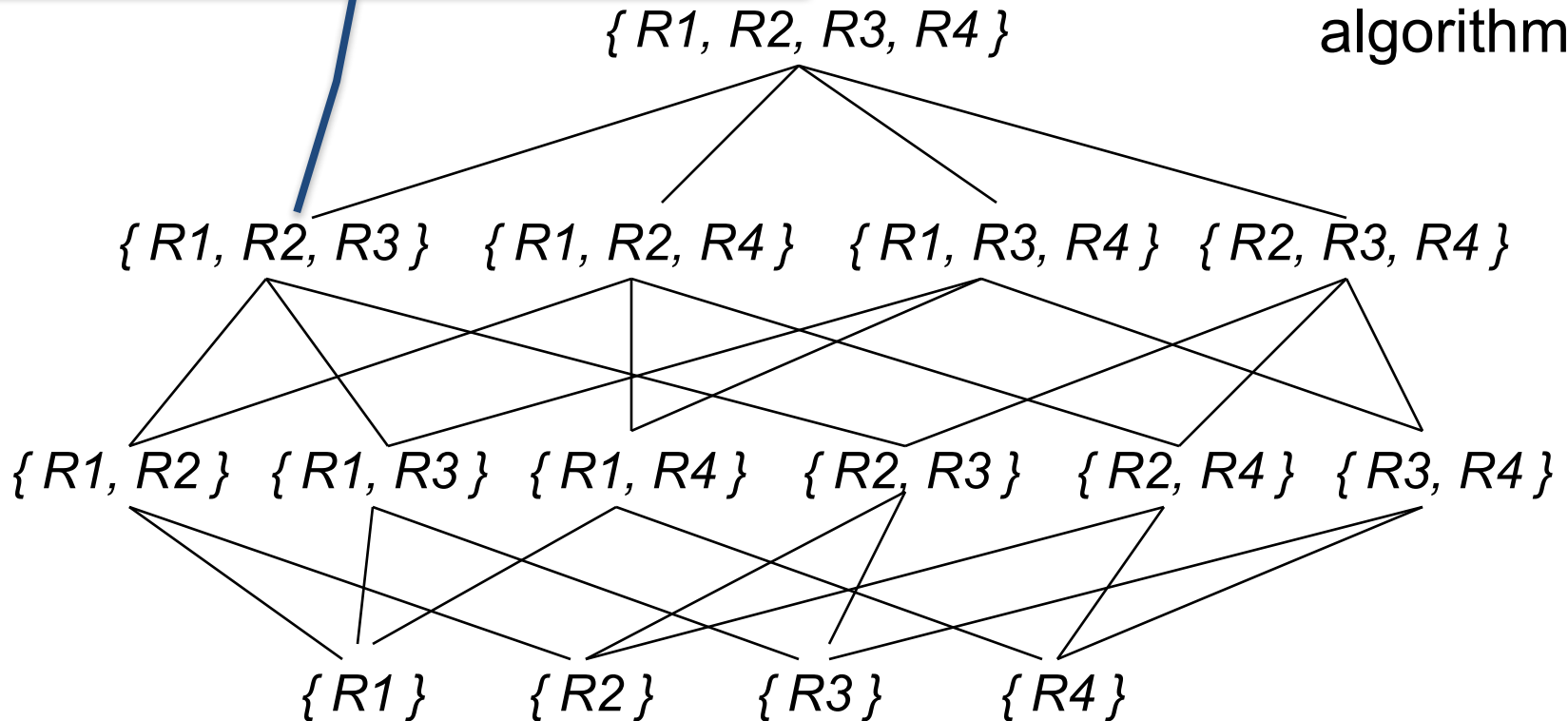
Number of tuples in  $R1 \bowtie R2 \bowtie R3$

# Selinger Algorithm:

Query:  $R1 \bowtie R2 \bowtie R3 \bowtie R4$

e.g. All possible permutations of R1, R2, R3  
have been considered  
after  $\text{OPT}(\{R1, R2, R3\})$  has been computed

Progress  
of  
algorithm



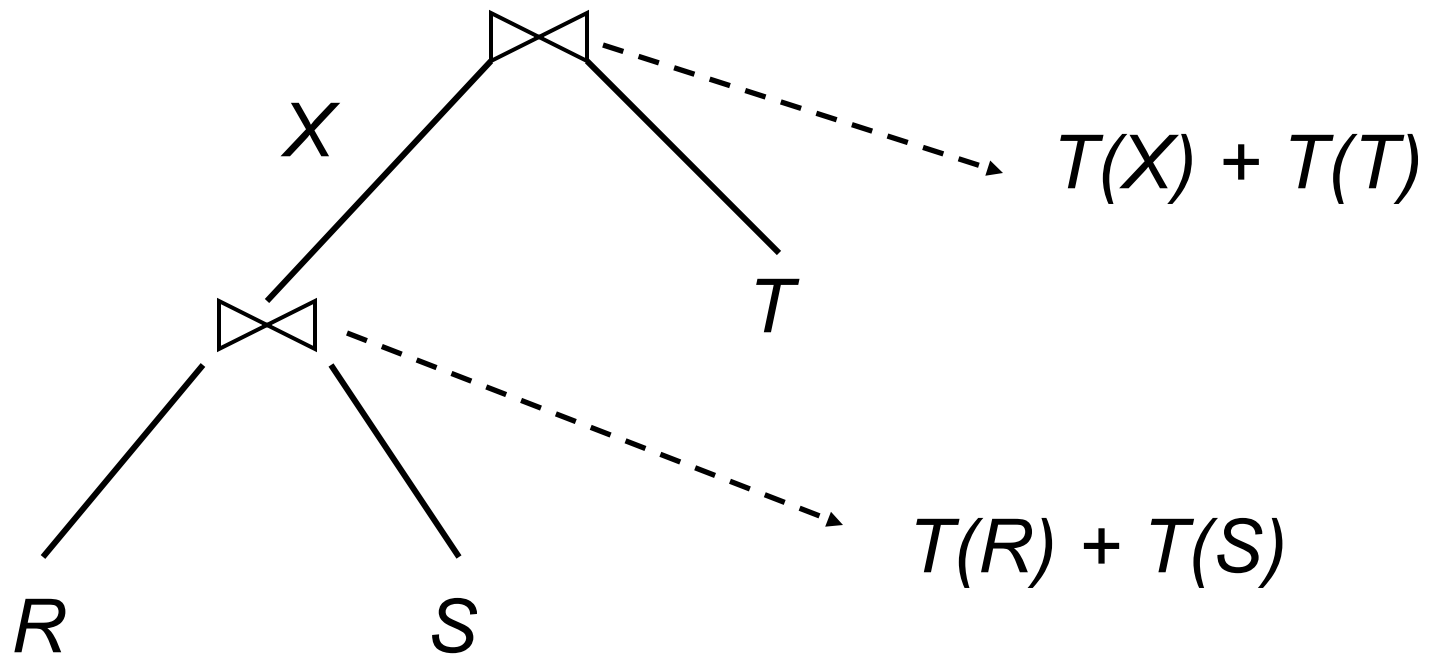
# Simple Cost Model

$$\text{Cost (R } \bowtie \text{ S)} = T(\text{R}) + T(\text{S})$$

All other operators have 0 cost

**Note: The simple cost model used for illustration only,  
it is not used in practice**

# Cost Model Example



Total Cost:  $T(R) + T(S) + T(T) + T(X)$

# Selinger Algorithm:

OPT ( { R1, R2, R3 } ):

Min

$$\text{OPT} ( \{ R1, R2 \} ) + T ( \{ R1, R2 \} ) + T(R3)$$

$$\text{OPT} ( \{ R2, R3 \} ) + T ( \{ R2, R3 \} ) + T(R1)$$

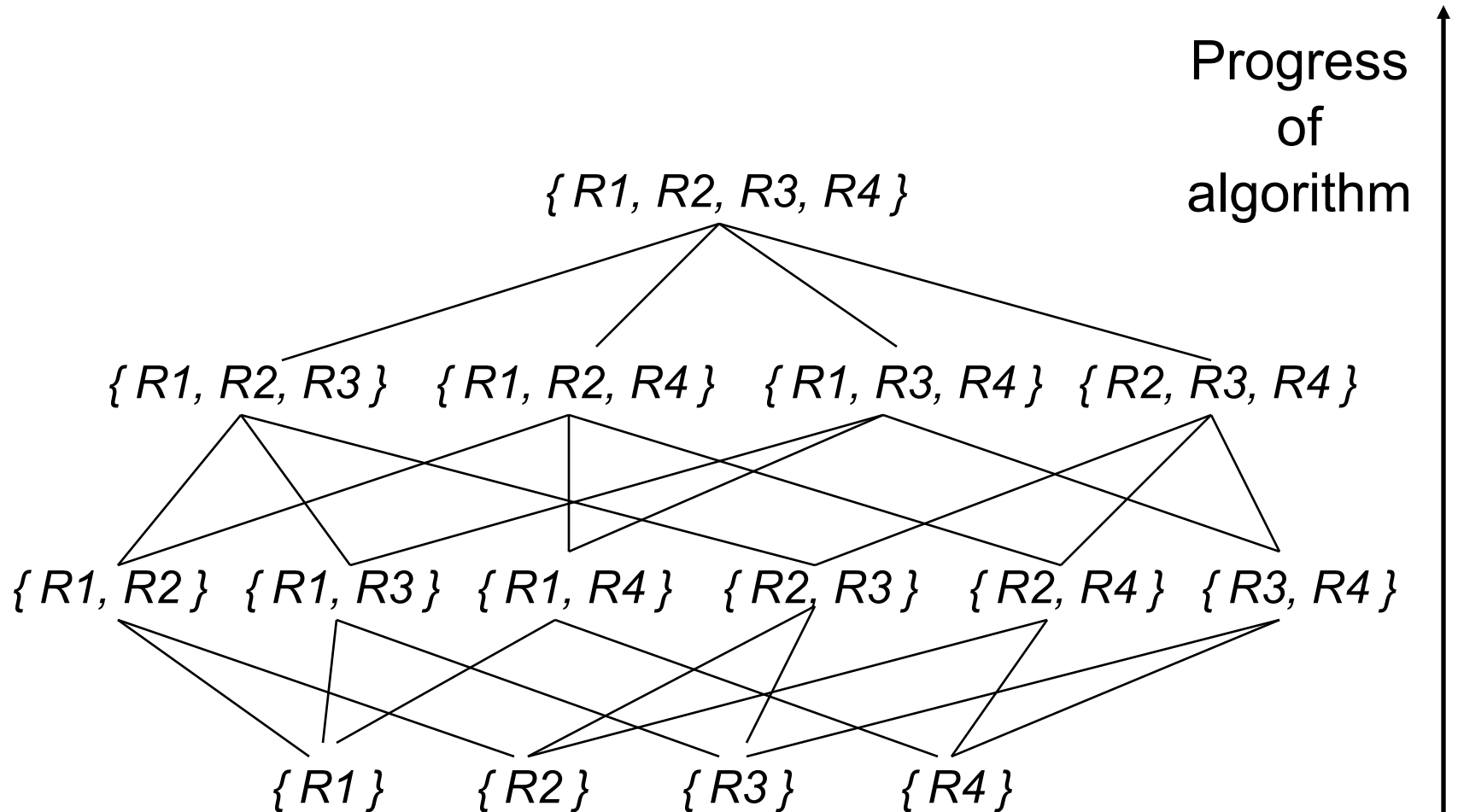
$$\text{OPT} ( \{ R1, R3 \} ) + T ( \{ R1, R3 \} ) + T(R2)$$

*Note: Valid only for the simple cost model*



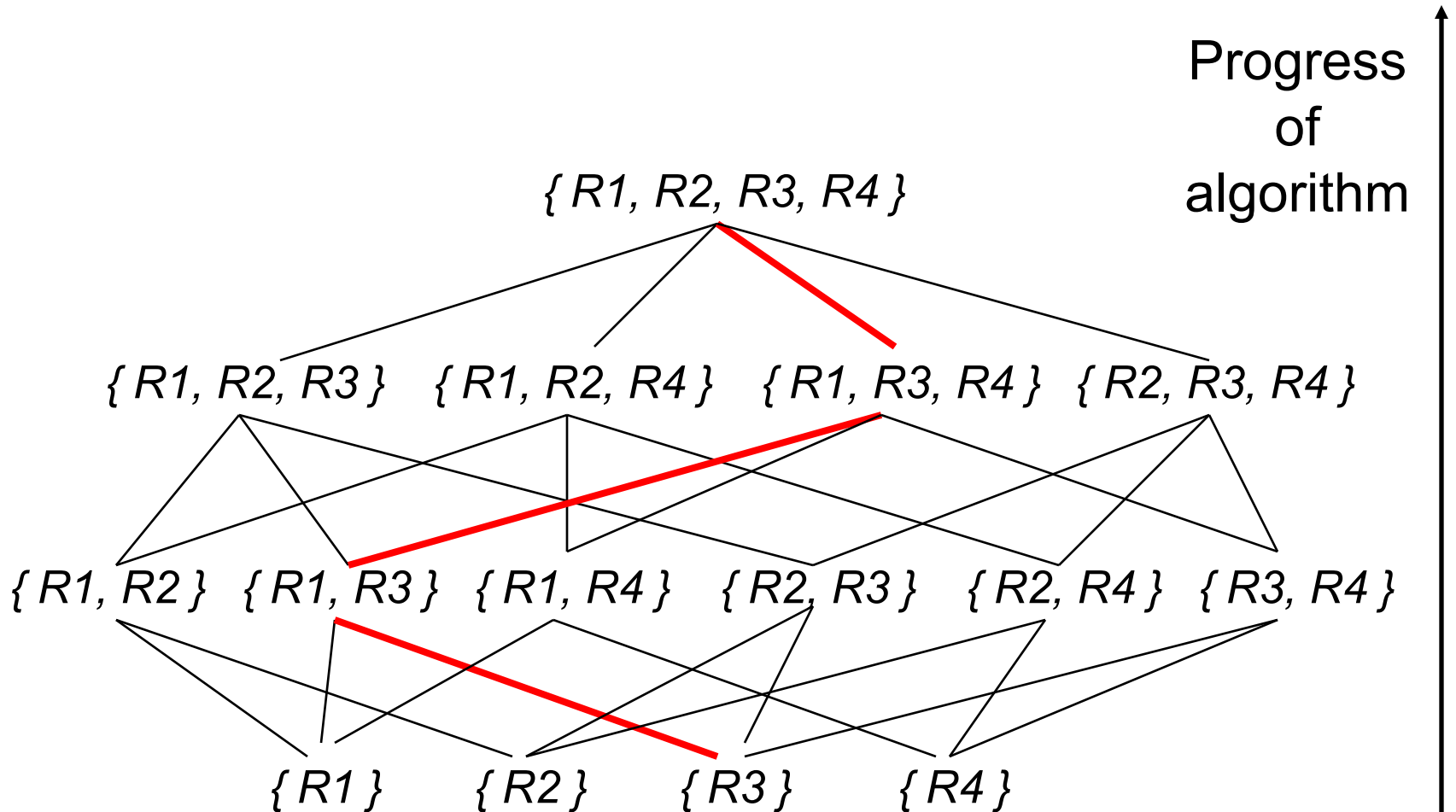
# Selinger Algorithm:

Query:  $R1 \bowtie R2 \bowtie R3 \bowtie R4$



# Selinger Algorithm:

Query:  $R1 \bowtie R2 \bowtie R3 \bowtie R4$



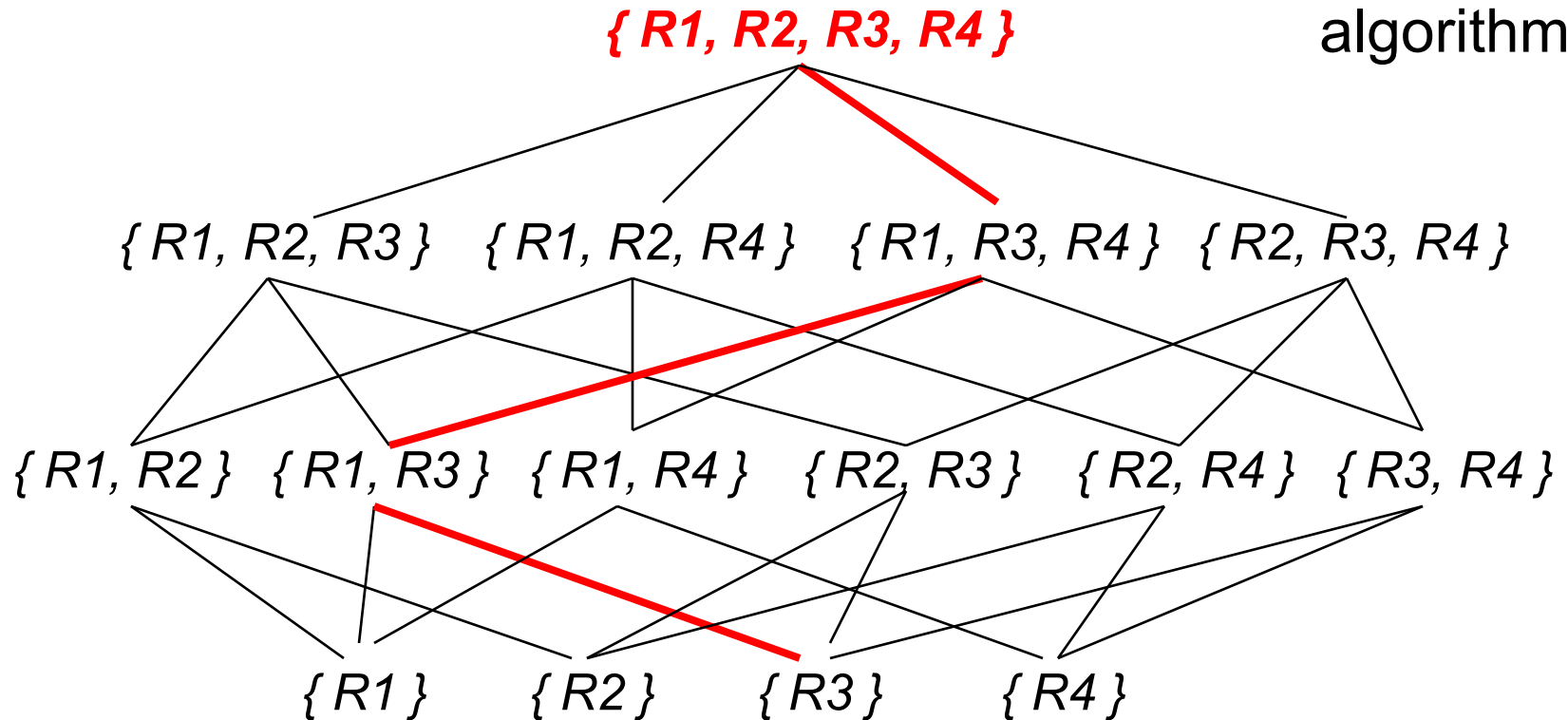
# Selinger Algorithm:

Query:  $R1 \bowtie R2 \bowtie R3 \bowtie R4$

Q. How to optimally compute join of  $\{R1, R2, R3, R4\}$ ?

Ans: First optimally join  $\{R1, R3, R4\}$  then join with  $R2$  as inner.

Progress  
of  
algorithm



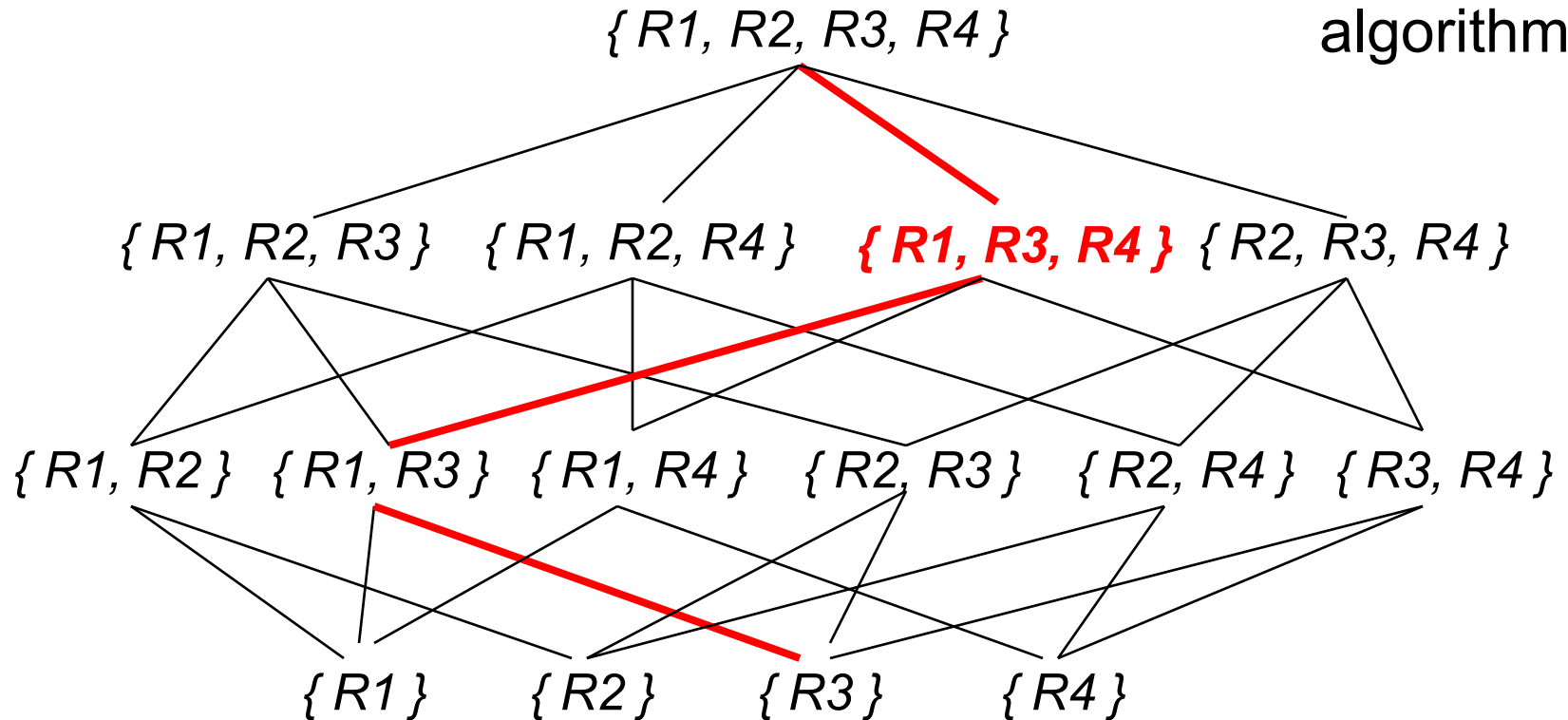
# Selinger Algorithm:

Query:  $R1 \bowtie R2 \bowtie R3 \bowtie R4$

Q. How to optimally compute join of  $\{R1, R3, R4\}$ ?

Ans: First optimally join  $\{R1, R3\}$ , then join with  $R4$  as inner.

Progress  
of  
algorithm



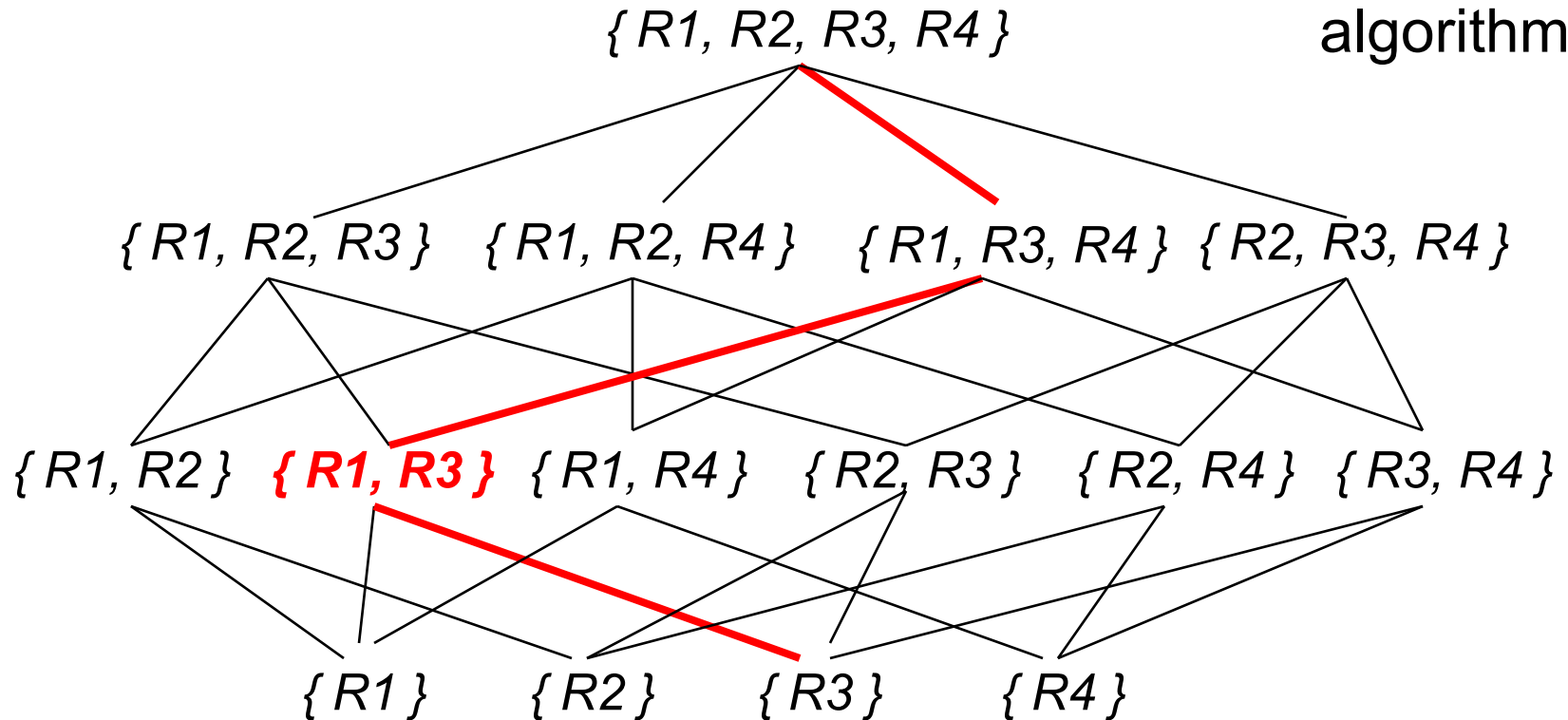
# Selinger Algorithm:

Query:  $R1 \bowtie R2 \bowtie R3 \bowtie R4$

Q. How to optimally compute join of  $\{R1, R3\}$ ?

Ans: First optimally join  $\{R3\}$ , then join with  $R1$  as inner.

Progress  
of  
algorithm



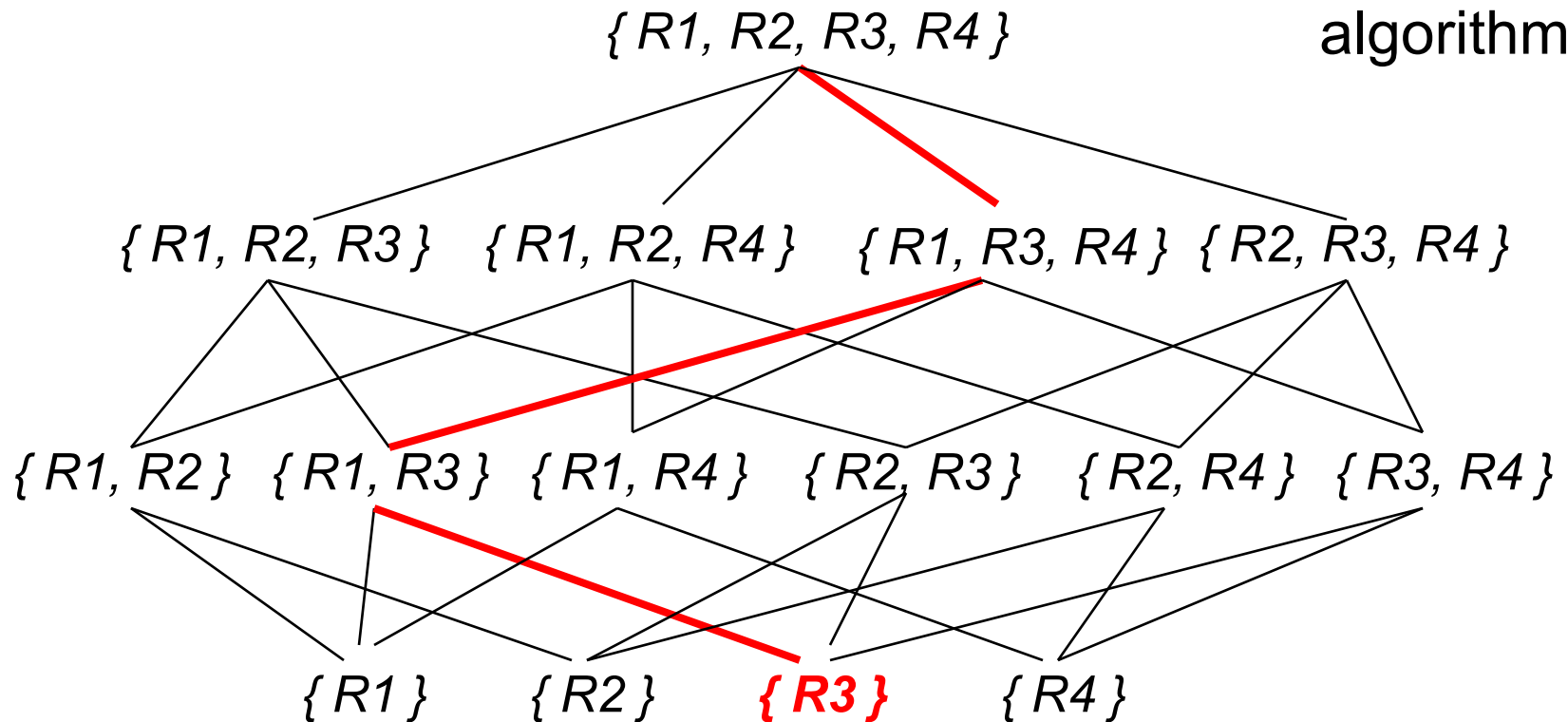
# Selinger Algorithm:

Query:  $R1 \bowtie R2 \bowtie R3 \bowtie R4$

Q. How to optimally compute join of {R3}?

Ans: Single relation – so **optimally scan R3.**

Progress  
of  
algorithm

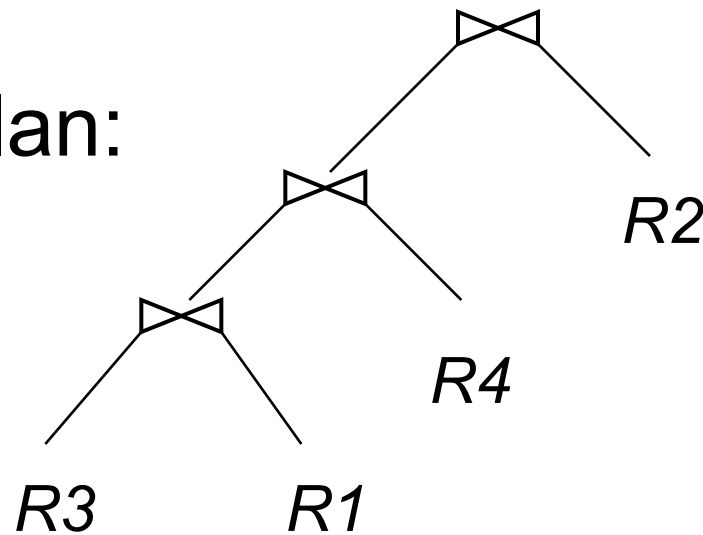


# Selinger Algorithm:

Query:  $R1 \bowtie R2 \bowtie R3 \bowtie R4$

---

Final optimal plan:



**NOTE :** There is a one-one correspondence between the permutation (R3, R1, R4, R2) and the above left deep plan

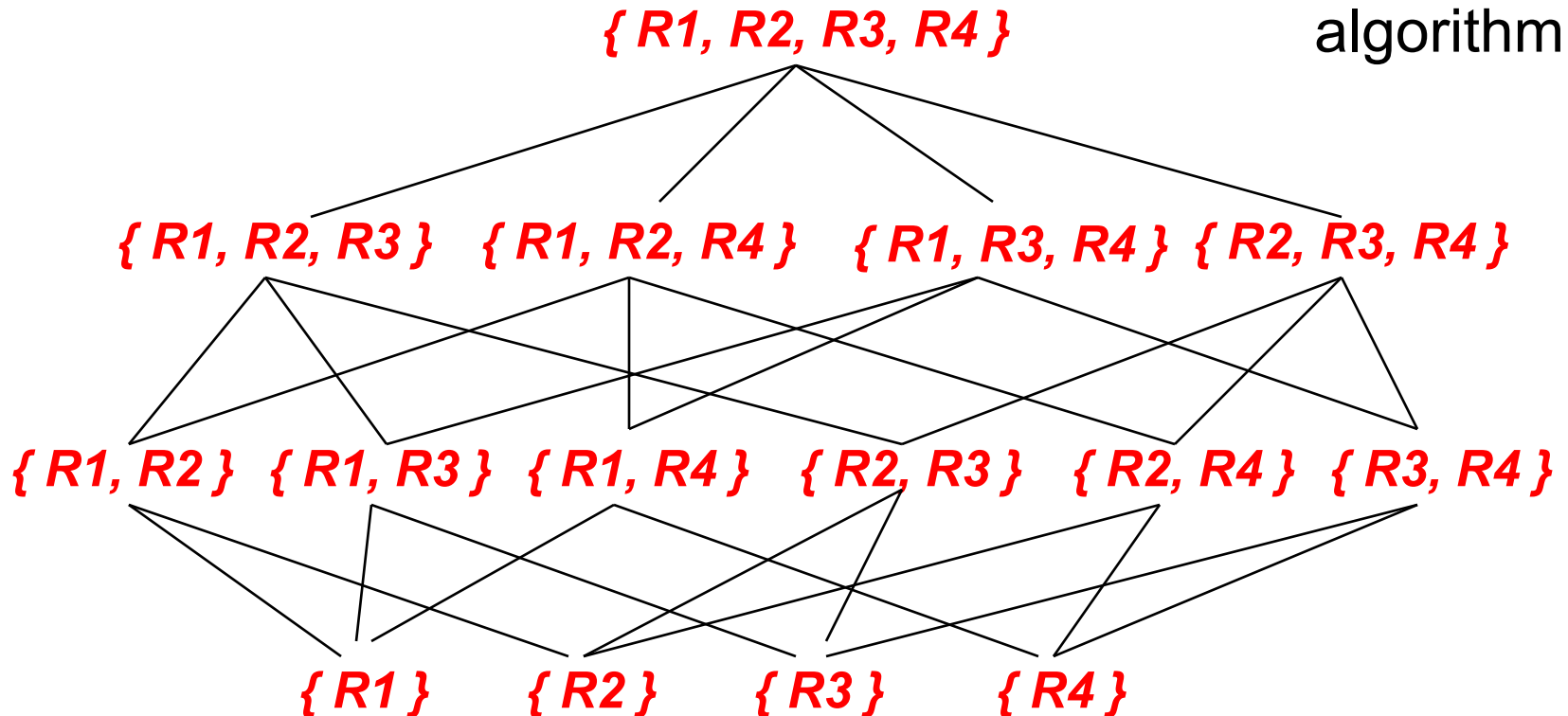
# Selinger Algorithm:

Query:  $R1 \bowtie R2 \bowtie R3 \bowtie R4$

NOTE: (\*VERY IMPORTANT\*)

- This is \*NOT\* done by top-down recursive calls.
- This is done BOTTOM-UP computing the optimal cost of \*all\* nodes in this lattice only once (dynamic programming).

Progress  
of  
algorithm





# More on Query Optimizations

- See the survey (on course website):  
“An Overview of Query Optimization in Relational Systems” by Surajit Chaudhuri
- Covers other aspects like
  - Pushing group by before joins
  - Merging views and nested queries
  - “Semi-join”-like techniques for multi-block queries
    - covered later in distributed databases
  - Statistics and optimizations
  - Starbust and Volcano/Cascade architecture, etc
- We may revisit Query Optimization again in our “Advanced Topics” later in the course.