CS848 Fall 2025: Algorithmic Aspects of Query Processing

Output-Optimal Algorithms for Join-Aggregate Queries

Xiao Hu

Sep 22, 2025

Agenda

- Last class: Worst-case optimal join algorithms
- This class: join-aggregate queries
  - Matrix multiplication and its Variant
  - Limitations of Yannakakis algorithm
  - Output-optimal algorithm for Chain Matrix Multiplication
  - General join-aggregate queries
  - General Algorithm

Related Pointers

- X. HU, "Output-optimal Algorithms for Join-Aggregate Queries," PODS 2025.
- S. ABITEBOUL, R. HULL and V. VIANU, "Foundations of Databases."
- M. YANNAKAKIS, "Algorithms for acyclic database schemes," VLDB 1981.
- G. GOTTLOB, N. LEONE and F. SCARCELLO, "Hypertree Decompositions and Tractable Queries," Journal of Computer and System Sciences 64 (2002) .
- M. GROHE, T. SCHWENTICK and L. SEGOUFIN, "When is the evaluation of conjunctive queries tractable ?," STOC 2001 .
- G. GOTTLOB, G. GRECO and F. SCARCELLO, "Treewidth and Hypertree Width".

Matrix Multiplication $= \pi_{A,C}\, R_1(A,B) \bowtie R_2(B,C)$

$$\underset{A}{\overset{B}{\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}} \cdot \underset{B}{\overset{C}{\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}} = \underset{A}{\overset{C}{\begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}}}$$

$R_1(A,B)$

A	B
1	4
2	1
2	3
3	1
4	2

$\bowtie$

$R_2(B,C)$

B	C
1	1
1	2
2	2
2	4
3	2
4	3

$R_1 \bowtie R_2$

A	B	C
1	4	3
2	1	1
2	1	2
2	3	2
3	1	1
3	1	2
4	2	2
4	2	4

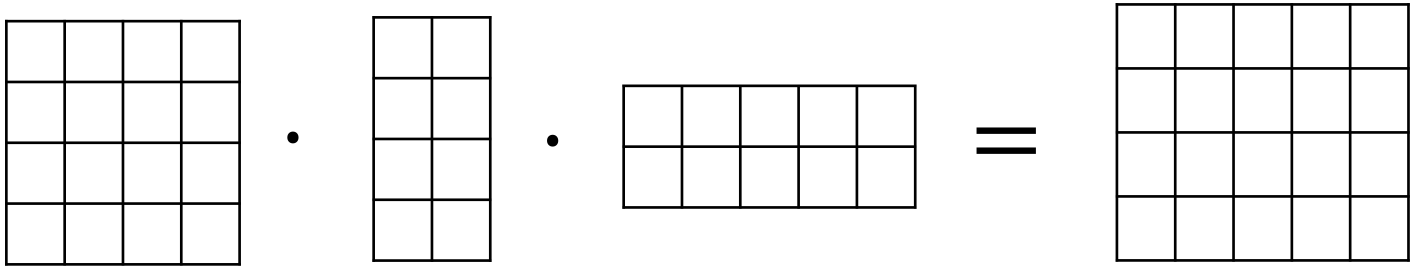
$\pi_{A,C}$ →

$R_3(A,C)$

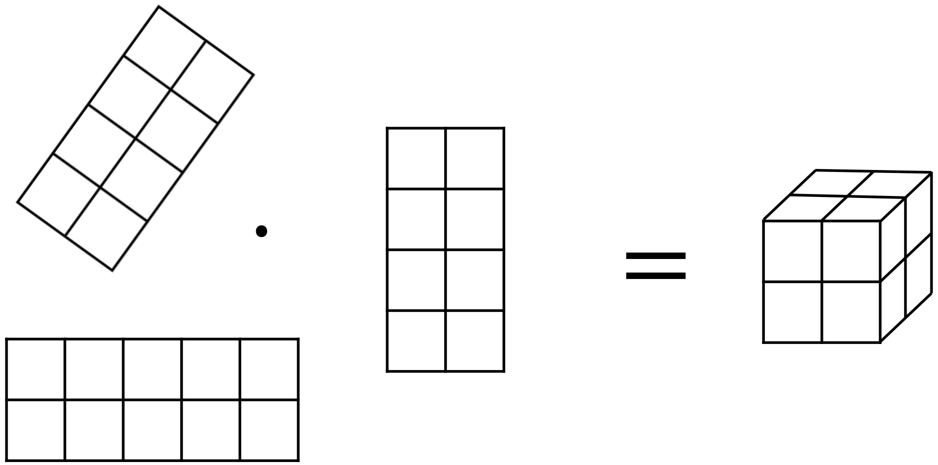
A	C
1	3
2	1
2	2
3	1
3	2
4	2
4	4

Variants of Matrix Multiplication

- Chain Matrix Multiplication: $\pi_{A_1, A_{k+1}} R_1(A_1, A_2) \bowtie R_2(A_2, A_3) \bowtie \cdots \bowtie R_k(A_k, A_{k+1})$

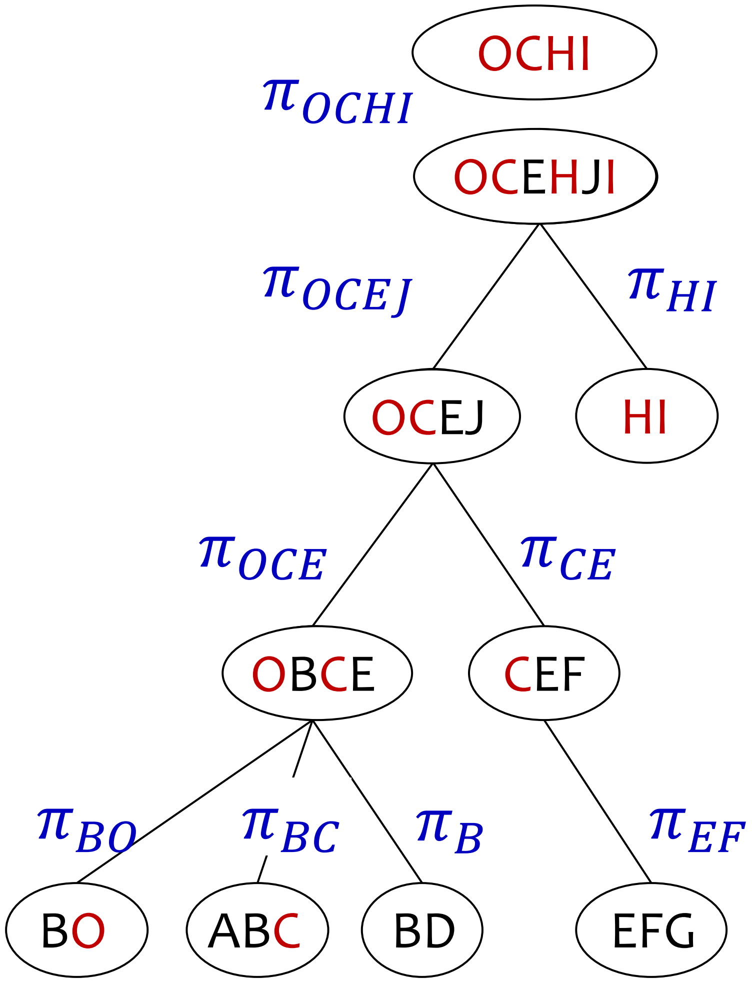


- Star Matrix Multiplication: $\pi_{A_1, A_2, \cdots, A_k} R_1(A_1, B) \bowtie R_2(A_2, B) \bowtie \cdots \bowtie R_k(A_k, B)$



Yannakakis for Acyclic Join-Project/Aggregate Queries

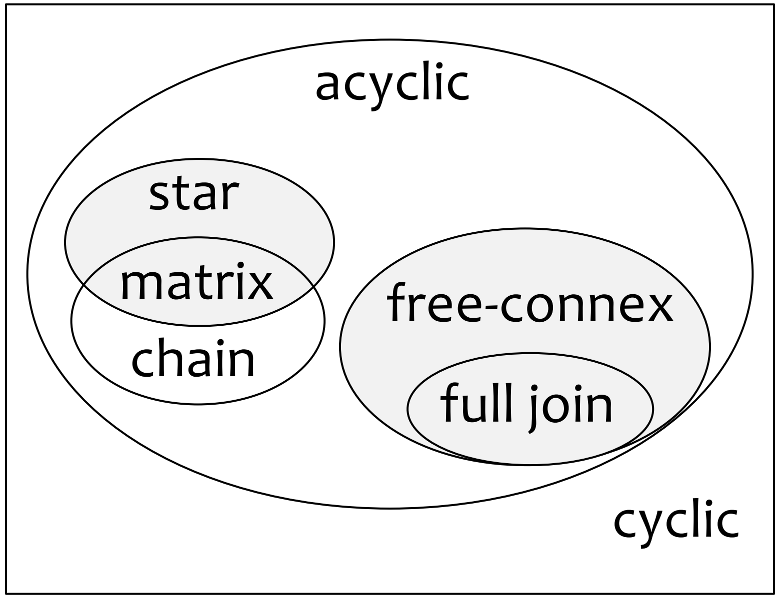
- Let $\boldsymbol{y}$ be the set of output attributes
- Semi-join Reducer
  - If $\boldsymbol{y} = \emptyset$, output true if and only if $R_r \neq \emptyset$
- In a bottom-up phase:
  - Project as early as possible!
  - For a non-output attribute, if it does not appear in any node above, then project it away



output attributes

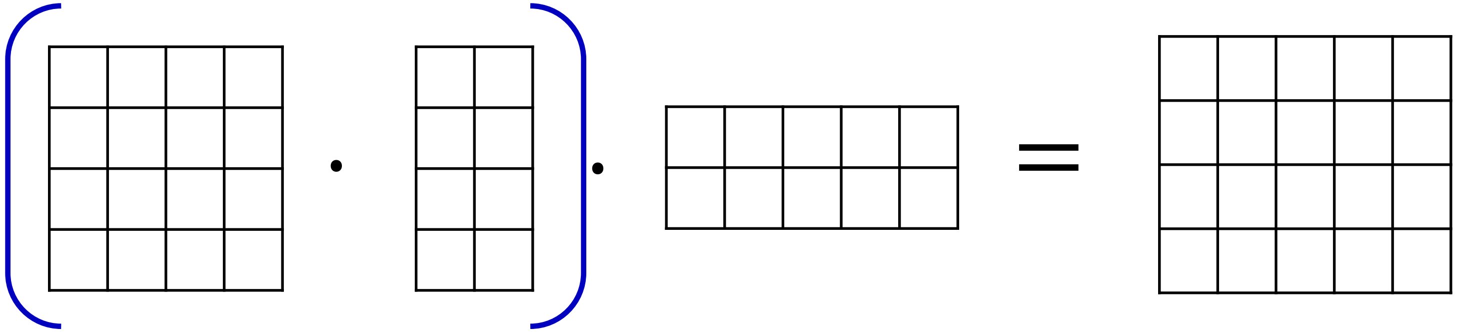
Yannakakis Algorithm Revisited

- Yannakakis algorithm [Y81] for acyclic queries [BFMY83]
  - Acyclic join queries: $\Theta(N + OUT)$ [Lecture 2-3]
  - Free-connex queries: $\Theta(N + OUT)$ [Lecture 3]
  - Acyclic but non-free-connex queries:
    - Matrix multiplication: $\Theta\left(N \cdot \sqrt{OUT}\right)$ [AP09]
      - Output-optimal
    - Star Matrix Multiplication : $\Theta\left(N \cdot OUT^{1-\frac{1}{k}}\right)$ [AP09]
      - Output-optimal
    - Chain Matrix Multiplication and all other queries: $O(N \cdot OUT)$ [Lecture 3]



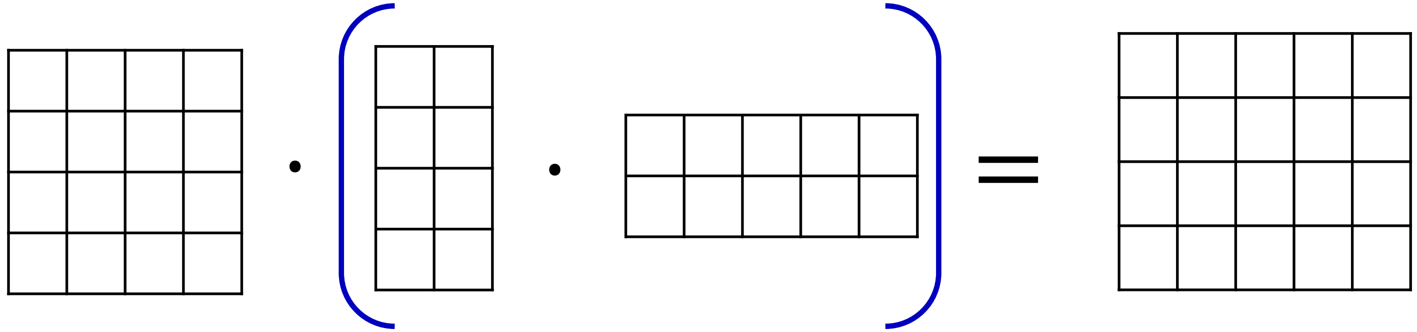
The limitation of Yannakakis Algorithm

Consider $\pi_{A_1,A_4} R_1(A_1, A_2) \bowtie R_2(A_2, A_3) \bowtie R_3(A_3, A_4)$



Query plan 1:

$\pi_{A_1,A_4}\left(\pi_{A_1,A_3} R_1 \bowtie R_2\right) \bowtie R_3$



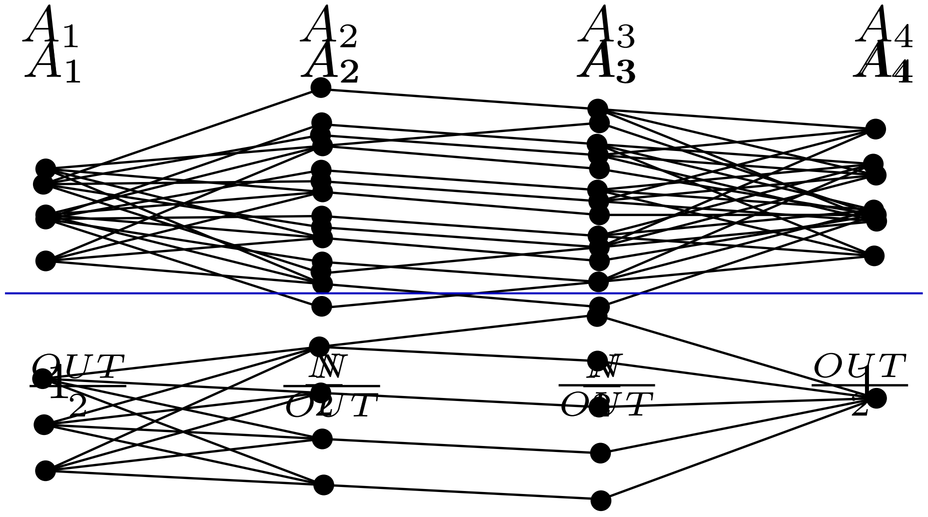
Query plan 2:

$\pi_{A_1,A_4} R_1 \bowtie \left(\pi_{A_2,A_4} R_2 \bowtie R_3\right)$

The limitation of Yannakakis Algorithm

Consider $\pi_{A_1,A_4} R_1(A_1, A_2) \bowtie R_2(A_2, A_3) \bowtie R_3(A_3, A_4)$

- Plan 1 $= \Theta(N \cdot OUT)$ but Plan 2 $= O(N)$
- Plan 2 $= \Theta(N \cdot OUT)$ but Plan 1 $= O(N)$
- Both plans $= \Theta(N \cdot OUT)$



Hybrid Yannakakis Algorithm

Consider $\pi_{A_1,A_4} R_1(A_1, A_2) \bowtie R_2(A_2, A_3) \bowtie R_3(A_3, A_4)$

Assume the value of OUT is known

- A value $b \in A_2$ is heavy if it appears in more than $\sqrt{OUT}$ tuples in R_1 and light otherwise.
- $\pi_{A_1,A_4} R_1\left(A_1, A_2^{\text{heavy}}\right) \bowtie R_2\left(A_2^{\text{heavy}}, A_3\right) \bowtie R_3(A_3, A_4)$
  - There are at most $\sqrt{OUT}$ distinct values in A_4

Query plan 2:
$\pi_{A_1,\, A_4} R_1 \bowtie \left(\pi_{A_2,A_4} R_2 \bowtie R_3\right)$

- $\pi_{A_1,A_4} R_1\left(A_1, A_2^{\text{light}}\right) \bowtie R_2\left(A_2^{\text{light}}, A_3\right) \bowtie R_3(A_3, A_4)$

Hybrid Yannakakis Algorithm

Consider $\pi_{A_1,A_4} R_1\left(A_1, A_2^{\text{light}}\right) \bowtie R_2\left(A_2^{\text{light}}, A_3\right) \bowtie R_3(A_3, A_4)$

Compute $S_1(A_1, A_3) = \pi_{A_1,A_3} R_1\left(A_1, A_2^{\text{light}}\right) \bowtie R_2\left(A_2^{\text{light}}, A_3\right)$

- A value $c \in A_3$ is heavy if it appears in more than $\sqrt{OUT}$ tuples in S_1 and light otherwise.
- $\pi_{A_1,A_4} R_1\left(A_1, A_2^{\text{light}}\right) \bowtie R_2\left(A_2^{\text{light}}, A_3^{\text{heavy}}\right) \bowtie R_3\left(A_3^{\text{heavy}}, A_4\right)$
  - There are at most $\sqrt{OUT}$ distinct values in A_4
- $\pi_{A_1,A_4} R_1\left(A_1, A_2^{\text{light}}\right) \bowtie R_2\left(A_2^{\text{light}}, A_3^{\text{light}}\right) \bowtie R_3\left(A_3^{\text{light}}, A_4\right)$
  - Each tuple in R_2 and R_3 can be joined with at most $\sqrt{OUT}$ distinct values in A_1

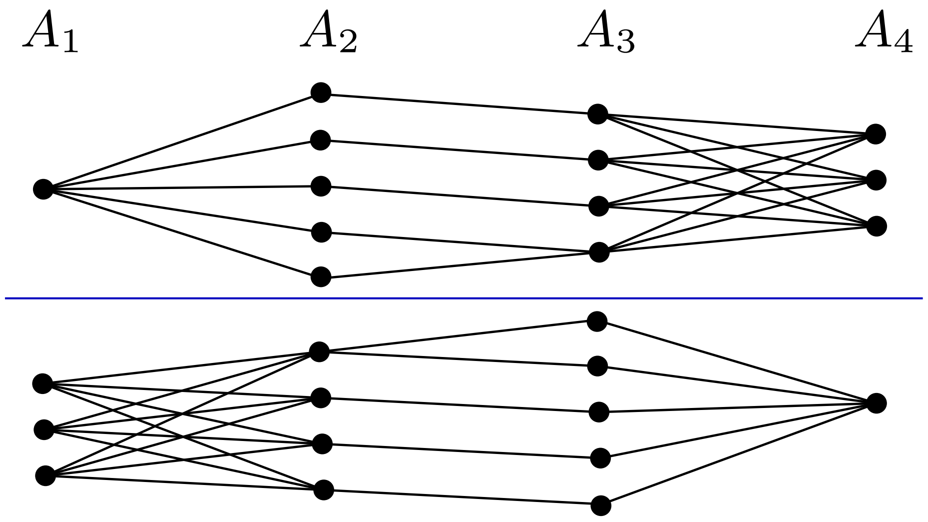
Query plan 2: $\pi_{A_1, A_4} R_1 \bowtie \left(\pi_{A_2,A_4} R_2 \bowtie R_3\right)$

Query plan 1: $\pi_{A_1,A_4} \left(\pi_{A_1,A_3} R_1 \bowtie R_2\right) \bowtie R_3$

Hybrid Yannakakis Algorithm for Chain MM

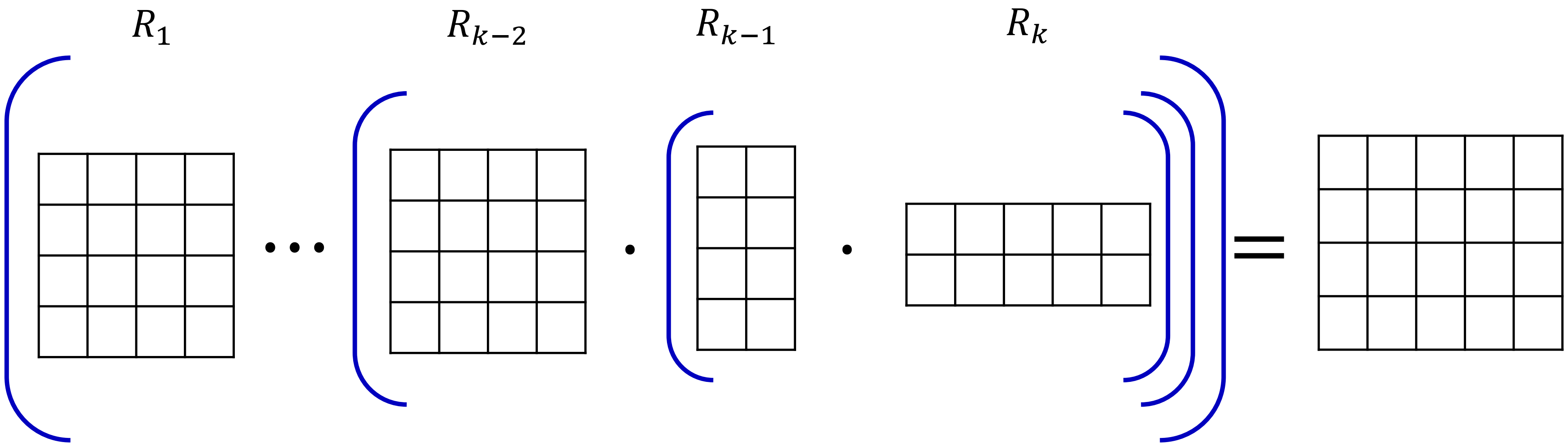
Compute $\pi_{A_1,A_{k+1}} R_1(A_1, A_2) \bowtie R_2(A_2, A_3) \bowtie \cdots \bowtie R_k(A_k, A_{k+1})$

- Challenges
  - What to partition?
  - How to partition?
  - Which plan to pick?



Candidate Plans for Chain MM

Compute $\pi_{A_1,A_{k+1}} R_1(A_1, A_2) \bowtie R_2(A_2, A_3) \bowtie \cdots \bowtie R_k(A_k, A_{k+1})$



$V_{k-1}(A_{k-1}, A_k, A_{k+1}) = R_{k-1} \bowtie R_k$

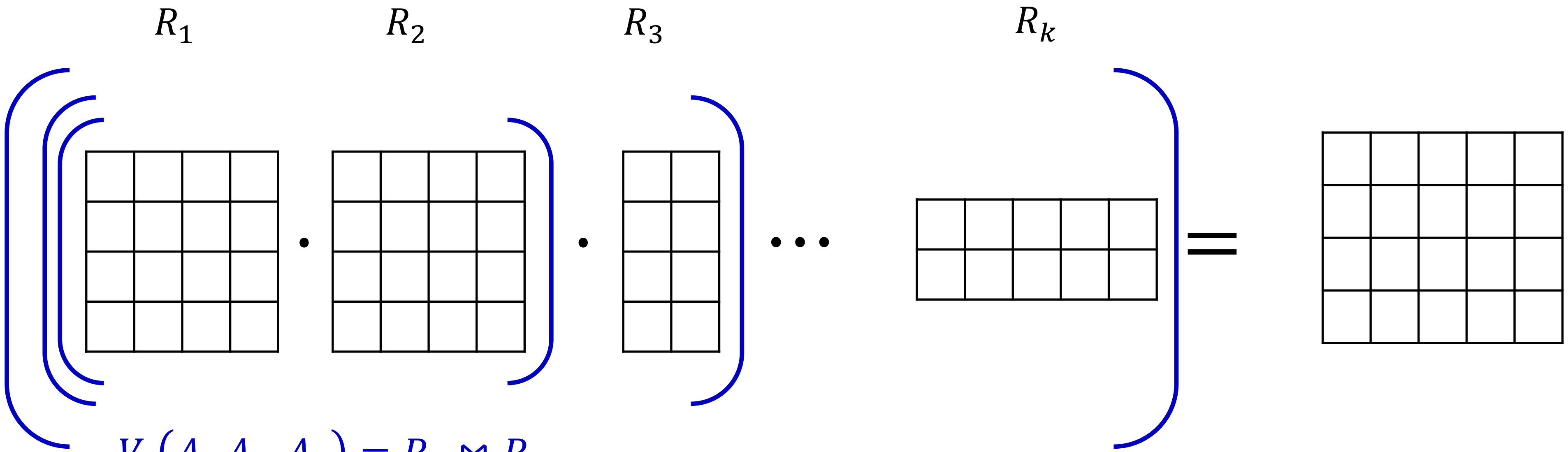
$V_{k-2}(A_{k-2}, A_{k-1}, A_{k+1}) = R_{k-2} \bowtie (\pi_{A_{k-1},A_{k+1}} V_{k-1})$

$\cdots$

$\pi_{A_1,A_{k+1}} V_1$ $\quad V_1(A_1, A_2, A_{k+1}) = R_1 \bowtie (\pi_{A_2,A_{k+1}} V_2)$

Candidate Plans for Chain MM

Compute $\pi_{A_1,A_{k+1}} R_1(A_1, A_2) \bowtie R_2(A_2, A_3) \bowtie \cdots \bowtie R_k(A_k, A_{k+1})$



$V_1(A_1, A_2, A_3) = R_1 \bowtie R_2$

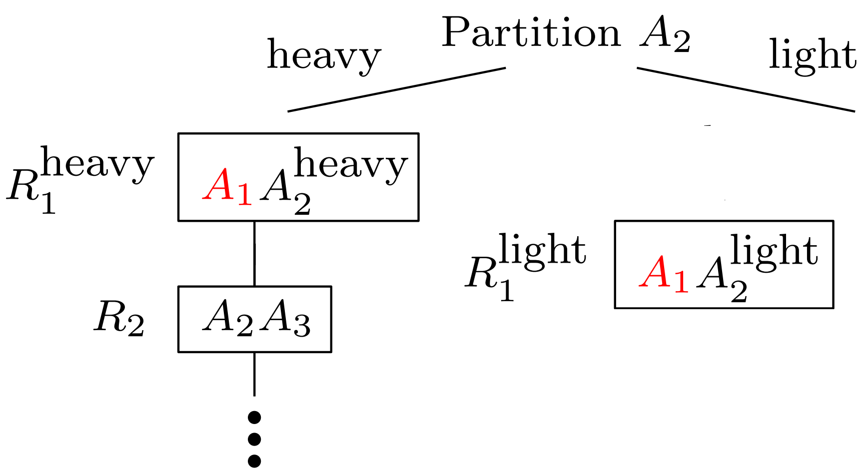
$V_2(A_1, A_3, A_4) = \left(\pi_{A_1,A_3} V_1\right) \bowtie R_3$

$\ddots$

$\pi_{A_1,A_{k+1}} V_{k-1}$ $\quad V_{k-1}(A_1, A_k, A_{k+1}) = \left(\pi_{A_1,A_k} V_{k-2}\right) \bowtie R_k$

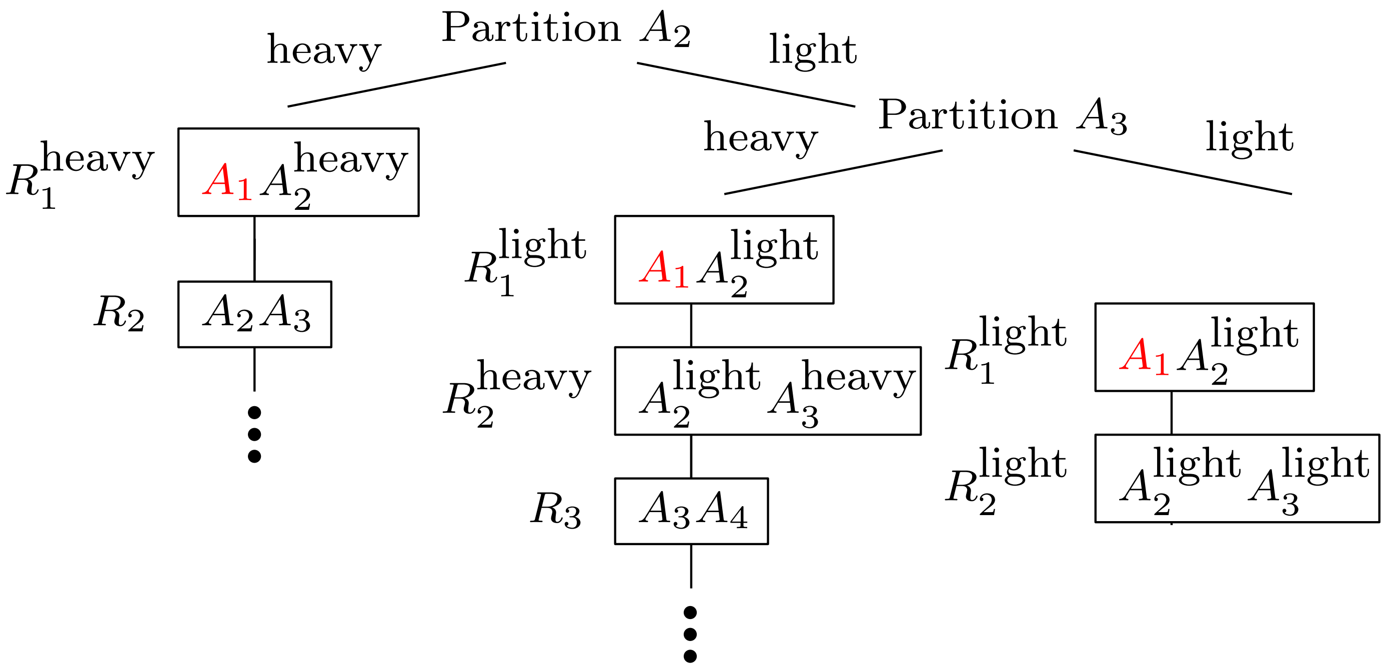
How and What to Partition?

$a \in A_2$ is heavy if it appears in $\geq \tau$ tuples in R_1



$a \in A_2$ is heavy if it appears in $\geq \tau$ tuples in R_1

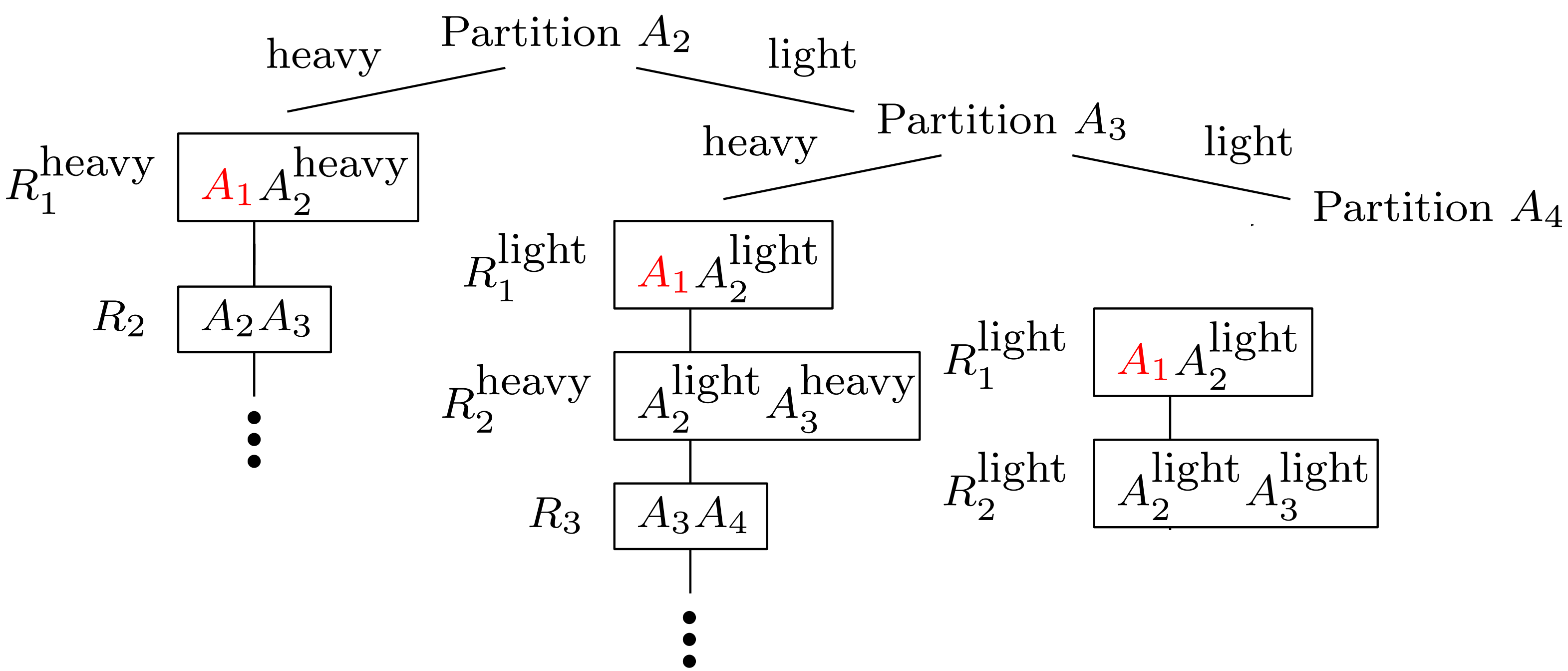
$a \in A_3$ is heavy if it appears in $\geq \tau$ tuples in $\pi_{A_1,A_3} R_1^{\text{light}} \bowtie R_2$



$a \in A_2$ is heavy if it appears in $\geq \tau$ tuples in R_1

$a \in A_3$ is heavy if it appears in $\geq \tau$ tuples in $\pi_{A_1,A_3} R_1^{\text{light}} \bowtie R_2$

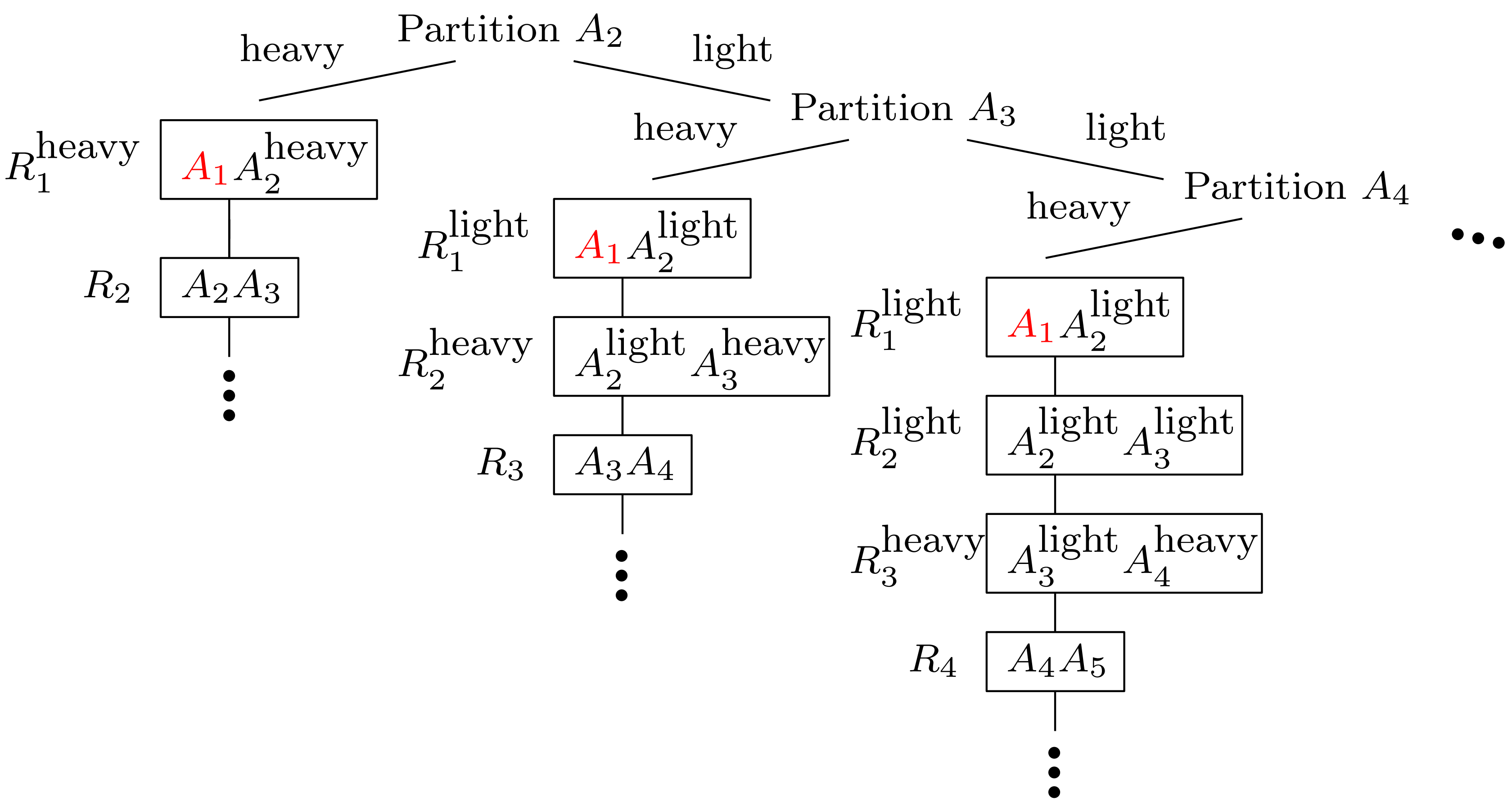
$a \in A_4$ is heavy if it appears in $\geq \tau$ tuples in $\pi_{A_1,A_4} R_1^{\text{light}} \bowtie R_2^{\text{light}} \bowtie R_3$



$a \in A_2$ is heavy if it appears in $\geq \tau$ tuples in R_1

$a \in A_3$ is heavy if it appears in $\geq \tau$ tuples in $\pi_{A_1,A_3} R_1^{\text{light}} \bowtie R_2$

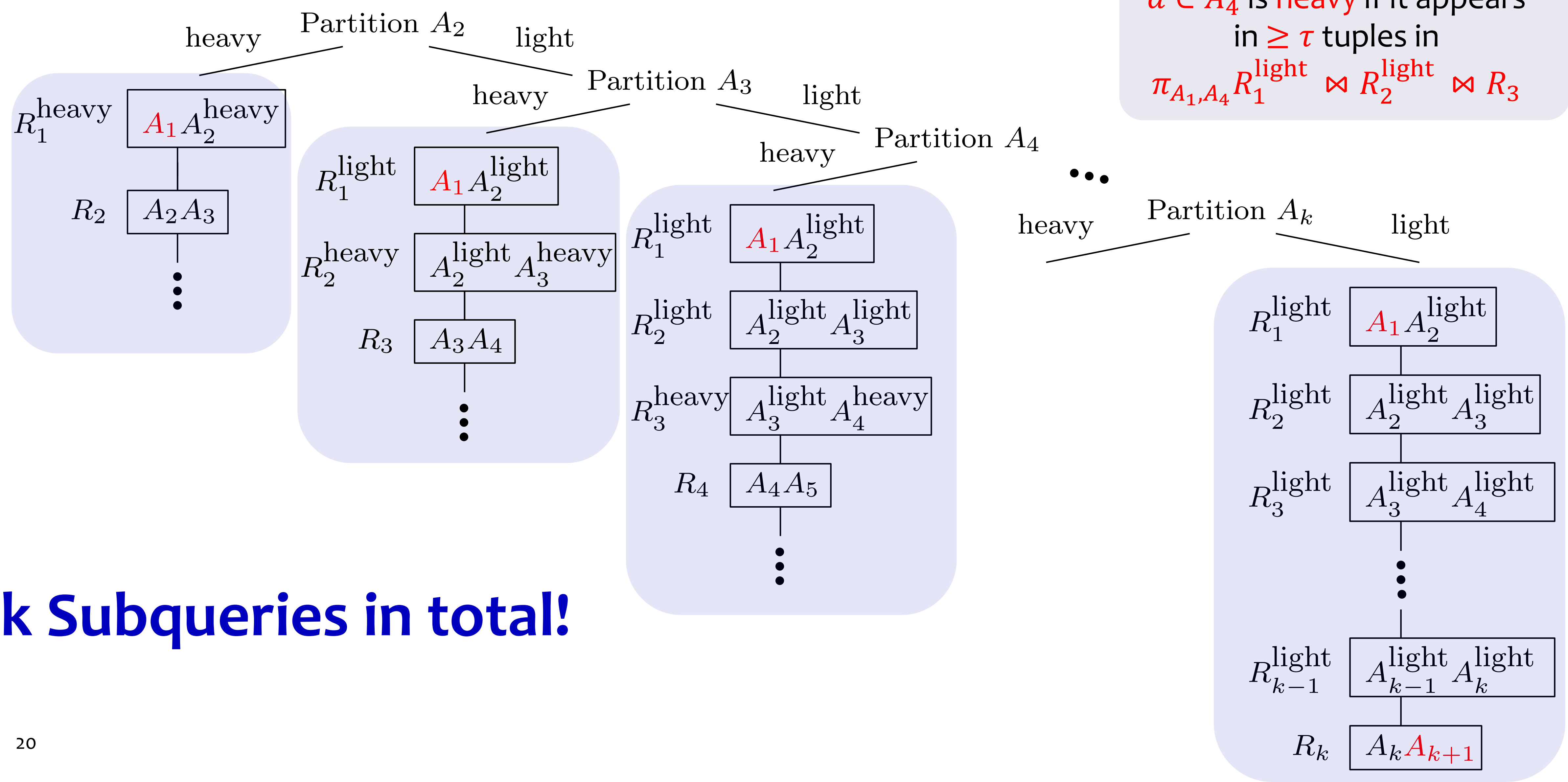
$a \in A_4$ is heavy if it appears in $\geq \tau$ tuples in $\pi_{A_1,A_4} R_1^{\text{light}} \bowtie R_2^{\text{light}} \bowtie R_3$



$a \in A_2$ is heavy if it appears in $\geq \tau$ tuples in R_1

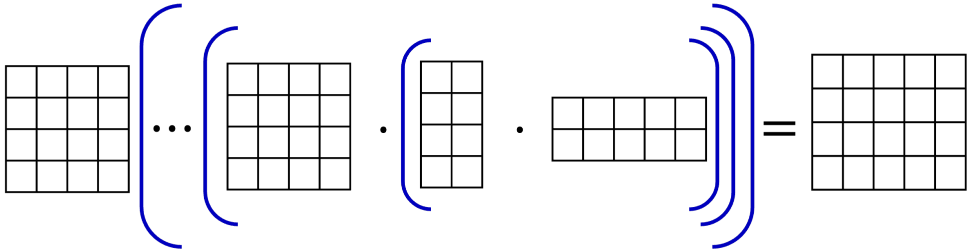
$a \in A_3$ is heavy if it appears in $\geq \tau$ tuples in $\pi_{A_1,A_3} R_1^{\text{light}} \bowtie R_2$

$a \in A_4$ is heavy if it appears in $\geq \tau$ tuples in $\pi_{A_1,A_4} R_1^{\text{light}} \bowtie R_2^{\text{light}} \bowtie R_3$

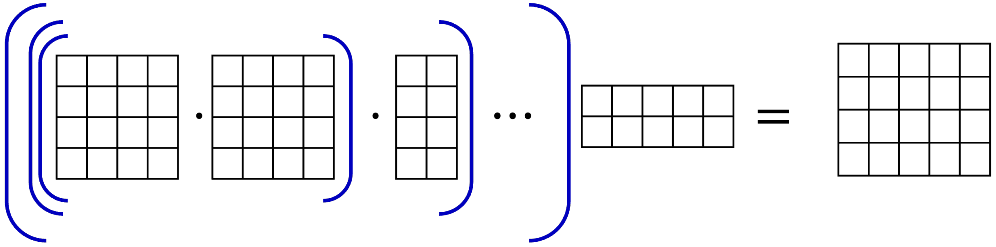


k Subqueries in total!

Which plan to choose?



- Case 1: Some A_i is heavy
  - $|\#$ active values in $A_{k+1}| \leq \frac{OUT}{\tau}$
  - $|V_i(A_i, A_{i+1}, A_{k+1})| \leq |R_i(A_i, A_{i+1})| \cdot |A_{k+1}| \leq N \cdot \frac{OUT}{\tau}$

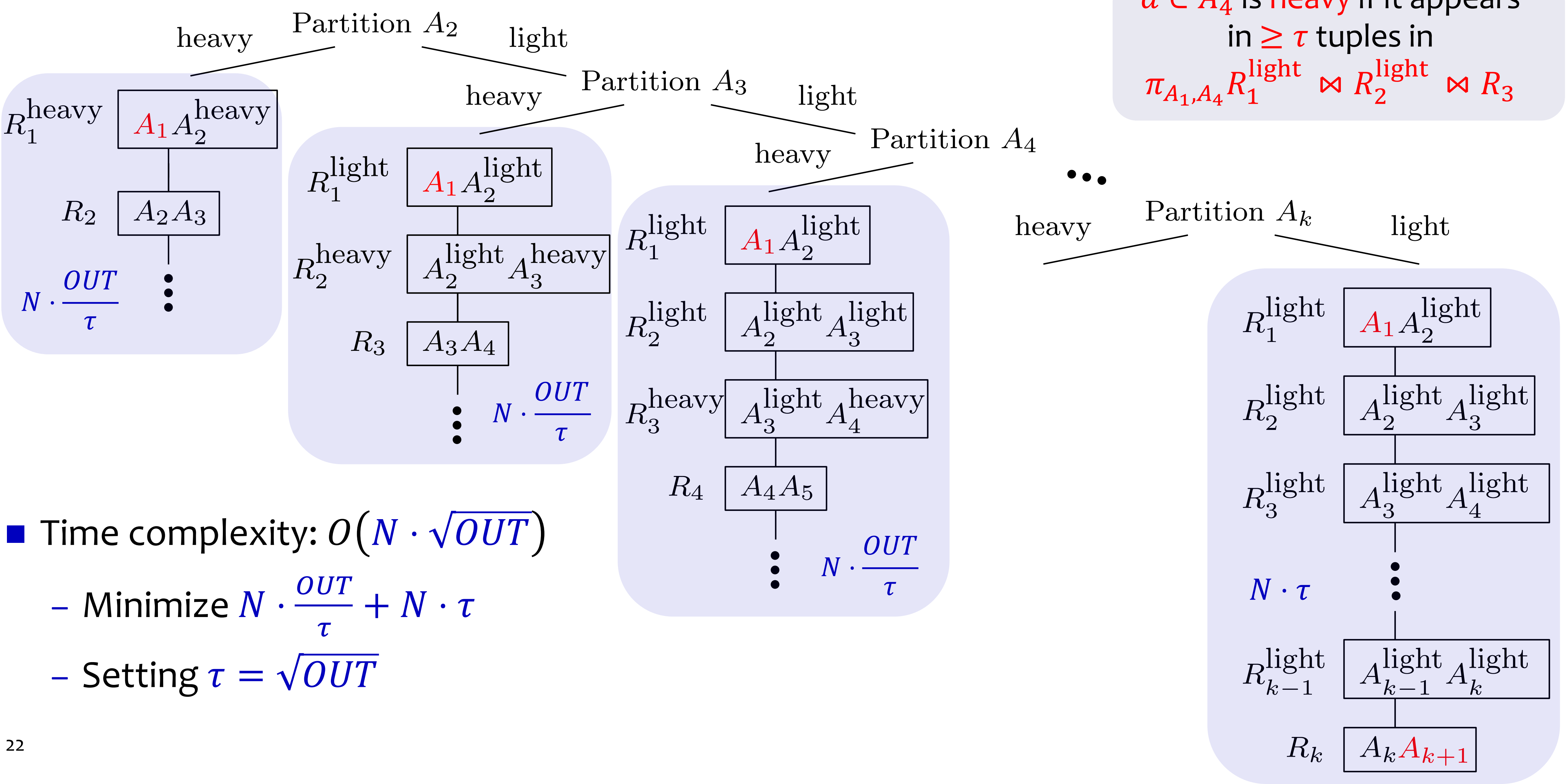


- Case 2: All A_i are light
  - Each value in A_i^{light} joins with $\leq \tau$ values in A_1
  - $|V_{i-1}(A_1, A_i, A_{i+1})| \leq |R_i(A_i, A_{i+1})| \cdot \deg\left(A_i^{\text{light}}, A_1\right) \leq N \cdot \tau$

N — $a \in A_2$ is heavy if it appears in $\geq \tau$ tuples in R_1

$N \cdot \tau$ — $a \in A_3$ is heavy if it appears in $\geq \tau$ tuples in $\pi_{A_1,A_3} R_1^{light} \bowtie R_2$

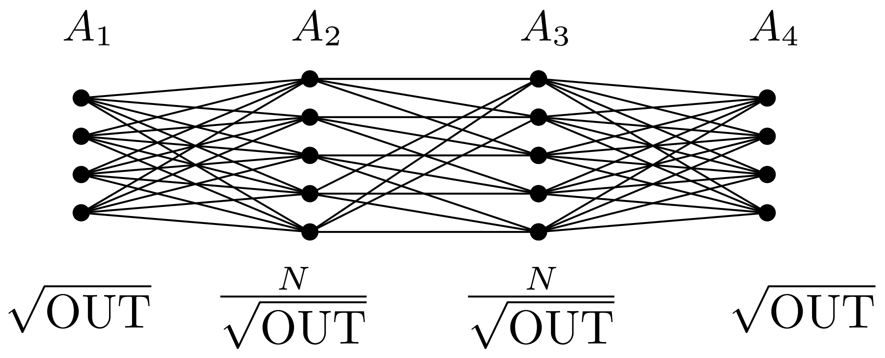
$N \cdot \tau$ — $a \in A_4$ is heavy if it appears in $\geq \tau$ tuples in $\pi_{A_1,A_4} R_1^{light} \bowtie R_2^{light} \bowtie R_3$



- Time complexity: $O\left(N \cdot \sqrt{OUT}\right)$
  - Minimize $N \cdot \frac{OUT}{\tau} + N \cdot \tau$
  - Setting $\tau = \sqrt{OUT}$

Output-Optimality of Hybrid Yannakakis Algorithm

- Intuition 1: Partition does not help this hard instance
- Intuition 2: Any query plan needs to materialize $\Omega(N \cdot \sqrt{OUT})$ intermediate join results



What is next?

- Hybrid Yannakakis Algorithm for Chain MM
  - Output-optimal: $O\left(N \cdot \sqrt{OUT}\right)$
- Hybrid Yannakakis Algorithm for General Join-Aggregate Queries: $O\left(N \cdot OUT^{?} + OUT\right)$
  - Acyclic join queries: $\Theta(N + OUT)$
  - Free-connex queries: $\Theta(N + OUT)$
  - MM: $\Theta\left(N \cdot \sqrt{OUT}\right)$
  - Star MM: $\Theta\left(N \cdot OUT^{1-\frac{1}{k}}\right)$
  - Chain MM: $O\left(N \cdot \sqrt{OUT}\right)$