WEPA 2022

Enumeration and Related Problems in Query Answering

Nofar Carmeli

- Enumeration in query answering
- Enumeration-related tasks
- Enumeration-related tasks in query answering

Example

Employees

Name	Role	Address
Jack	Junior dev	Boston
Jill	Senior dev	Brookline
Joanna	Senior dev	Braintree

Remuneration

Period	Role	Salary
11/2020	Junior dev	4000
11/2020	Senior dev	4500
12/2020	Junior dev	7000
12/2020	Senior dev	7100

Travel

Address	Cost
Boston	50
Brookline	100
Braintree	200

- Join query: $\boldsymbol{Q(N, R, A, P, S, C) \leftarrow Employees(N, R, S), Remuneration(P, R, S), Travel(A, C)}$

Join Results

Name	Role	Address	Period	Salary	Cost
Jack	Junior dev	Boston	11/2020	4000	50
Jill	Senior dev	Brookline	11/2020	4500	100
Joanna	Senior dev	Braintree	11/2020	4500	200
Jack	Junior dev	Boston	12/2020	7000	50
Jill	Senior dev	Brookline	12/2020	7100	100
Joanna	Senior dev	Braintree	12/2020	7100	200

Example

Employees

Name	Role	Address
Jack	Junior dev	Boston
Jill	Senior dev	Brookline
Joanna	Senior dev	Braintree

Remuneration

Period	Role	Salary
11/2020	Junior dev	4000
11/2020	Senior dev	4500
12/2020	Junior dev	7000
12/2020	Senior dev	7100

Travel

Address	Cost
Boston	50
Brookline	100
Braintree	200

- Conjunctive query: $\boldsymbol{Q(N,C) \leftarrow Employees(N,R,A), Travel(A,C)}$

Query Results

Name	Cost
Jack	50
Jill	100
Joanna	200

Challenges

- Many answers
- Many intermediate answers

R

x	y
a1	b1
a2	b1
a3	b1

S

y	z
b1	c1
b1	c2

T

x	z
a2	c1
a4	c2

$$Q_1(x,y,z) \leftarrow R(x,y), S(y,z)$$

x	y	z
a1	b1	c1
a1	b1	c2
a2	b1	c1
a2	b1	c2
a3	b1	c1
a3	b1	c2

$$Q_2(x,y,z) \leftarrow R(x,y), S(y,z), T(x,z)$$

x	y	z
a2	b1	c1

Complexity Guarantees

- Data complexity
  - input = database
  - query size = constant

- Possibly: output $\gg$ input
  (Polynomial number of answers)

- Minimal requirements:
  - Linear time (to read input)
  - Constant time per answer (to print output)

- RAM model
- We allow log factors

Complexity Measures

[**C**, Kröll; TODS 21]

- Linear total time / Amortized constant delay
  - Total time $O(n+m)$

- Linear partial time
  - Time before the ith answer is $O(n+i)$

- Linear preprocessing and constant delay
  - Time before the first answer $O(n)$
  - Time between successive answers $O(1)$

equivalent
assuming
polynomial space
(Cheater's Lemma)

n = input size, m = output size

Type of Results

- Can we solve a task for a given query in a given time complexity?

Yes / No

Conditional Lower Bounds [Bagan, Durand, Grandjean; CSL 07]

Assumption: Boolean $n \times n$ matrices cannot be multiplied in time $O(n^2)$

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$$

Indices of ones

$\boldsymbol{R_1}$

R	C
1	1
1	2
2	2

$\boldsymbol{R_2}$

R	C
1	2
2	2

$\boldsymbol{Q}$

R	C
1	2
2	2

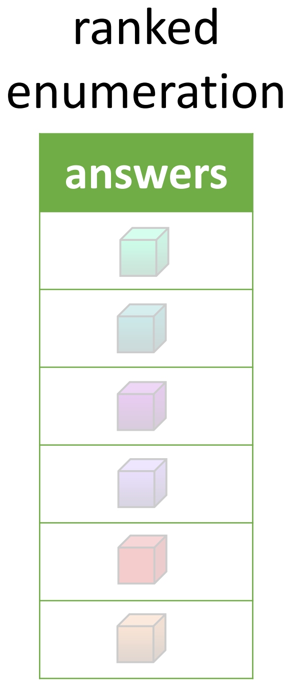
$$Q(x, z) \leftarrow R_1(x, y), R_2(y, z)$$

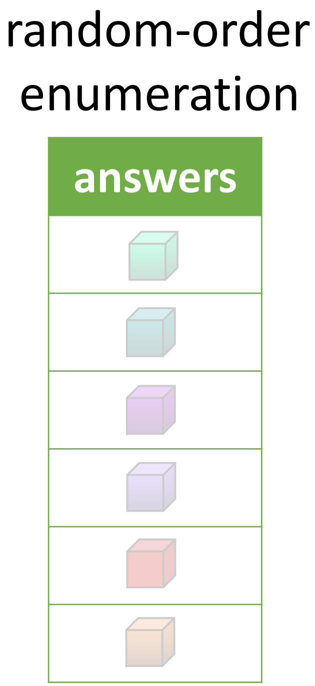
$O(n^2)$ preprocessing + $O(1)$ delay = $O(n^2)$ total $\implies$ no linear preprocessing constant delay

- Enumeration in query answering
- Enumeration-related tasks
- Enumeration-related tasks in query answering

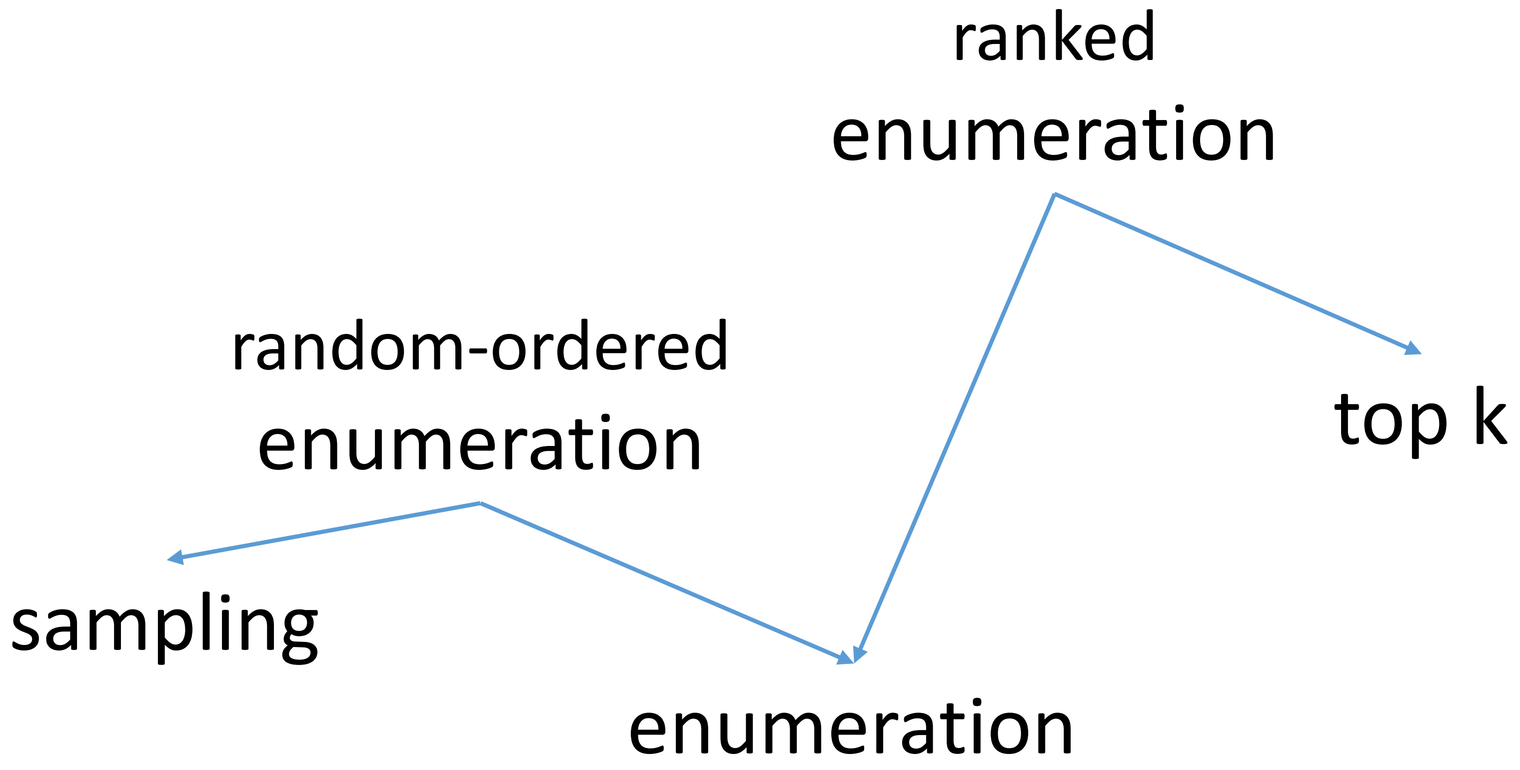
Limitations of Enumeration

- Must produce all answers to get:
  - The best answer
  - The median answer
  - A random answer
- Partial solution: ordered enumeration





Enumeration-Related Problems

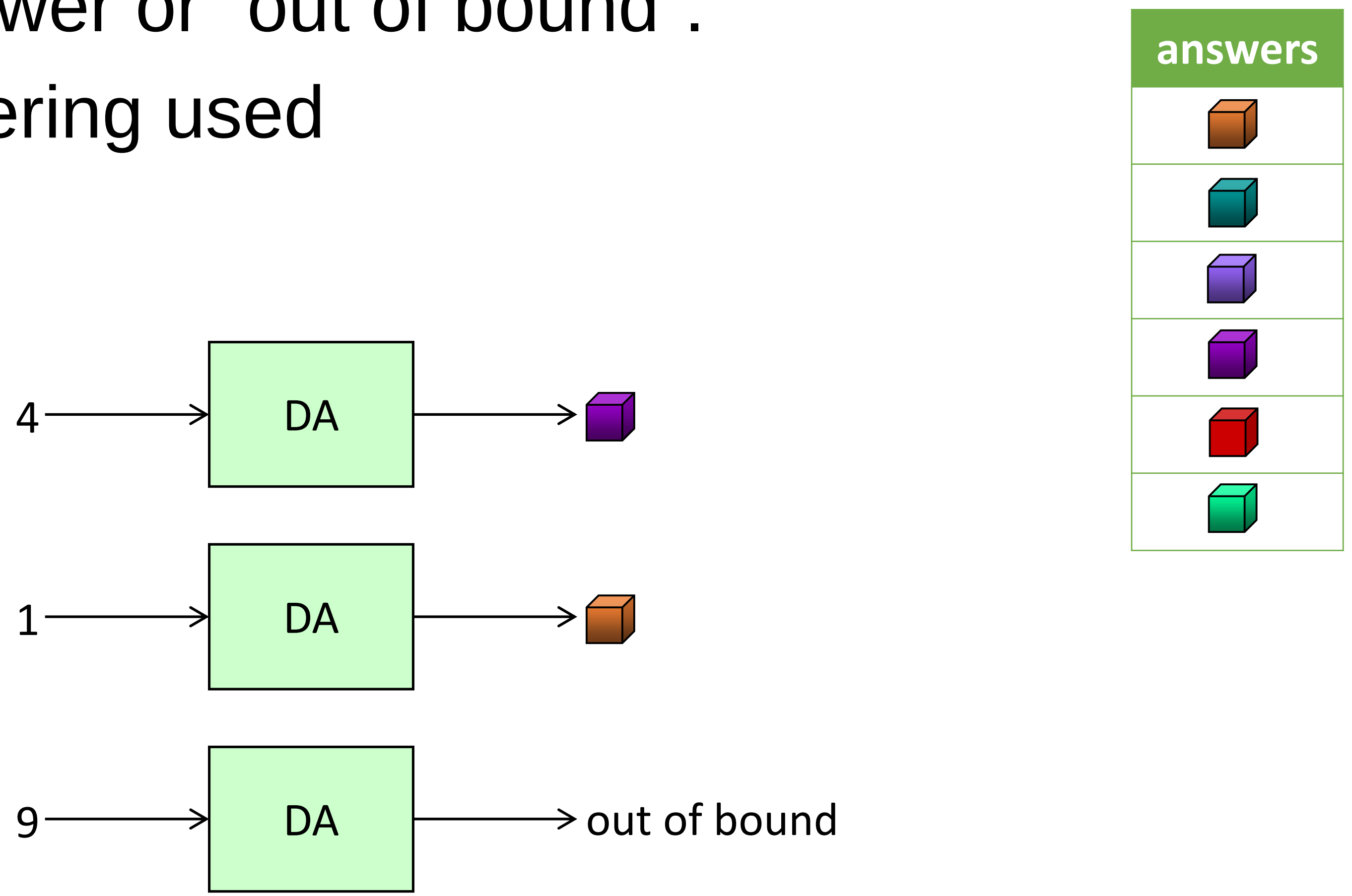


Enumeration as a data structure

- Enumeration provides:
  - Initialize
  - Get next answer
- An array of answers provides access to any index:
  - Initialize
  - Get answer number i

Direct Access Definition

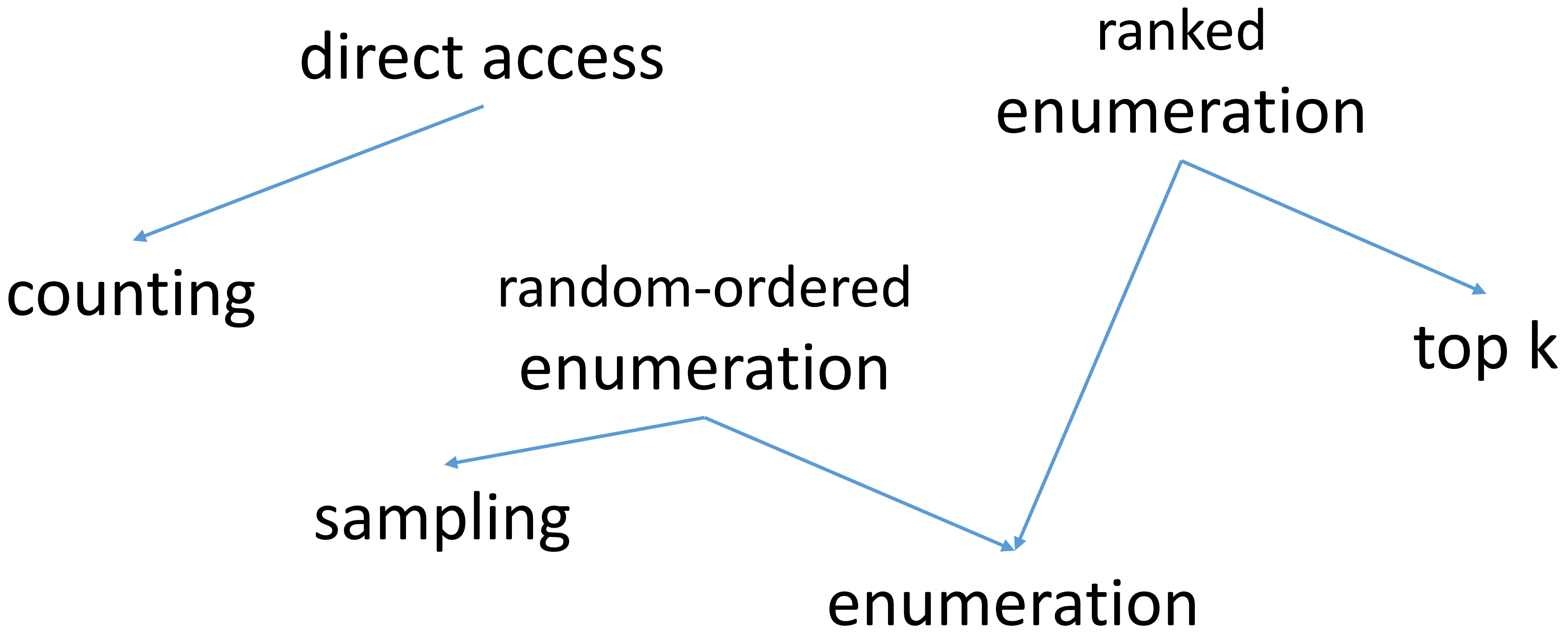
- Given i, returns the i^{th} answer or “out of bound”.
- No constraints on the ordering used



Counting via Direct Access

- Assumption: the number of answers is bounded by a polynomial
- Direct Access returns "out of bound" if needed
  - Allows checking if $|answers| > k$
- Binary search for $|answers|$
  - Requires $O(\log(|answers|))$ calls for Direct Access
  - If $|answers|$ is polynomial, $\log(|answers|) = O(\log(input))$
  - This takes $O(\log(input) \cdot cost(access))$ time

Connection between problems



* with log time per answer after linear preprocessing

Random-Ordered Enumeration via Direct Access

[C, Zeevi, Berkholz, Kimelfeld, Schweikardt; PODS 20]

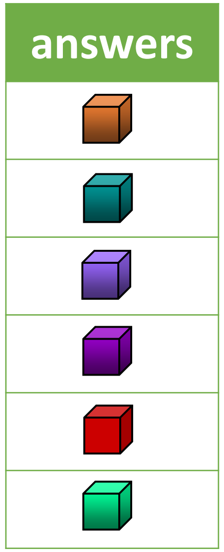
1) Find the number N of answers

6

2) Find a random permutation of 1,…,N

5 6 4 2 1 3

3) Direct access to answers



Direct Access
+
Binary Search

Modified Fisher-Yates Shuffle

Direct Access

Fisher-Yates Shuffle

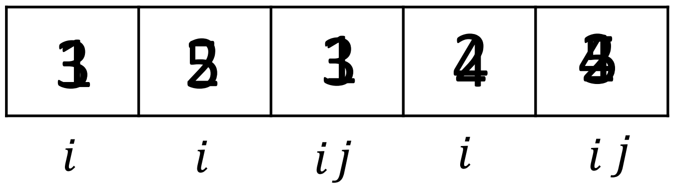
[Durstenfeld 1964]

Place $1, \ldots, n$ in array

For i in $1, \ldots, n$:

    choose j randomly from $\{i, \ldots, n\}$

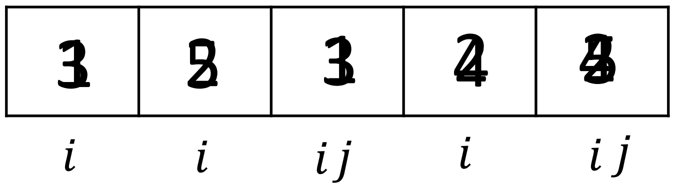
    replace i and j



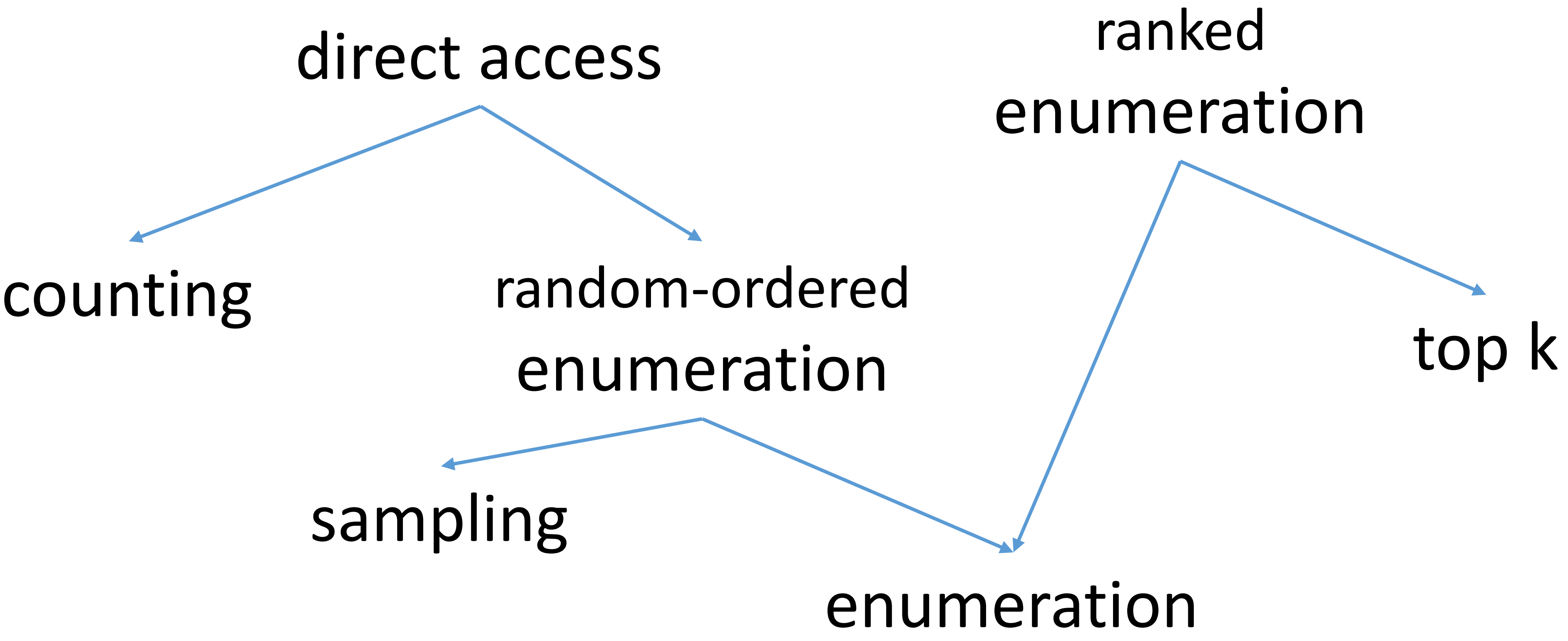
Fisher-Yates Shuffle

Constant delay variant:

place $1, \ldots, n$ in array (lazy initialization)
for i in $1, \ldots, n$:
    choose j randomly from $\{i, \ldots, n\}$
    replace i and j
    print $a[i]$



Connection between problems



* with log time per answer after linear preprocessing

Quantile Computation via Ranked Access

Employees

Name	Role	Address
Jack	Junior dev	Boston
Jill	Senior dev	Brookline
Joanna	Senior dev	Braintree

Remuneration

Period	Role	Salary
11/2020	Junior dev	4000
11/2020	Senior dev	4500
12/2020	Junior dev	7000
12/2020	Senior dev	7100

Travel

Address	Cost
Boston	50
Brookline	100
Braintree	200

- What is the median monthly cost of an employee?
  - Solution 1:
    join, sort, access the middle
  - Solution 2:
    count, ranked enumeration until the middle
  - Solution 3:
    count, ranked access to the middle

Join Results

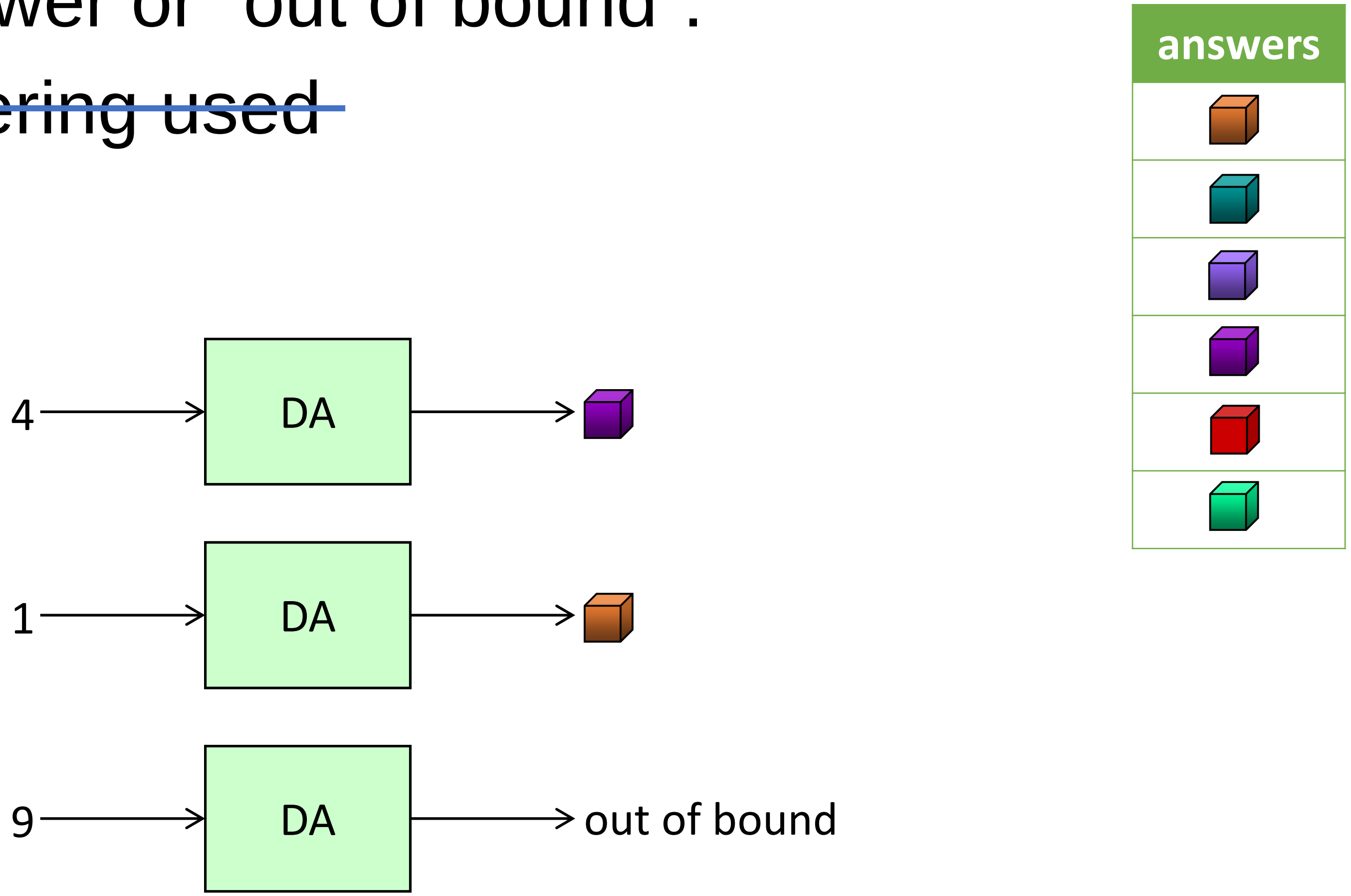
Name	Role	Address	Period	Salary	Cost
Jack	Junior dev	Boston	11/2020	4000	50
Jill	Senior dev	Brookline	11/2020	4500	100
Joanna	Senior dev	Braintree	11/2020	4500	200
Jack	Junior dev	Boston	12/2020	7000	50
Jill	Senior dev	Brookline	12/2020	7100	100
Joanna	Senior dev	Braintree	12/2020	7100	200

3rd

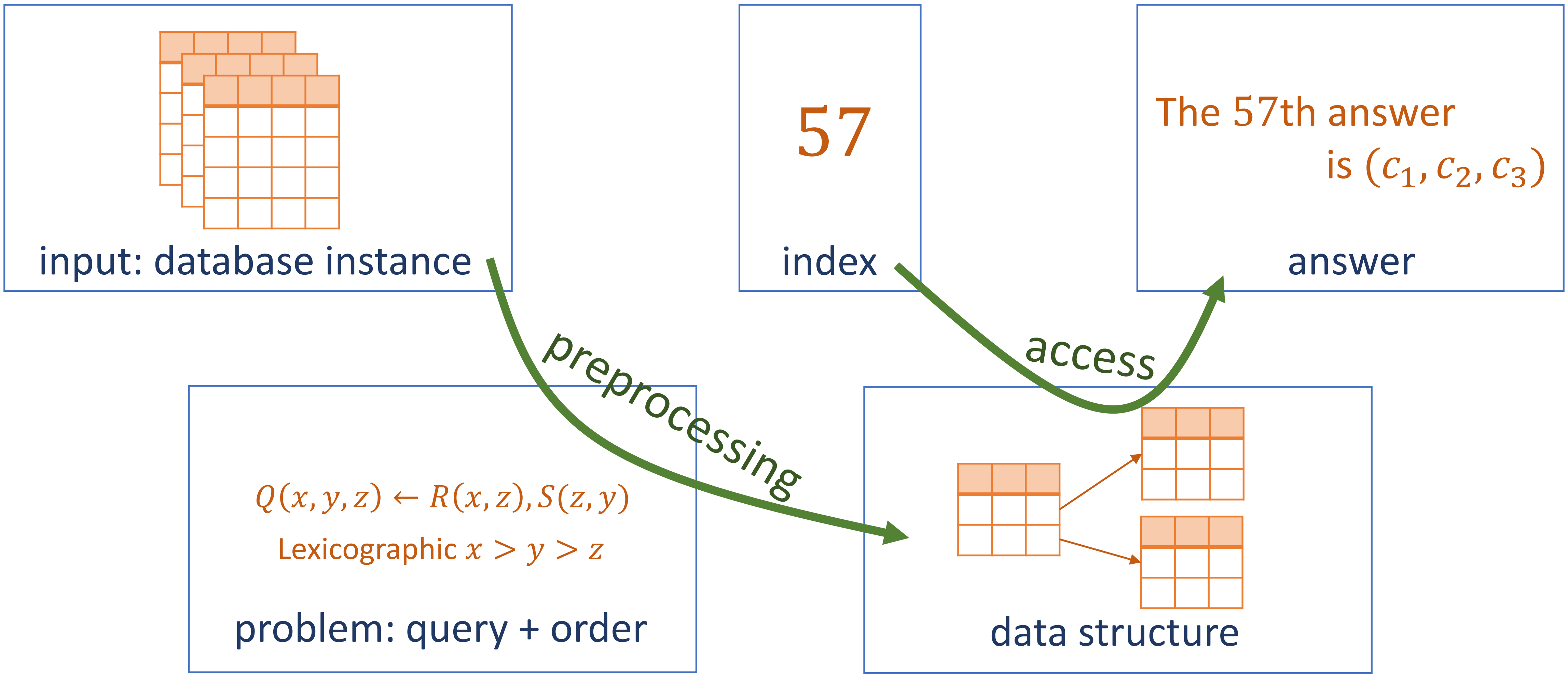
Count = 6

~~Direct~~ Ranked Access Definition

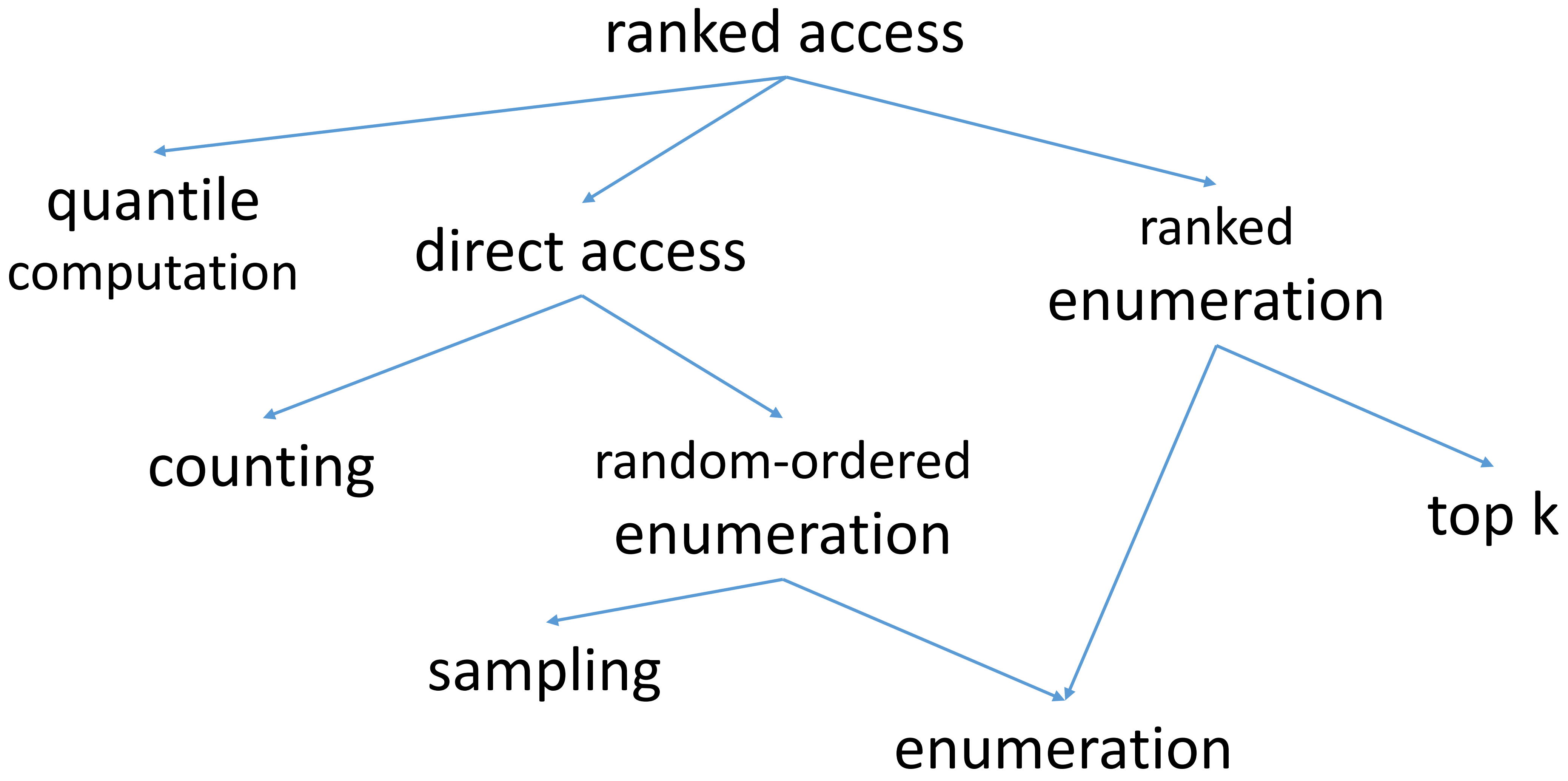
- Given i, returns the i^{th} answer or “out of bound”.
- ~~No constraints on the ordering used~~
  User-specified order



Goal: efficient ranked access



Overview of Tasks



* with log time per answer after linear preprocessing

- Enumeration in query answering
- Enumeration-related tasks
- Enumeration-related tasks in query answering

Challenges

- Many answers
- Many intermediate answers

R

x	y
a1	b1
a2	b1
a3	b1

S

y	z
b1	c1
b1	c2

T

x	z
a2	c1
a4	c2

dangling tuples

$Q_1(x,y,z) \leftarrow R(x,y), S(y,z)$

x	y	z
a1	b1	c1
a1	b1	c2
a2	b1	c1
a2	b1	c2
a3	b1	c1
a3	b1	c2

$Q_2(x,y,z) \leftarrow R(x,y), S(y,z), T(x,z)$

x	y	z
a2	b1	c1

Definitions

[BaganDurandGrandjean 2007]

An acyclic CQ has a graph with:

A free-connex CQ also requires:

1. a node for every atom
possibly also subsets

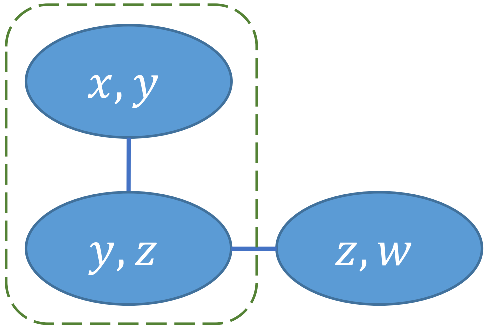
2. tree

3. for every variable X:
the nodes containing X form a subtree

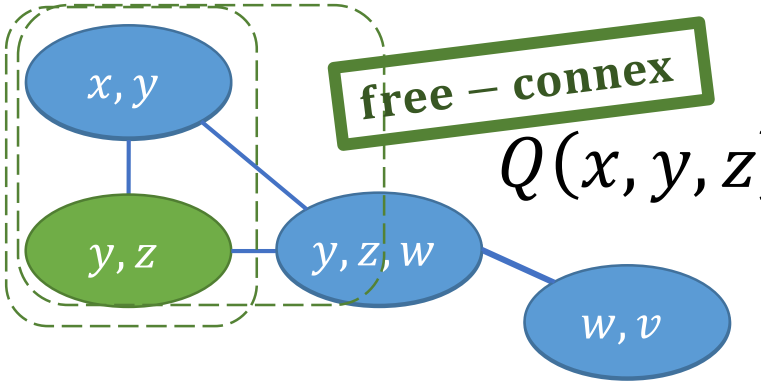
free − connex

acyclic

$$Q(x, y, z) \leftarrow R_1(x, y), R_2(y, z), R_3(z, w)$$



4. a subtree with exactly the free variables



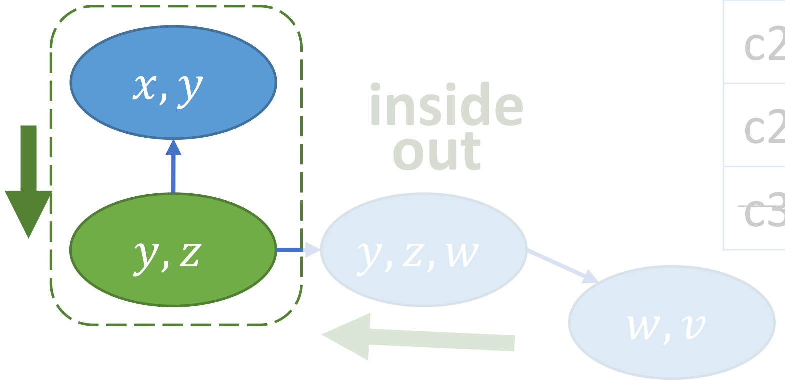
free − connex

$$Q(x, y, z) \leftarrow R_1(x, y), R_2(y, z, w), R_3(w, v)$$

Free-Connex CQs

$$Q(x,y,z) \leftarrow R_1(x,y), R_2(y,z,w), R_3(w,v)$$

x	y
~~a1~~	~~b1~~
a1	b2
a2	b2



w	v
c2	d1
c2	d2
~~c3~~	~~d2~~

y	z
~~b1~~	~~e1~~
b2	e2
~~b3~~	~~e3~~

y	z	w
~~b1~~	~~e1~~	~~c1~~
b2	e2	c2
~~b3~~	~~e3~~	~~c3~~

Reduce to acyclic no projections

1. Find a join tree
2. Remove dangling tuples [Yannakakis81]
3. Ignore existential variables

Then, join efficiently

1. Nested loops

Lower Bound: acyclic non-free-connex

[Bagan, Durand, Grandjean; CSL 07]

Assumption: Boolean $n \times n$ matrices cannot be multiplied in time $O(n^2)$

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$$

Indices of ones

$\boldsymbol{R_1}$

R	C
1	1
1	2
2	2

$\boldsymbol{R_2}$

R	C
1	2
2	2

$\boldsymbol{Q}$

R	C
1	2
2	2

Works also for log delay

works for every self-join-free acyclic non-free-connex conjunctive query

Acyclic non-free-connex: $Q(x, z) \leftarrow R_1(x, y), R_2(y, z)$

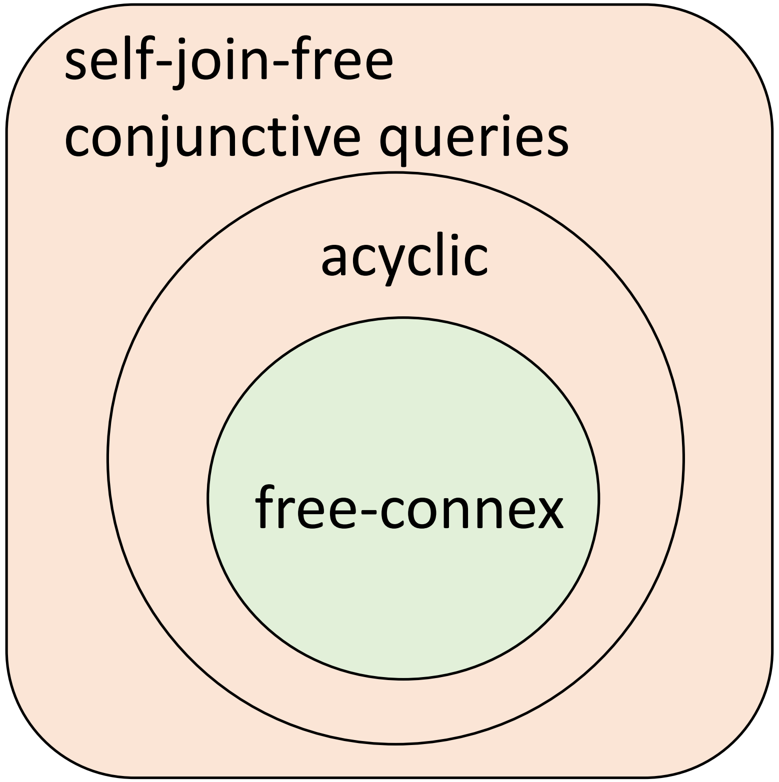
$O(n^2)$ preprocessing + $O(1)$ delay = $O(n^2)$ total $\implies$ no linear preprocessing constant delay

Enumeration Dichotomy

[BaganDurandGrandjean 2007]
[Brault-Baron 2013]

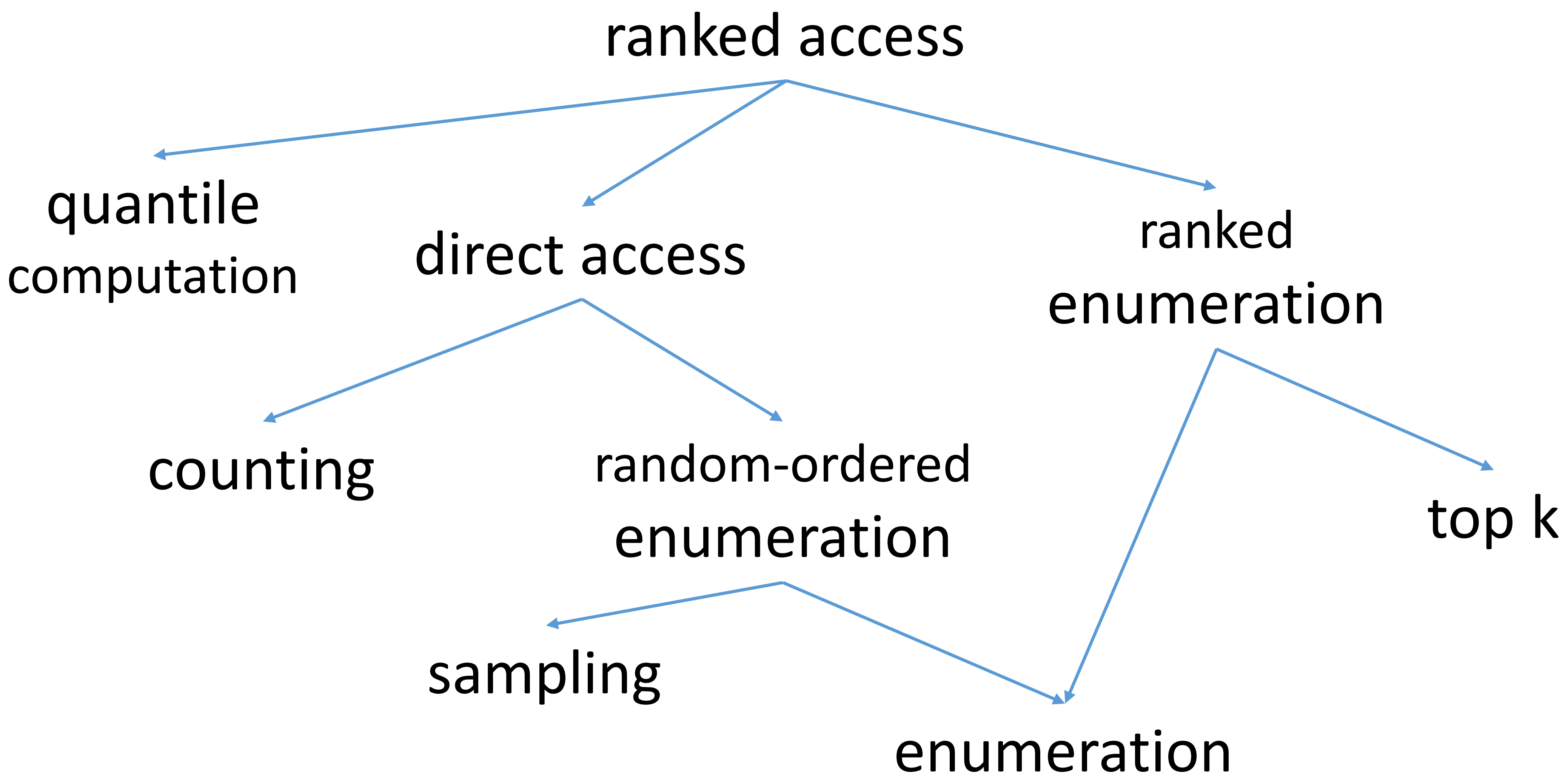
enumerable in
linear preprocessing
and log delay

$\Longleftrightarrow$ free-connex



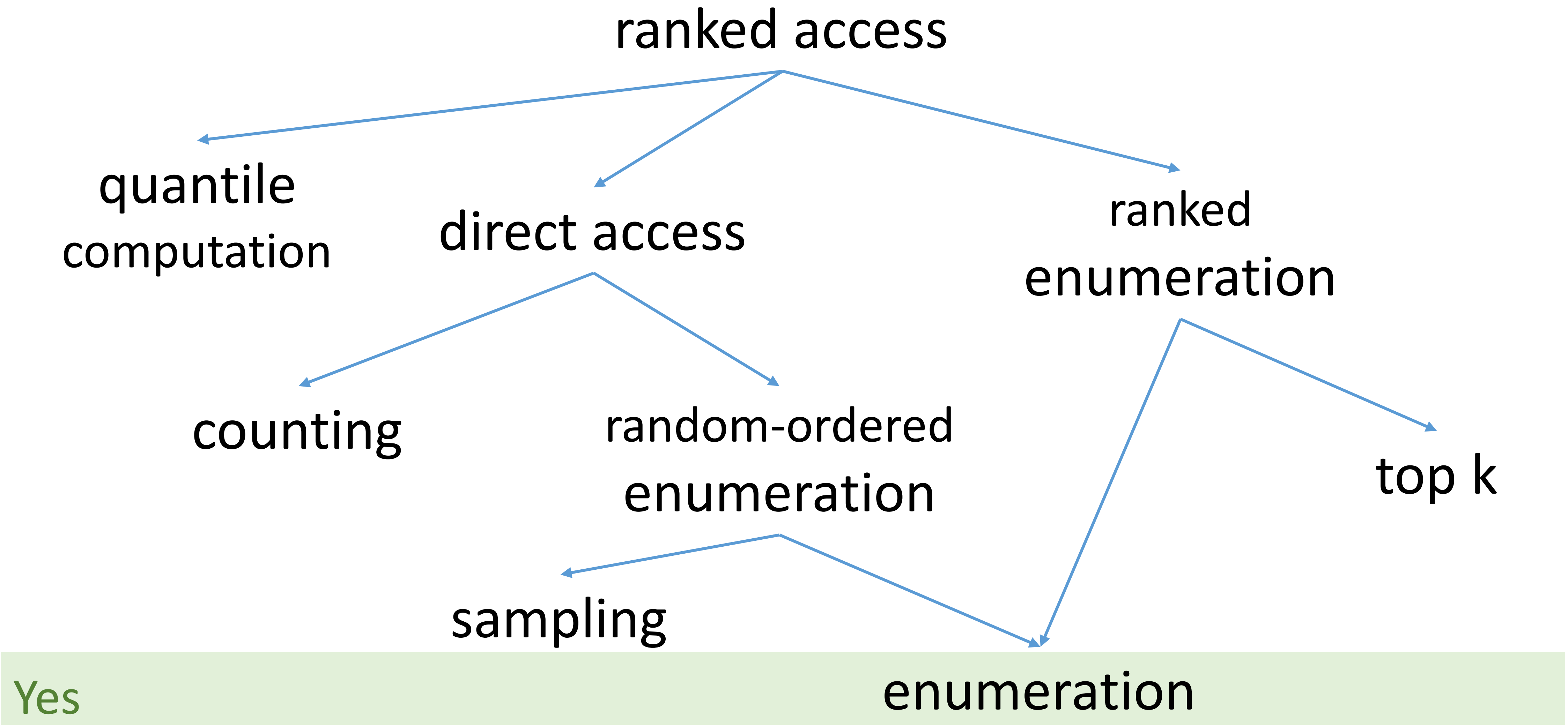
* Assuming the hardness of Boolean matrix multiplication and hyperclique detection

Overview of Tasks



* with log time per answer after linear preprocessing

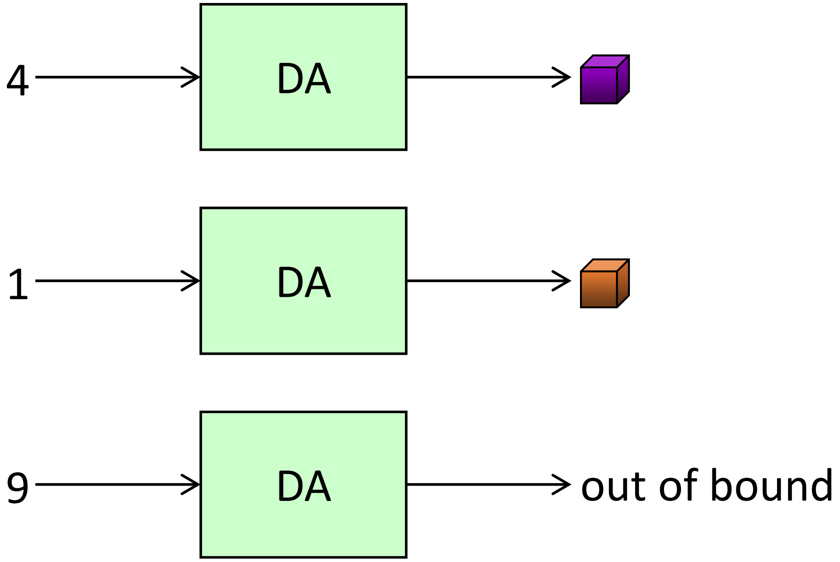
Can be solved efficiently* for all free-connex CQs?

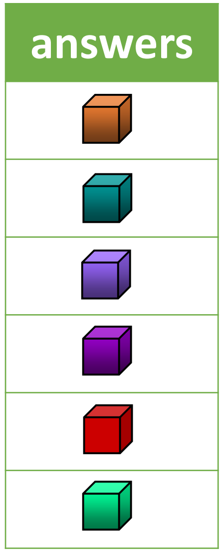


* with log time per answer after linear preprocessing

Direct Access Definition

- Given i, returns the i^{th} answer or “out of bound”.
- No constraints on the ordering used





Direct Access

[Brault-Baron 2013]

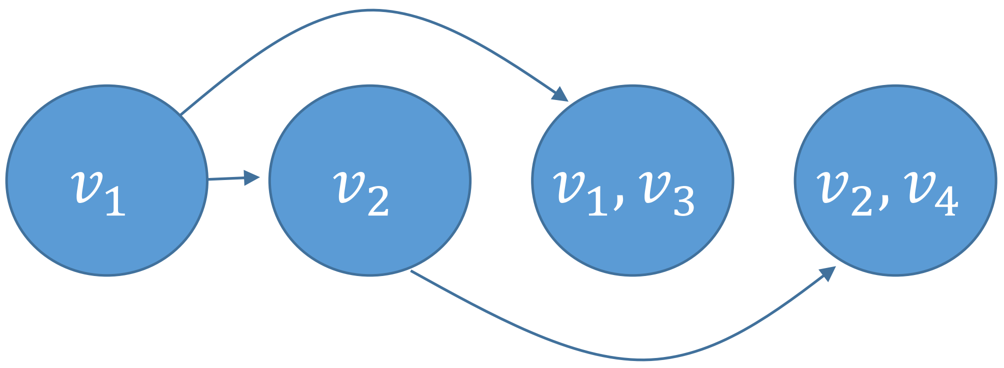
Direct Access Algorithm

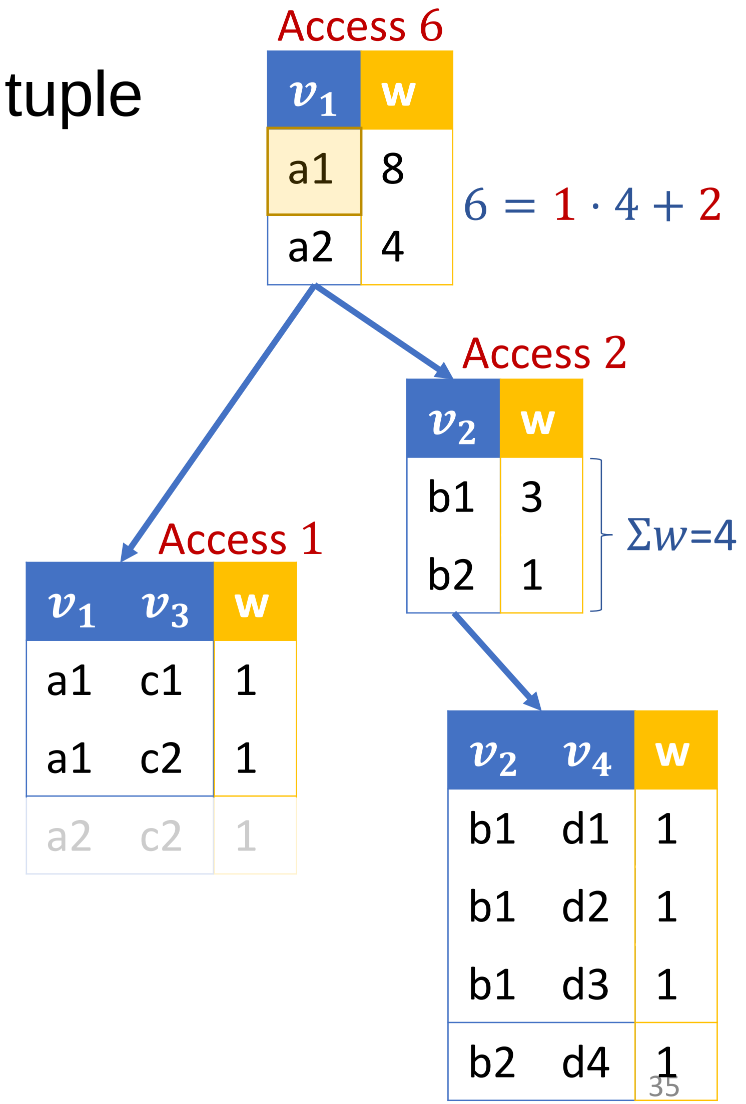
linear preprocessing + log access

Algorithm

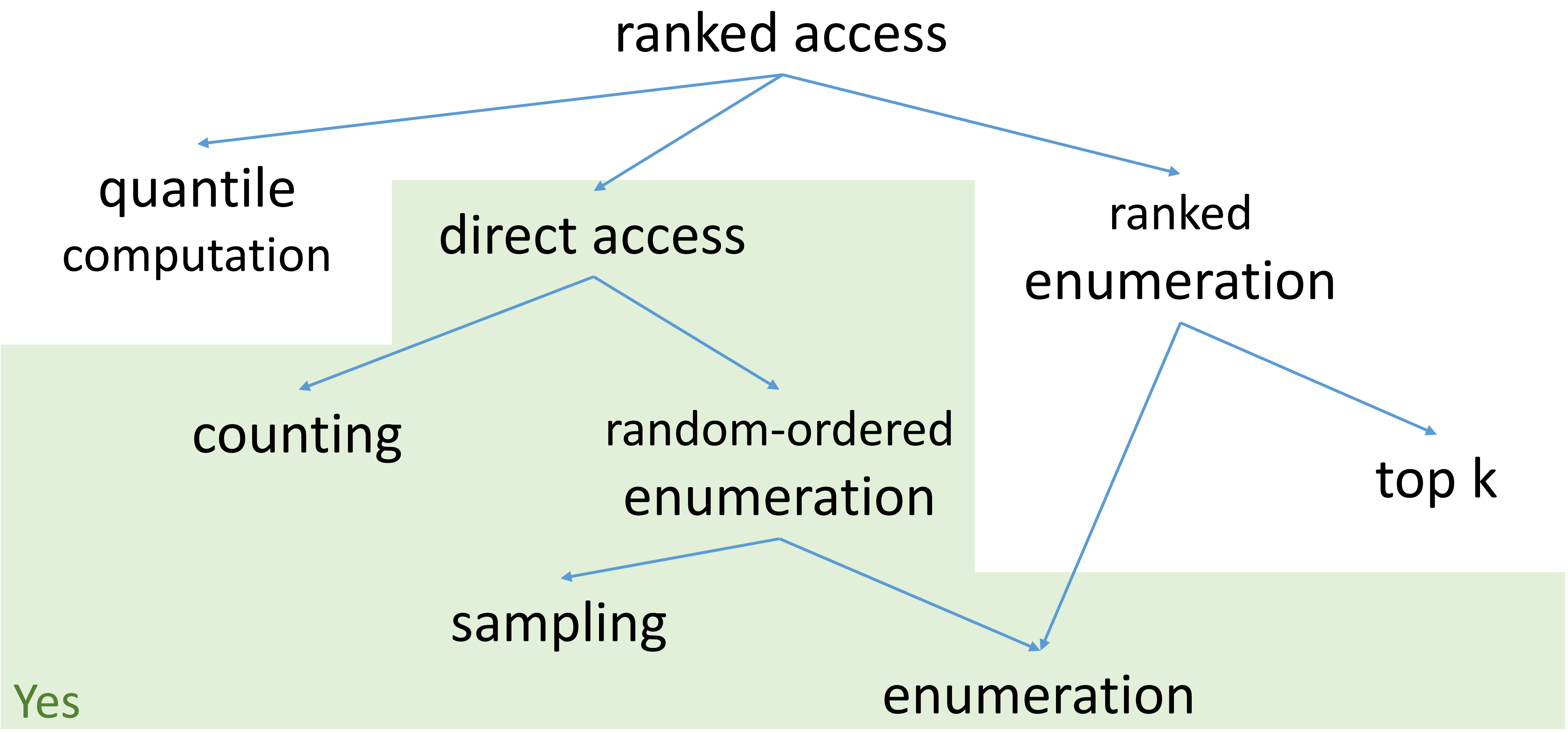
[C, Zeevi, Berkholz, Kimelfeld, Schweikardt; PODS 20]

- Preprocessing:
  - DP up the tree
  - computes how many answers in a subtree use each tuple
- Access:
  - recurse down the tree
  - splits the desired index between the children





Can be solved efficiently* for all free-connex CQs?



* with log time per answer after linear preprocessing

Algorithm

[C, Zeevi, Berkholz, Kimelfeld, Schweikardt; PODS 20]

- Preprocessing:
  - DP up the tree
  - computes how many answers in a subtree use each tuple
- Access:
  - recurse down the tree
  - splits the desired index between the children

Resulting order:

v_1	v_3	v_2	v_4
a1	c1	b1	d1
a1	c1	b1	d2
a1	c1	b1	d3
a1	c1	b2	d4
a1	c2	b1	d1
a1	c2	b1	d2
a1	c2	b1	d3
a1	c2	b2	d4
			…

Access 6

v_1	w
a1	8
a2	4

$6 = 1 \cdot 4 + 2$

Access 1

v_1	v_3	w
a1	c1	1
a1	c2	1
a2	c2	1

Access 2

v_2	w
b1	3
b2	1

Σw=4

v_2	v_4	w
b1	d1	1
b1	d2	1
b1	d3	1
b2	d4	1

Algorithm

[**C**, Zeevi, Berkholz, Kimelfeld, Schweikardt; PODS 20]

- Preprocessing:
  - DP up the tree
  - computes how many answers in a subtree use each tuple
- Access:
  - recurse down the tree
  - splits the desired index between the children

Orders the algorithm can achieve:
DFS of a join tree

v_1	w
a1	8
a2	4

v_1	v_3	w
a1	c1	1
a1	c2	1
a2	c2	1

v_2	w
b1	3
b2	1

v_2	v_4	w
b1	d1	1
b1	d2	1
b1	d3	1
b2	d4	1

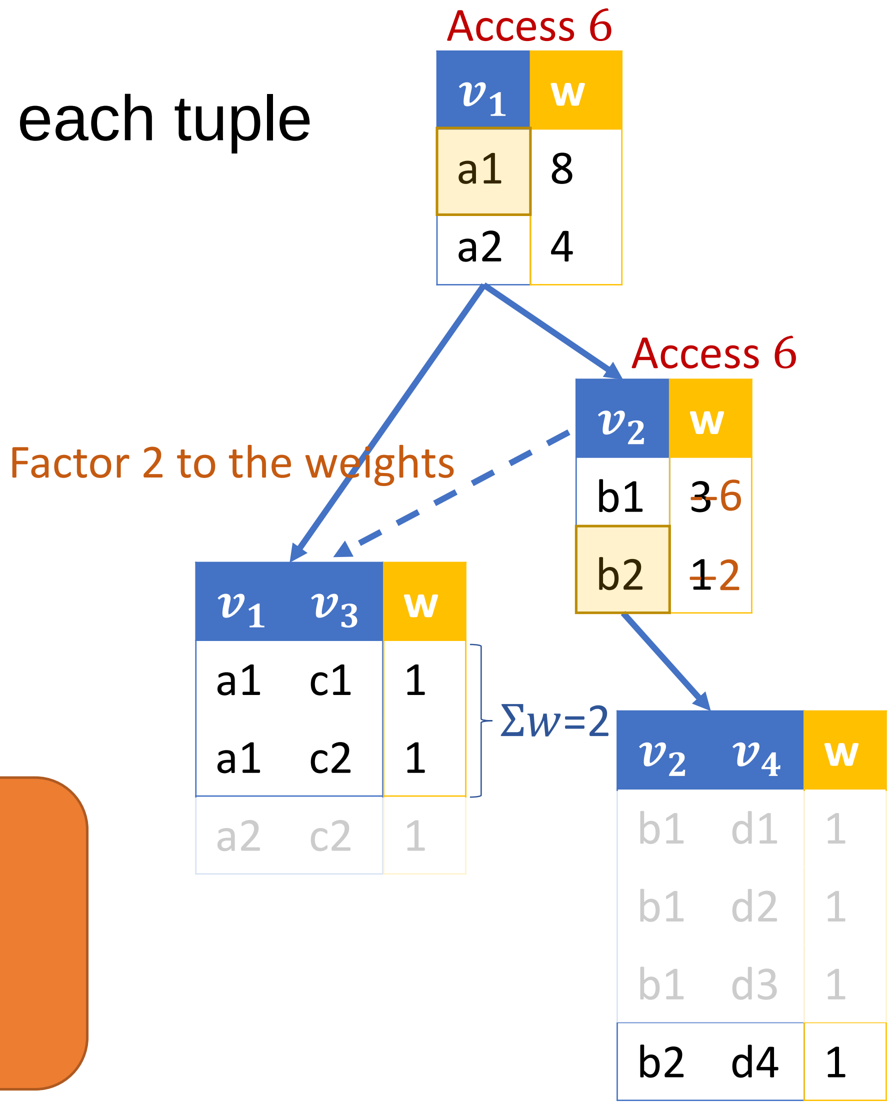
Example

$$Q_2(v_1, v_2, v_3, v_4) \leftarrow R(v_1, v_3), S(v_2, v_4)$$

- Not a DFS of a join tree
- Can it be solved with ideal guarantees?
- Yes!

Algorithm

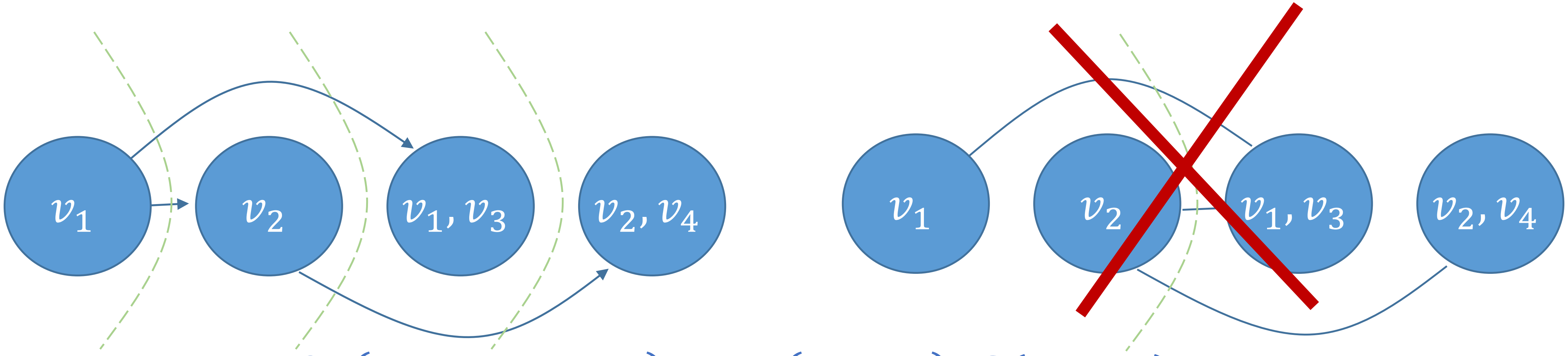
- Preprocessing:
  - DP up the tree
  - computes how many answers in a subtree use each tuple
- Access:
  [**C**, Zeevi, Berkholz, Kimelfeld, Schweikardt; PODS 20]
  - recurse down the tree
  - splits the desired index between the children
- Modified Access:
  [**C**, Tziavelis, Gatterbauer, Kimelfeld, Riedewald; PODS 21]
  - Move children on the fly



Orders the algorithm can achieve:
Orders matching a layered join tree

Layered Trees

- Layered tree for a **CQ** and a variable **ordering**:
  - Join-tree for an inclusive extension
  - Layer i = one node with last variable v_i
  - The induced graph by the first k layers is a tree, for all k



$Q_2(v_1, v_2, v_3, v_4) \leftarrow R(v_1, v_3), S(v_2, v_4)$

Enumeration with Projections via Ranked Access

[**C**, Tziavelis, Gatterbauer, Kimelfeld, Riedewald; PODS 21]

- Reduction:

v_1	v_2	v_3
a_1	b_1	c_1
a_1	b_1	c_2
a_1	b_1	c_3
a_1	b_1	c_4
a_1	b_1	c_5
a_1	b_2	c_1
a_1	b_2	c_2
a_2	b_1	c_1

binary search for next different v_1, v_2 values

Enumerate

$$Q_1(v_1, v_2) \leftarrow R(v_1, v_3), S(v_3, v_2)$$

Not free-connex

using

Lexicographic access

$$Q_2(v_1, v_2, v_3) \leftarrow R(v_1, v_3), S(v_3, v_2)$$

Disruptive trio

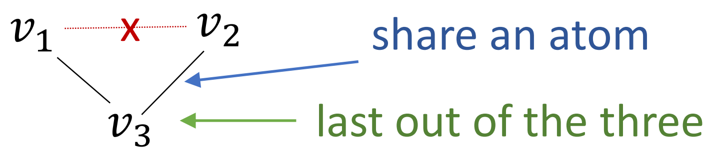
Log number of direct-access calls between answers

Q_1 has no enumeration with polylog delay $\Rightarrow$ Q_2 has no lexicographic access with polylog access time

Hardness Result [**C**, Tziavelis, Gatterbauer, Kimelfeld, Riedewald; PODS 21]

- Can be extended whenever there is a disruptive trio
- Example: $Q_2(v_1, v_2, v_3) \leftarrow R(v_1, v_3), S(v_3, v_2)$

<u>Def: disruptive trio</u>



∃ Layered join tree ⇔ ¬ ∃ disruptive trio

Ranked Access

[**C**, Tziavelis, Gatterbauer, Kimelfeld, Riedewald; PODS 21]

<u>Ranked Access Dichotomy</u>

linear preprocessing + log access

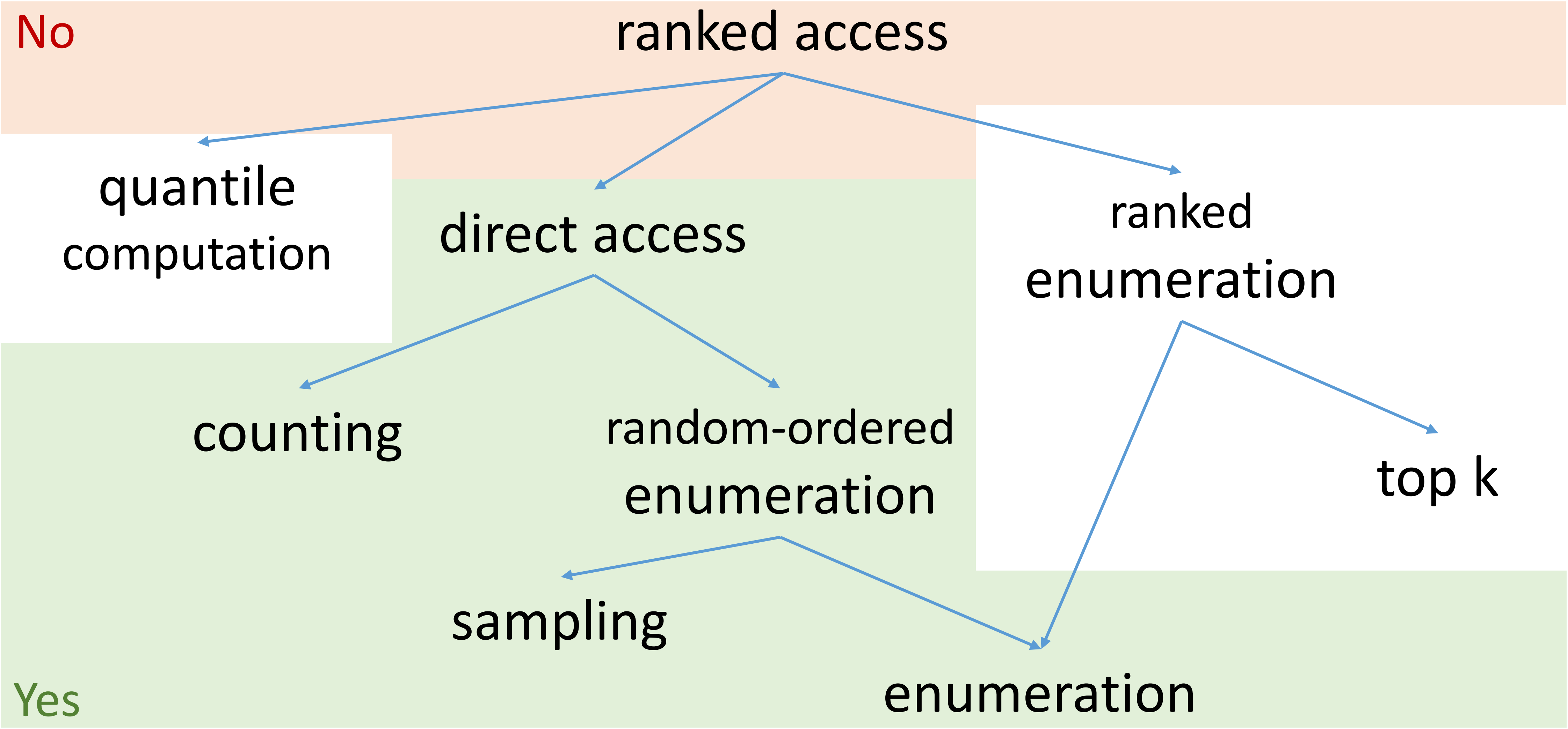
$\Updownarrow$

no disruptive trio in order

* Assuming the hardness of Boolean matrix multiplication and hyperclique detection

Can be solved efficiently* for all free-connex CQs?

For lexicographic orders:



* with log time per answer after linear preprocessing

Ranked Enumeration

[Tziavelis, Gatterbauer, Riedewald; VLDB 21]

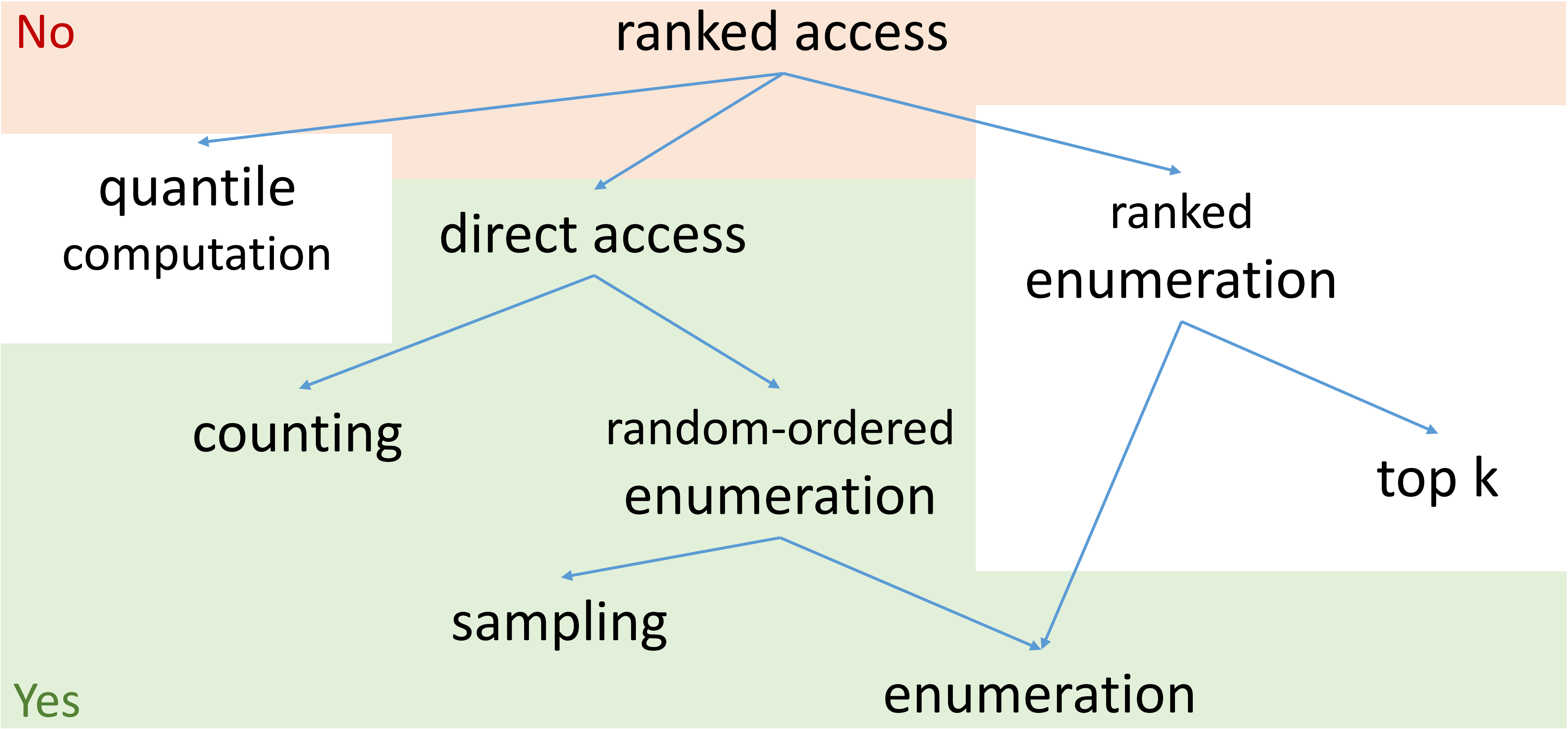
Ranked Enumeration Algorithm

for any lexicographic user-specified order

linear preprocessing + log delay

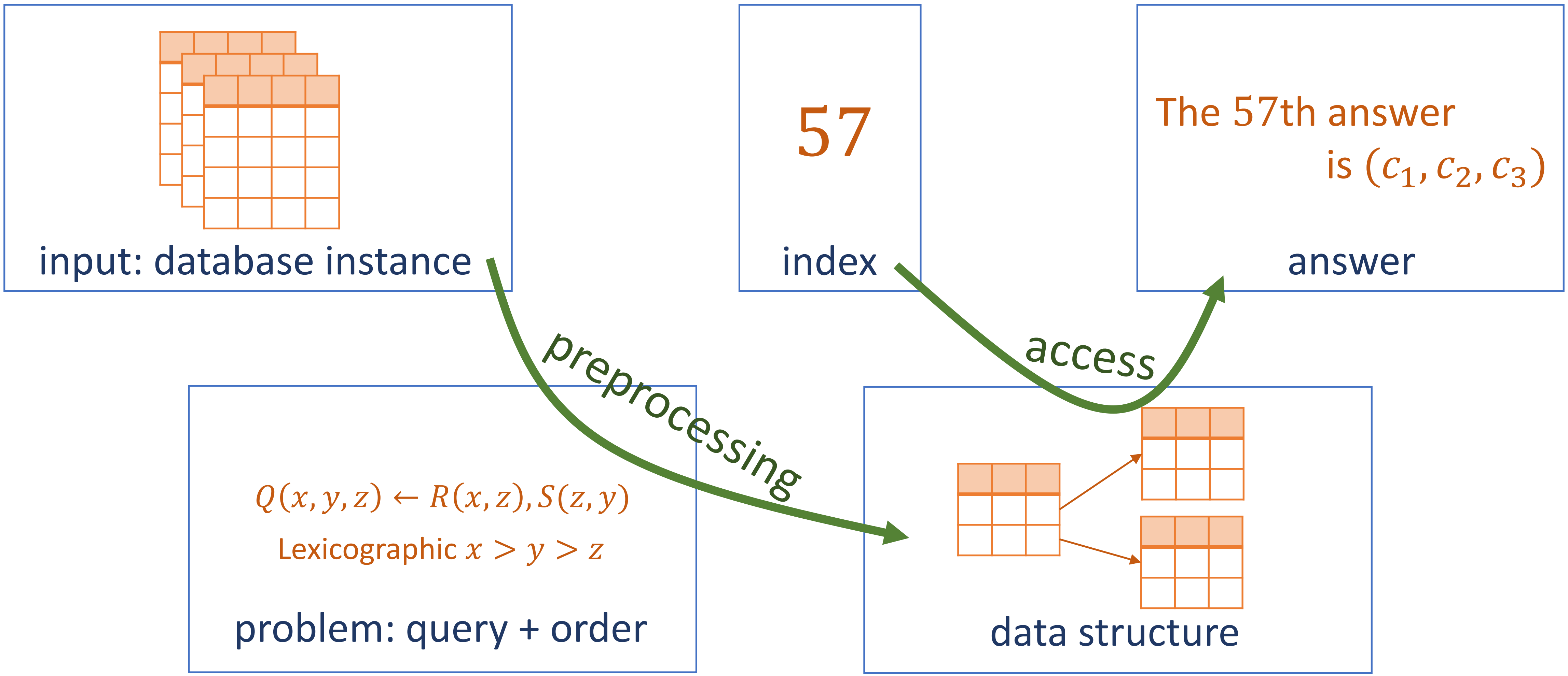
Can be solved efficiently* for all free-connex CQs?

For lexicographic orders:

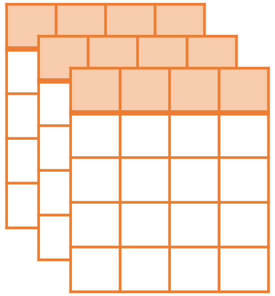


* with log time per answer after linear preprocessing

Ranked Access Problem



Selection Problem (supports a single access call)



Input: database instance

57

index

The 57th answer is (c_1, c_2, c_3)

answer

access

$Q(x, y, z) \leftarrow R(x, z), S(z, y)$

Lexicographic $x > y > z$

Problem: query + order

factorized database

Selection

[**C**, Tziavelis, Gatterbauer, Kimelfeld, Riedewald; PODS 21]

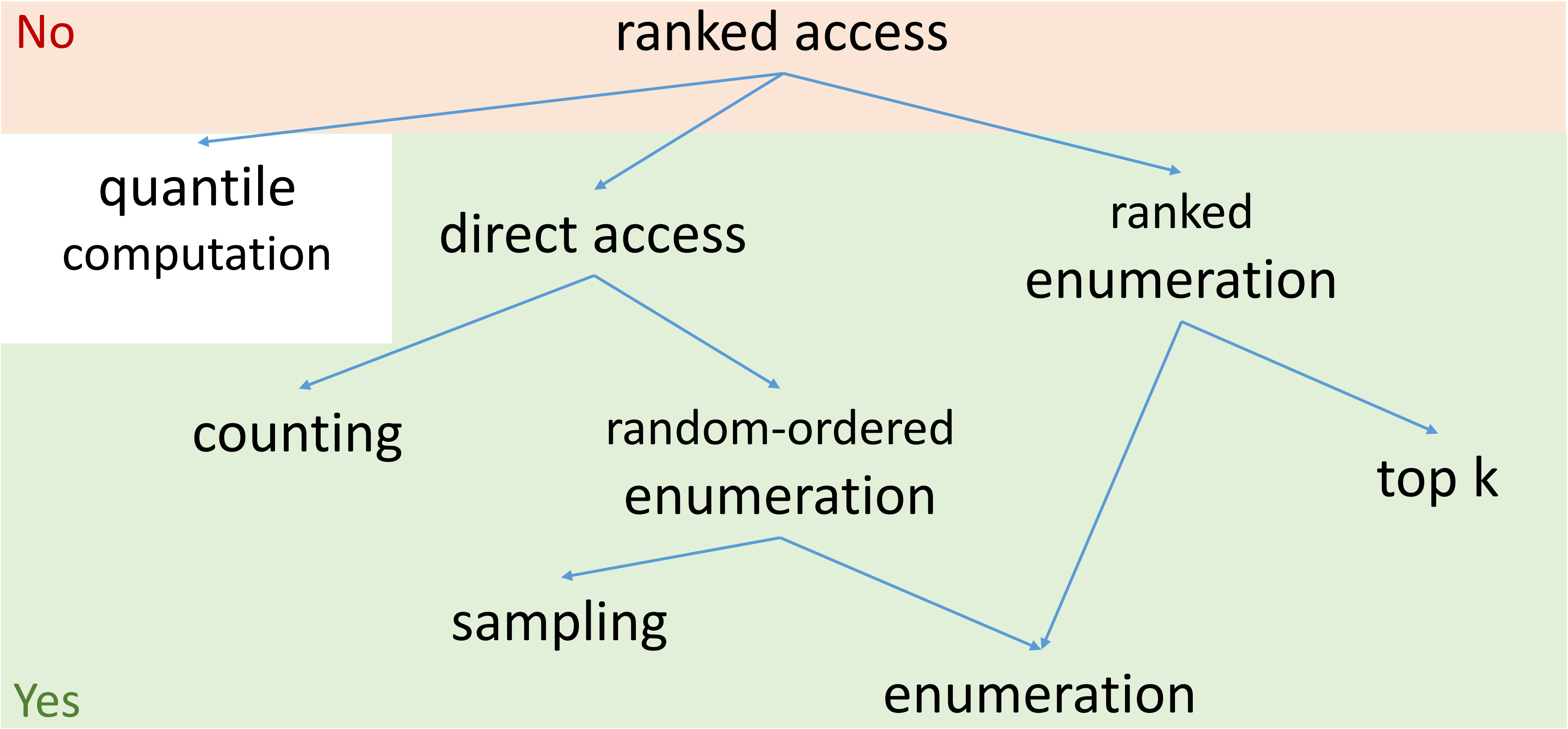
Selection Algorithm

for any lexicographic order
linear time

More tractable <query,order> pairs (than ranked access)
Example: $Q_2(v_1, v_2, v_3) \leftarrow R(v_1, v_3), S(v_3, v_2)$

Can be solved efficiently* for all free-connex CQs?

For lexicographic orders:



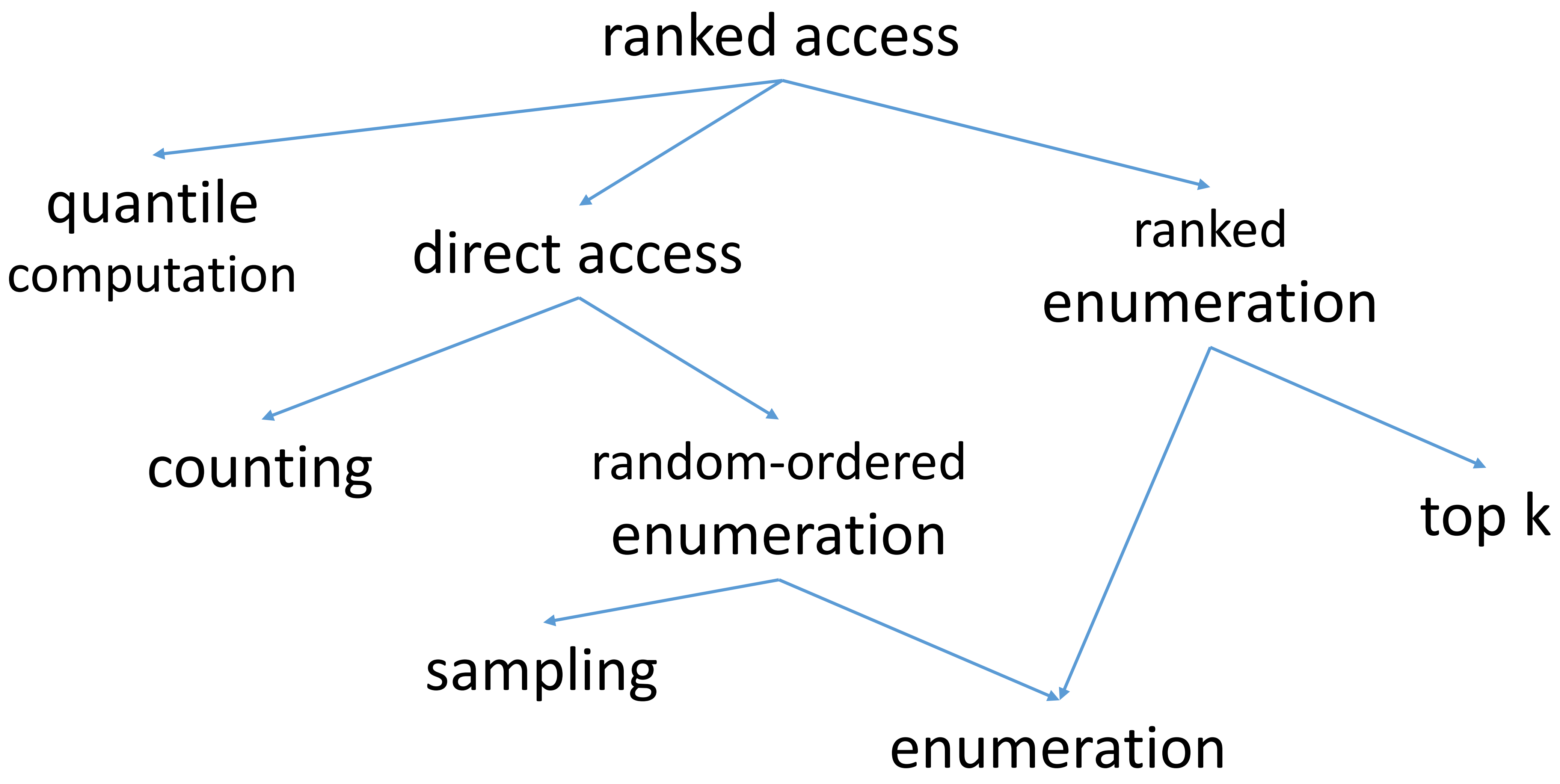
* with log time per answer after linear preprocessing

- Enumeration in query answering
- Enumeration-related tasks
- Enumeration-related tasks in query answering

Conclusion

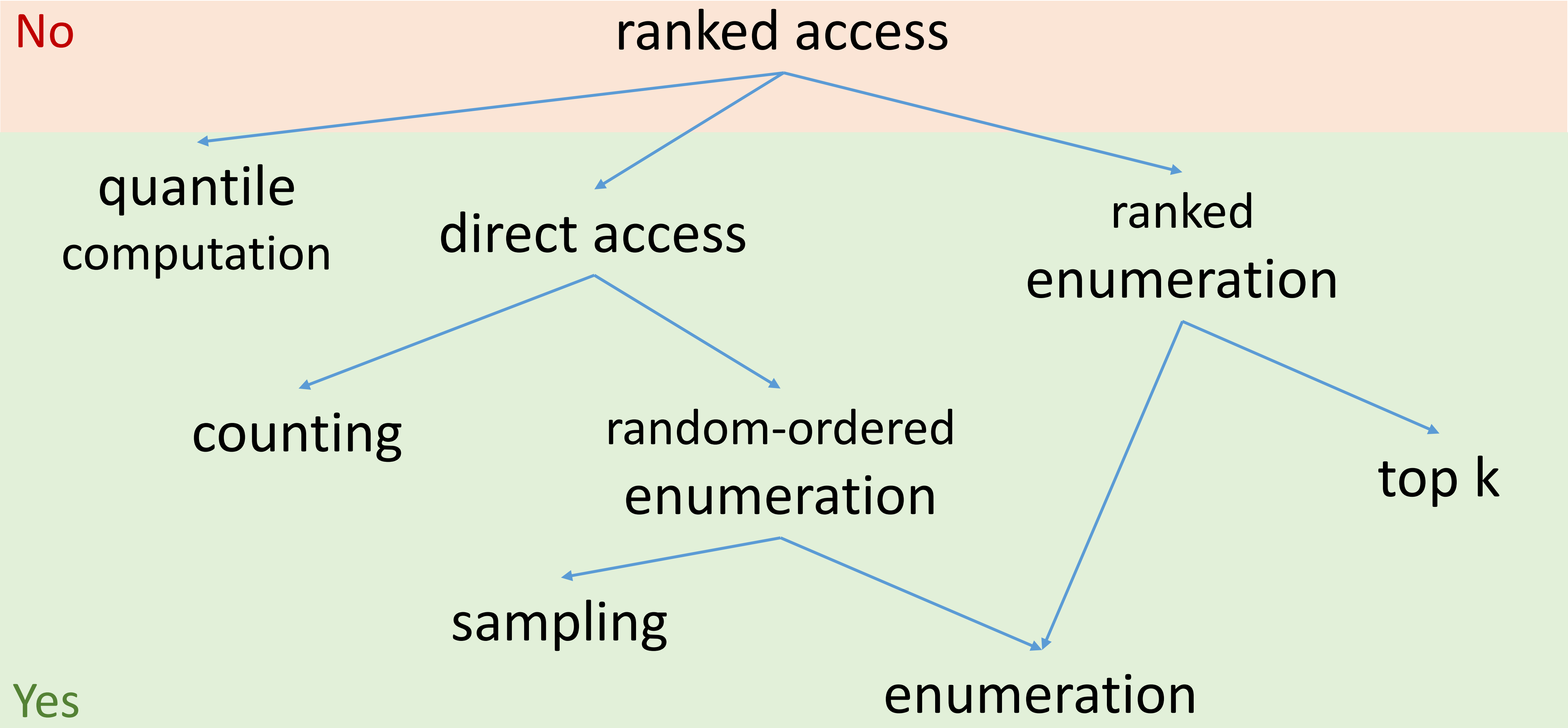
- Change of approach for answering queries:
  materializing answers → structure for accessing answers
- Defined relevant tasks, studied their connections
- Sometimes, can solve more elaborate tasks without higher complexity

Enumeration-Related Problems



Can be solved efficiently* for all free-connex CQs?

For lexicographic orders:



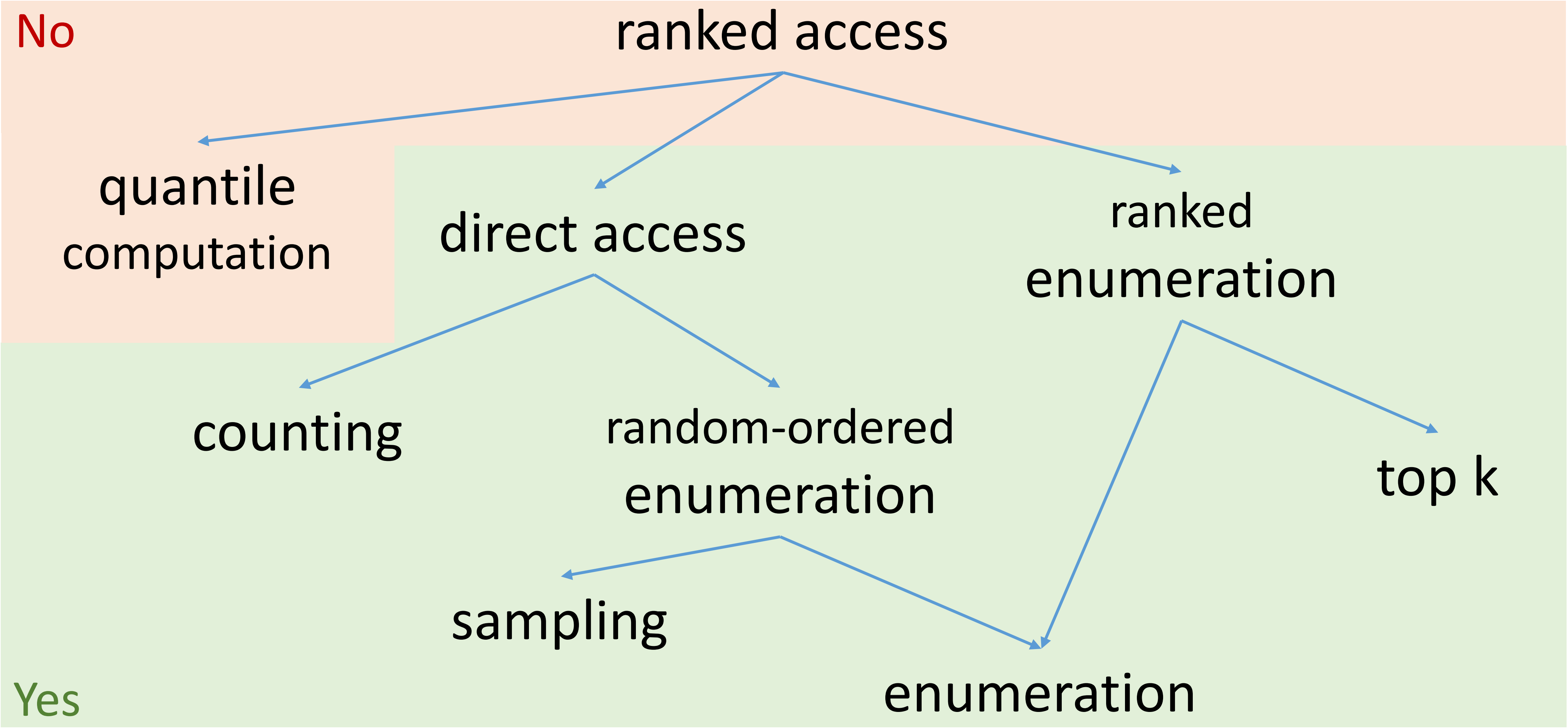
* with log time per answer after linear preprocessing

Outlook

- Handle hard cases (next talk)
- Consider other orders and queries

Can be solved efficiently* for all free-connex CQs?

For sum of weights orders:



* with log time per answer after linear preprocessing

Outlook

- Handle hard cases (next talk)
- Consider other orders and queries
- Enumeration-related tasks in other domains

Extra Slides

Self-Joins

- Lower bounds do not apply with self-joins
- Can they be easier?
  - Yes! [Berkholz, Gerhardt, Schweikardt; SIGLOG News 20]

- A simpler example:

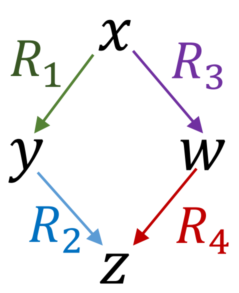
No Constant delay

$$Q_1(x,y,z,w) \leftarrow R_1(x,y), R_2(y,z), R_3(x,w)\ R_4(w,z)$$

$$Q_2(x,y,z,w) \leftarrow R_1(x,y), R_2(y,z), R_1(x,w)\ R_2(z,w)$$

Constant delay

Q_1:



Q_2: