Towards Efficient Random-Order Enumeration for Join Queries Technical Report

0.1 Proof of Lemma 1.

Proof. To simplify the analysis, we assume that $\text{RRAccess}^{Q,\varphi}(i)$ returns the interval $[i,i]$ when $\varphi_Q(i) =\perp$. This choice of a minimal trivial interval, while less efficient, enables us to derive upper bounds on both the total running time and the delay of Algorithm 2.

Total running time. Since $\log|B| \le \log N \le O(\log|Q|)$, all of *B.ban*, *B.pick* and RRAccess run in $O(\log|Q|)$ amortized time. Moreover, since they run at most N times, the total running time of Algorithm 2 is at most $O(N\log|Q|)$.

Enumeration delay. Let $N_i (0 \le i \le |Res(Q)|)$ be the random variable of the number of times that $\text{RRAccess}^{Q,\varphi}$ returns $\perp$ in the process of outputting the first i result tuples (the $(i+N_i)$-th run of $\text{RRAccess}^{Q,\varphi}$ returns the i-th result tuple, and in particular, N_0=0). By the definition of φ_Q, for each tuple $t \in Res(Q)$, there exists only one integer $i \in \mathbb{N}[1,N]$ such that $\varphi_Q(i) = t$. Then there are exactly $|Res(Q)|$ non-trivial integers in $\mathbb{N}[1,N]$, and for each $0 \le i \le |Res(Q)|$, N_i is the number of trivial integers obtained by sampling (without replacement) from $\mathbb{N}[1,N]$ until exactly i non-trivial integers are picked. This implies that for each $1 \le i \le |Res(Q)|$, N_i follows the negative hypergeometric distribution [1] with parameters N, $N-|Res(Q)|$ and i, and its expectation is:

$$E[N_i] = \frac{i\,(N-|Res(Q)|)}{|Res(Q)|+1}. \tag{1}$$

Therefore, the expected number of calls of $\text{RRAccess}^{Q,\varphi}$ between the output of the $(i-1)$-th result (the start of the algorithm when $i=1$) and the output of the i-th result is

$$E\left[N_i - N_{i-1} + 1\right] = \frac{N+1}{|Res(Q)|+1} = O\left(\frac{N}{|Res(Q)|+1}\right). \tag{2}$$

And the expected running time of $\text{RRAccess}^{Q,\varphi}$ between the output of the last result tuple and the end of the algorithm is

$$E\left[N - N_{|Res(Q)|} - |Res(Q)|\right] = \frac{N-|Res(Q)|}{|Res(Q)|+1} = O\left(\frac{N}{|Res(Q)|+1}\right). \tag{3}$$

Then, since $\text{RRAccess}^{Q,\varphi}$ runs in $O(\log^2|Q|)$ worst-case time, the expected delay of Algorithm 2 is $O(\frac{N}{|Res(Q)|+1}\log^2|Q|)$. □

0.2 Proof of Lemma 2.

Proof. We prove the contrapositive: if $\varphi^*_Q(i) \ne \mathit{false}$, then $i \le upp(\psi_r)$. By the definition of φ^*, for each integer $i > 0$ satisfying $\varphi^*_Q(i) \ne \mathit{false}$, there is a tuple $t \in Res(Q)$ along with a root-to-leaf path $u_1,\ldots,u_h$ ($u_1 = r, u_h = u_t$) such that (11) holds. Then, let

$$\begin{aligned}\text{sum}_j &= \sum_{\substack{\psi\prec\psi_{u_{j+1}}\\ \psi\in children(\psi_{u_j})}} upp(\psi)\\ &\le \sum_{\psi\in children(\psi_{u_j})} upp(\psi) - upp(\psi_{u_{j+1}})\\ &\le upp(\psi_{u_j}) - upp(\psi_{u_{j+1}}),\end{aligned} \tag{4}$$

therefore

$$\begin{aligned} i &= \sum_{j=1}^{h-1}\text{sum}_j + 1 \le \sum_{j=1}^{h-1}\left(upp(\psi_{u_j}) - upp(\psi_{u_{j+1}})\right) + 1\\ &= upp(\psi_{u_1}) - upp(\psi_{u_h}) + 1 = upp(\psi_r).\end{aligned} \tag{5}$$

□

0.3 Proof of Theorem 2.

Proof. For $\forall\varepsilon > 0$, if there exists a random-order enumeration algorithm $\mathcal{A}$ for join queries with expected $\tilde{O}(\frac{|Q|^{\rho^*-\varepsilon}}{|Res(Q)|+1})$ delay after a $\tilde{O}(|Q|)$-time preprocessing, we can define $\mathcal{B}$ as the random algorithm obtained by executing $\mathcal{A}$ until the first result tuple is output and then stopping the enumeration. Then $\mathcal{B}$ is a uniform sampling algorithm for join queries, and according to Markov's inequality, $\mathcal{B}$ runs in $\tilde{O}(|Q| + \frac{|Q|^{\rho^*-\varepsilon}}{|Res(Q)|})$ time with high probability. This contradicts Theorem 1. □

0.4 Proof of Lemma 4.

Proof. For each tuple $t \in R$ satisfying ψ, we have $l_i \le t(x_i) \le h_i$ for $\forall i \in \{r_1,\ldots,r_d\}$. This implies $t^\psi_l \preceq t \preceq t^\psi_h$, and consequently $t \in R[t^\psi_l, t^\psi_h]$. Therefore, $R|_\psi \subseteq R[t^\psi_l, t^\psi_h]$. For each tuple $t \in R[t^\psi_l, t^\psi_h]$, we have $t^\psi_l \preceq t \preceq t^\psi_h$. We need to prove that for each $x_i \in \text{att}(R)$, $l_i \le t(x_i) \le h_i$. Let s be a split position of ψ, then:

1. For $\forall x_i \in \text{att}(R)$ with $\forall 1 \le i < s$, since $l_i = h_i$, it follows that $l_i = t(x_i) = h_i$. Otherwise, one of the conditions $t \succeq t^\psi_l$ or $t \preceq t^\psi_h$ would be violated.
2. If $x_s \in \text{att}(R)$ then $l_s \le t(x_s) \le h_s$, otherwise $t^\psi_l \preceq t \preceq t^\psi_h$ would be violated.
3. For $\forall x_i \in \text{att}(R)$ with $\forall s < i \le n$, since $l_i = \min_Q(x_i)$ and $h_i = \max_Q(x_i)$, it follows that $l_i \le t(x_i) \le h_i$.

Then t satisfies ψ, *i.e.*, $t \in R|_\psi$. Therefore, $R[t^\psi_l, t^\psi_h] \subseteq R|_\psi$ and then we have $R[t^\psi_l, t^\psi_h] = R|_\psi$, which implies that $R.cnt(\psi) = |R|_\psi|$. □

0.5 Proof of Lemma 6.

Proof. Since $upp(\psi) \le 1$, two cases need to be discussed. Firstly, for the case $upp(\psi) = 0$, we have $|Res(Q|_\psi)| \le upp(\psi) = 0$, which implies that $Res(Q|_\psi) = \emptyset$. Secondly, for the case $upp(\psi) = 1$, we have $|Res(Q|_\psi)| \le 1$. Let $s = \max\{i | l_i \ne h_i\}$, by Property 1, there exists at most one integer $p \in \mathbb{N}[l_s, h_s]$ such that

- $upp([[l_1,h_1],\ldots,[l_s,p-1],\ldots,[l_n,h_n]]) = 0$,
- $upp([[l_1,h_1],\ldots,[p,p],\ldots,[l_n,h_n]]) = 1$,
- $upp([[l_1,h_1],\ldots,[p+1,h_s],\ldots,[l_n,h_n]]) = 0$.

If $upp([[l_1,h_1],\ldots,[p,p],\ldots,[l_n,h_n]]) = 0$ follows for every $p \in \mathbb{N}[l_s,h_s]$, we have $Res(Q|_\psi) = \emptyset$. Otherwise, it is able to calculate the only integer $p \in \mathbb{N}[l_s, h_s]$ by a multi-head binary search (introduced in Section 4.4) in $O(\log |Q|)$ time. Then apply the same procedure to the next unfixed position of $[[l_1, h_1], \ldots, [p, p], \ldots, [l_n, h_n]]$, and continue this process iteratively on other positions as needed. If this process finally obtains $\psi^* = [[l_1^*, h_1^*], \ldots, [l_n^*, h_n^*]]$ such that $upp(\psi^*) = 1$ and $\forall 1 \le i \le n, l_i^* = h_i^*$, then $(l_1^*, \ldots, l_n^*)$ is the only tuple in $Res(Q|_\psi)$. Otherwise, we have $Res(Q|_\psi) = \emptyset$. The time it takes to decide whether $Res(Q|_\psi) = \emptyset$ and calculate ψ^* (if exists) is at most $O(n \log |Q|)$, which is $O(\log |Q|)$ in data complexity. □

0.6 Proof of Lemma 7.

Proof. Otherwise, there exists $s < s' \le n$ such that $[l_{s'}, h_{s'}] \neq [\min_Q(x_{s'}), \max_Q(x_{s'})]$, then for $\forall i \neq s'$, i is not a split position of ψ. Moreover, s' is not a split position since $l_s \neq h_s$, then ψ is not a prefix range filter, which leads to a conflict. □

0.7 Proof of Lemma 8.

Proof. For the convenience of the proof, a dummy element $+\infty$ is appended to the end of each sequence, that is, $A_i[|A_i|] = +\infty$ for each $1 \le i \le k$. If $F(|A_1|, \ldots, |A_k|) \le T$, the algorithm obviously returns p^*. Then for the case where $F(|A_1|, \ldots, |A_k|) > T$, we show that Algorithm 4 maintains the invariant

$$A_i[l_i] \le p^* \le A_i[r_i], \quad \forall\, 1 \le i \le k, \tag{6}$$

and then prove that Algorithm 4 correctly returns p^* after at most $O(\log \sum_{i=1}^k |A_i|)$ iterations.

Base case. Initially, for each $1 \le i \le k$ we set $l_i = 0$ and $r_i = |A_i|$. Since each A_i is sorted,

$$A_i[0] \le \min A_i \le p^* \le \max A_i \le A_i[n], \tag{7}$$

then the invariant holds before the first iteration.

Inductive step. Suppose at the start of some iteration we have $A_i[l_i] \le p^* \le A_i[r_i]$ for any $1 \le i \le k$. We aim to demonstrate that the invariant is preserved after the update performed in the current iteration. Without loss of generality, we assume that for any $1 \le i \le k$, $r_i - l_i > 1$ satisfies. (If some i already has $r_i - l_i \le 1$ and the algorithm has not yet terminated, then by the invariant we have $A_i[l_i] < p^* \le A_i[r_i]$ and $N_i(A_i[r_i]) \le N_i(A_i[l_i] + 1) = r_i$, then $N_i(p^*) = r_i = m_i$, and the problem reduces to the remaining $k-1$ sequences.) In each iteration, the algorithm modifies exactly one of the two bounds: it either updates the lower bound $l_{i_{\min}}$ for some index $i_{\min}$, or updates the upper bound $r_{i_{\max}}$ for some index $i_{\max}$. We discuss the two cases:

Case 1: $F(m_1, \ldots, m_k) \le T$.

Let $p = A_{i_{\min}}[m_{i_{\min}}] = \min_{r_i - l_i > 1} A_i[m_i]$. Since

$$N_{i_{\min}}(p) = \left|\{x \in A_{i_{\min}} \mid x < A_{i_{\min}}[m_{i_{\min}}]\}\right| \le m_{i_{\min}}, \tag{8}$$

and for each $i \neq i_{\min}$,

$$N_i(p) \le N_i\big(A_i[m_i]\big) = \left|\{x \in A_i \mid x < A_i[m_i]\}\right| \le m_i, \tag{9}$$

it follows that

$$F\big(N_1(p), \ldots, N_k(p)\big) \le F(m_1, \ldots, m_k) \le T. \tag{10}$$

Hence $p^* \ge p$, then after updating $l_{i_{\min}} \leftarrow m_{i_{\min}}$, we still have $A_{i_{\min}}[l_{i_{\min}}] \le p^* \le A_{i_{\min}}[r_{i_{\min}}]$. For all $i \neq i_{\min}$, neither l_i nor r_i changes, so the invariant is preserved after the update.

Case 2: $F(m_1, \ldots, m_k) > T$.

Let $p = A_{i_{\max}}[m_{i_{\max}}] = \max_{r_i - l_i > 1} A_i[m_i]$. Since

$$N_{i_{\max}}(p+1) = |\{x \in A_{i_{\max}} | x \le A_{i_{\max}}[m_{i_{\max}}]\}| \ge m_{i_{\max}}, \tag{11}$$

and for each $i \neq i_{\min}$,

$$N_i(p+1) \ge N_i(A_i[m_i] + 1) = |\{x \in A_i | x \le A_i[m_i]\}| \ge m_i, \tag{12}$$

it follows that

$$F(N_1(p+1), \ldots, N_k(p+1)) \ge F(m_1, \ldots, m_k) > T. \tag{13}$$

Thus $p^* < p+1$ which implies $p^* \le p$, then after updating $r_{i_{\max}} \leftarrow m_{i_{\max}}$, we still have $A_{i_{\max}}[l_{i_{\max}}] \le p^* \le A_{i_{\max}}[r_{i_{\max}}]$. For all $i \neq i_{\max}$, neither l_i nor r_i changes, so the invariant is preserved after the update.

Convergence and termination. In each iteration, an update is performed on sequence A_i only if $r_i - l_i > 1$, and every such update reduces the length of the interval by at least one. Consequently, the length of each interval $[l_i, r_i]$ shrinks to at most 2 after at most $O(\log |A_i|)$ updates on A_i. Since the number of sequences is a constant, the total number of iterations of the algorithm is bounded by $O(\log \sum_{i=1}^k |A_i|)$.

Now we show that when Algorithm 4 terminates, it returns p^* correctly. If it terminates at Line 4 or Line 13, which implies

$$F\big(N_1(A_i[l_i] + 1), \ldots, N_k(A_i[l_i] + 1)\big) > T, \tag{14}$$

it follows that $p^* \le A_i[l_i]$, then by the invariant $p^* \ge A_i[l_i]$, we have $p^* = A_i[l_i]$.

If Algorithm 4 terminates at line 15, which implies $r_i - l_i \le 1$ and $F(N_1(A_i[l_i] + 1), \ldots, N_k(A_i[l_i] + 1)) \le T$ satisfies for every $1 \le i \le k$. Let $p = \min_{1 \le i \le k} A_i[r_i]$ and $q = \max_{1 \le i \le k} A_i[l_i]$, then

$$\begin{aligned} F(N_1(p), \ldots, N_k(p)) &\le F(N_1(A_1[r_1]), \ldots, N_k(A_k[r_k])) \\ &\le F(N_1(A_1[l_1] + 1), \ldots, N_k(A_k[l_k] + 1)) \\ &\le F(N_1(q+1), \ldots, N_k(q+1)) \le T. \end{aligned} \tag{15}$$

Thus $p^* \ge p$, and by the invariant, $p^* \le p$, we conclude $p^* = p = \min_{1 \le i \le k} A_i[r_i]$.

In summary, since $i_{\min}$, $i_{\max}$, and F can be computed in $O(1)$ time, and computing each N_i takes at most $O(\log |A_i|)$ time, Algorithm 4 returns p^* in $O(\log \sum_{i=1}^k |A_i|)$ time. □

0.8 Proof of Lemma 9.

Proof. Let $\psi = [[l_1, h_1], \ldots, [l_n, h_n]]$, since $s = \min\{i | l_i \neq h_i\}$ is a split position of ψ, it follows that

(1) $\forall 1 \le i < s, l_i = h_i$,
(2) $\forall s < i \le n, l_i = \min_Q(x_i)$ and $h_i = \max_Q(x_i)$.

If ψ_{left} is not an empty range, then $l_p \le mid_p - 1$, which implies that ψ_{left} is a prefix range filter with split position s. Similarly, if ψ_{left} is not an empty range, then $mid_p + 1 \le h_p$, which implies that ψ_{right} is a prefix range filter with split position s. Finally, since $mid_p \le mid_p$, ψ_{mid} is a prefix range filter with split position s. □

0.9 Proof of Theorem 3.

Proof. We prove the theorem by analyzing Algorithm 2 under the setting where $N = upp(\psi_r)$ and $upp(\psi) = \lfloor \mathrm{AGM}_{c^*}(Q|_\psi) \rfloor$ for any range filter ψ. First, the indexes of the relation tables can be built in $O(|Q| \log |Q|)$ time. After the indexes have been built, the computation of c^* and ψ_r takes a constant time. Then, according to Lemma 2, $\varphi_Q^*(i) = \perp$ for each $i > upp(\psi_r)$. Since

$upp(\psi) = \lfloor \text{AGM}_{c^*}(Q|_\psi) \rfloor \leq \text{AGM}(Q)$, by Lemma 3, Algorithm 3 is a relaxed random-access algorithm running in $O(\log^2 |Q|)$ worst-case time and $O(\log |Q|)$ amortized time. Lastly, since $N = upp(\psi_r)$, by Lemma 1, the expected enumeration delay of Algorithm 2 is $O(\frac{upp(\psi_r)}{|Res(Q)|+1} \log^2 |Q|) = O(\frac{\text{AGM}(Q)}{|Res(Q)|+1} \log^2 |Q|)$, and its total running time is at most $O(\text{AGM}(Q) \log |Q|)$. □

0.10 Proof of Lemma 10.

Proof. Let $\psi = [[l_1, h_1], \ldots, [l_n, h_n]]$ be any prefix range filter with split position s. Let $I_1, \ldots, I_k$ be a partition of the interval $[l_s, h_s]$. For $\forall 1 \leq i \leq k$, let $\psi_i = [[l_1, h_1], \ldots, I_i, \ldots, [l_n, h_n]]$, then

$$\sum_{i=1}^{k} upp(\psi_i) \leq \sum_{i=1}^{k} upp_{i^*}(\psi_i) \leq upp_{i^*}^*(\psi) = upp^*(\psi), \tag{16}$$

where $i^* = \arg\min_{i=1}^{k} upp_i(\psi)$. □

0.11 Proof of Lemma 11.

Proof. If $\text{att}(Q_s) = \text{att}(Q)$, then $Res(Q|_\psi) \subseteq Res(Q_s|_\psi)$, which implies $|Res(Q|_\psi)| \leq |Res(Q_s|_\psi)|$. If $\text{att}(Q_s) \subsetneq \text{att}(Q)$, then

$$\begin{aligned} |Res(Q|_\psi)| &= |Res(Q_s|_\psi) \bowtie Res(Q_r|_\psi)| \\ &\leq \sum_{t \in Res(Q_s|_\psi)} |Res((Q_r \ltimes t)|_\psi)| \\ &\leq |Res(Q_s|_\psi)| \max_{t \in Res(Q_s|_\psi)} |Res((Q_r \ltimes t)|_\psi)| \\ &\leq |Res(Q_s|_\psi)| \cdot |Res(Q_r^*|_\psi)| \\ &\leq |Res(Q_s|_\psi)| \cdot \text{AGM}_c(Q_r^*|_\psi). \end{aligned} \tag{17}$$

□

0.12 Skewness of Datasets.

Our experimental study **includes datasets with skewed distributions**. To quantify the skewness of these datasets, we computed two widely used skewness metrics on their attributes (*i.e.*, on the value-frequency distributions of the columns).

(1) **Max/Mean Frequency Ratio.** Given an attribute A, let its distinct values be $v_1, v_2, \ldots, v_k$, and let f_i denote the number of occurrences of value v_i. Define

$$\text{MM} = \frac{\max_{i=1}^{k} f_i}{\frac{1}{k}\sum_{i=1}^{k} f_i}. \tag{18}$$

A larger Max/Mean ratio indicates stronger skew (one or only a few values dominate), whereas a ratio close to 1 suggests a near-uniform distribution.

(2) **Coefficient of Variation (CV).** Using the same frequencies $f_1, \ldots, f_k$, let $\mu = \frac{1}{k}\sum_{i=1}^{k} f_i$ and $\sigma = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(f_i - \mu)^2}$. Then

$$\text{CV} = \frac{\sigma}{\mu}. \tag{19}$$

CV measures the relative dispersion of the frequency distribution; a higher CV implies greater skewness.

Table 1 shows the skewness metrics (Max/Mean Frequency Ratio and Coefficient of Variation) computed for each dataset. For the Twitter dataset, the statistics are computed on a 2-million-edge subgraph uniformly sampled from the original graph, which was used for the Q_Δ experiments.

Table 1: Skewness metrics for key attributes in each dataset.

Dataset	Attribute	MM	CV
	follow.src/dst	5.38	0.85
TPC-DS+	web_sales.ws_bill_customer_sk	2.69	0.52
	web_sales.ws_item_sk	5.40	0.55
Twitter	follow.src/dst	11589.56	24.62

These skewness metrics demonstrate that our experimental study includes a highly skewed real-world dataset, *i.e.*, Twitter, as well as a dataset with mild skew, *i.e.*, TPC-DS+. This highlights that **our evaluation comprehensively covers a wide spectrum of data distributions, ensuring that the proposed methods are tested under both extreme and relatively uniform scenarios.**

0.13 Queries in the Experiments.

Query 1: Q_A on TPC-DS+

```
SELECT *
FROM follow f, web_sales w1, web_sales w2, web_sales w3,
     web_sales w4
WHERE f.src = w1.ws_bill_customer_sk
     AND f.src = w2.ws_bill_customer_sk
     AND f.dst = w3.ws_bill_customer_sk
     AND f.dst = w4.ws_bill_customer_sk
     AND w1.ws_item_sk = w3.ws_item_sk
     AND w2.ws_item_sk = w4.ws_item_sk
```

Query 2: Q_Δ on Twitter

```
SELECT *
FROM follow A, follow B, follow C
WHERE A.dst = B.src AND B.dst = C.src AND C.dst = A.src
```

Query 3: Q_S on Twitter

```
SELECT *
FROM follow A, follow B, follow C, follow D, follow E,
     follow F
WHERE A.dst = B.src AND B.dst = C.src
     AND C.dst = D.src AND D.dst = A.src
     AND E.src = A.src AND E.dst = B.dst
     AND F.dst = A.dst AND F.src = D.src
```

REFERENCES

[1] Eugene F. Schuster and William R. Sype and. 1987. On the negative hypergeometric distribution. *International Journal of Mathematical Education in Science and Technology* 18, 3 (1987), 453–459. https://doi.org/10.1080/0020739870180316 arXiv:https://doi.org/10.1080/0020739870180316