Worst Case Optimal Joins

CompSci 590.04
Instructor: Ashwin Machanavajjhala
Multi-way Joins

\[ J(a,b,c) :- R(a,b) \ S(b,c) \ T(a,c) \]

• Historically databases designers decided that the best way to handle multi-way joins is to do them one pair at a time.
  – For efficiency reasons.
How fast is this approach?

\[ R = \{a_0\} \times \{b_0, \ldots, b_m\} \cup \{a_0, \ldots, a_m\} \times \{b_0\} \]

\[ S = \{b_0\} \times \{c_0, \ldots, c_m\} \cup \{b_0, \ldots, b_m\} \times \{c_0\} \]

\[ T = \{a_0\} \times \{c_0, \ldots, c_m\} \cup \{a_0, \ldots, a_m\} \times \{c_0\} \]
How fast is this approach?

\[
R = \{a_0\} \times \{b_0, \ldots, b_m\} \cup \{a_0, \ldots, a_m\} \times \{b_0\}
\]

\[
S = \{b_0\} \times \{c_0, \ldots, c_m\} \cup \{b_0, \ldots, b_m\} \times \{c_0\}
\]

\[
T = \{a_0\} \times \{c_0, \ldots, c_m\} \cup \{a_0, \ldots, a_m\} \times \{c_0\}
\]

- Each instance has \(2m+1\) rows.
- \(J(a, b, c)\) has \(3m+1\) rows
- Any pairwise join (e.g., \(J1(a,b,c) = R(a,b), S(b,c)\)) has size \(m^2 + m\)
What does this mean for triangle counting?

- Every database system necessarily takes $O(N^2)$
  - *Ignoring log terms*

- Find all pairs (b,c) are connected with a
- Check if (b,c) is an edge.

- Is this the best we can do?
We can do better!

- ... not only for triangle counting, but it seems database systems have been doing multi-way joins suboptimally for 40 years!!!

- Triangle counting can be solved in $O(N^{1.5})$, and so can any join of the form $R(a,b) S(b,c) T(a,c)$. 
How?

- Is there an O(N) algorithm for the following join problem:

\[
R = \{a_0\} \times \{b_0, \ldots, b_m\} \cup \{a_0, \ldots, a_m\} \times \{b_0\}
\]

\[
S = \{b_0\} \times \{c_0, \ldots, c_m\} \cup \{b_0, \ldots, b_m\} \times \{c_0\}
\]

\[
T = \{a_0\} \times \{c_0, \ldots, c_m\} \cup \{a_0, \ldots, a_m\} \times \{c_0\}
\]
Power of Two Choices: Heavy vs Light

• Consider attribute A

• For all ai not equal to a0, there is exactly one tuple in R (ai, b0) and one tuple in T (ai, c0)

  Compute \( \sigma_{A=a_i}(R) \bowtie \sigma_{A=a_i}(T) \) and filter the results by probing against S

• The above strategy is bad for a0
  – Joining tables R and T on a0 results in an intermediate of \( N^2 \).
Power of Two Choices: Heavy vs Light

- Consider attribute A

- For all ai not equal to a0, and one tuple in T (ai, c0)

  Compute $\sigma_{A=a_i}(R) \bowtie \sigma_{A=a_i}(T)$ and filter the results by probing against S

- For ai = a0:

  Consider each tuple in $(b, c) \in S$ and check if $(a_i, b) \in R$ and $(a_i, c) \in T$.

There are O(N) values ai, each resulting in a single join record (ai, b0, c0). Checking whether (b0, c0) is in S is O(1) ... assuming an index

There are N rows in S. Again, checking (ai, b) is in R and (ai, c) is in T takes O(1) ... assuming an index

Lecture 19 : 590.04 Fall 15
Power of Two Choices: Heavy vs Light

- Consider attribute A
- For all ai not equal to a0, and one tuple in T (ai, c0)
  
  Compute $\sigma_{A=a_i}(R) \bowtie \sigma_{A=a_i}(T)$ and filter the results by probing against S

- For ai = a0:
  
  Consider each tuple in $(b, c) \in S$ and check if $(a_i, b) \in R$ and $(a_i, c) \in T$.

Such ai’s are called light nodes. Traditional join processing works here.

Such ai’s are called heavy nodes. Need to compute the join jointly.

Lecture 19 : 590.04 Fall 15
Power of Two Choices Algorithm

**Algorithm 1** Computing $Q_\Delta$ with power of two choices.

**Input:** $R(A, B), S(B, C), T(A, C)$ in sorted order

1. $Q_\Delta \leftarrow \emptyset$
2. $L \leftarrow \pi_A (R) \cap \pi_A (T)$
3. **For** each $a \in L$ **do**
   4. **If** $|\sigma_{A=a} R| \cdot |\sigma_{A=a} T| \geq |S|$ **then**
      5. **For** each $(b, c) \in S$ **do**
          6. **If** $(a, b) \in R$ and $(a, c) \in T$ **then**
             7. Add $(a, b, c)$ to $Q_\Delta$
      8. **else**
         9. **For** each $b \in \pi_B (\sigma_{A=a} R) \wedge c \in \pi_C (\sigma_{A=a} T)$ **do**
         10. **If** $(b, c) \in S$ **then**
             11. Add $(a, b, c)$ to $Q_\Delta$
4. **Return** $Q$

---

Lecture 19 : 590.04 Fall 15
Runtime Analysis

- Computing L takes:

\[ \min (|\sigma_{A=a}R| \cdot |\sigma_{A=a}T|, |S|) \]

- Rest of the algorithm takes:

\[
\sum_{a \in L} \min (|\sigma_{A=a}R| \cdot |\sigma_{A=a}T|, |S|) \leq \sqrt{|S|} \cdot \sqrt{|R|} \cdot \sqrt{|T|}
\]
Can we do better?

• NO!

• A matching lower bound by Atserias Grohe and Marx (or the AGM bound)