CSE 544
Principles of Database Management Systems

Fall 2016

Lecture 11 – Worst Case Optimal Algorithm

Midterm

- This Thursday

- Closed books, no computers
- Bring a pen

- Material up to today

Yannakakis Algorithm

- Given: acyclic query Q
- Compute Q on any database in time O(|Input|+|Output|)

- Step 1: semi-join reduction
  - Pick any root node x in the tree decomposition of Q
  - Do a semi-join reduction sweep ~~from x to the leaves~~
  - Do a semi-join reduction sweep ~~from the leaves to x~~
- Step 2: compute the joins bottom up, with early projections

from the leaves to x

from x to the leaves

CORRECTION

Worst-Case Optimal Algorithm

- Given database statistics |R1|=N1, |R2|=N2, ...
- The query's output is bound by $|Q| \leq AGM(Q)$

- An algorithm for computing Q is called <u>*worst-case optimal*</u> if its runtime is O(AGM(Q))

- (Why do we call it optimal?)

Query Plans Are Not Worst-Case Optimal

$Q(x,y,z) = R(x,y), S(y,z), T(z,x)$,

$AGM(Q) = N^{3/2}$

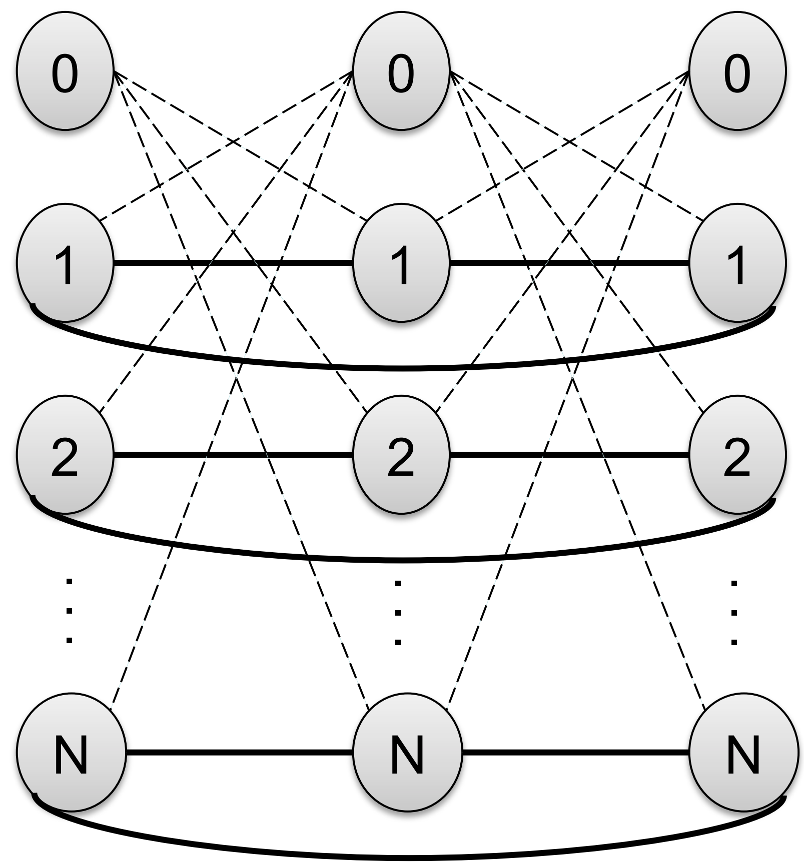
Three plans:

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

What is their runtime?

$R = S = T =$

$= \{(i,i) \mid i = 1,N\}$
$\cup \{(0,i) \mid i = 1,N\}$
$\cup \{(i,0) \mid I =1,N\}$



Generic-Join

$Q(x_1,\ldots, x_k) = R_1(\ldots), R_2(\ldots), \ldots, R_m(\ldots)$; $N = \max_j |R_j|$

Recursive algorithm on the structure of the query Q:

- If Q has no variables, then output current bindings of vars
- If Q has some variable x:
  - Let $J_1 = \{j \mid R_j \text{ contains } x\}$, $J_0 = \{j \mid Rj \text{ does not contain } x\}$
- Compute $A = \cap \{ \Pi_x(R_j) \mid j \in J_1 \}$
- For each $a \in A$, compute $Q[a/x]$

needs to be done in time $\tilde{O}(\min_j \Pi_x(R_j))$

Theorem: the runtime is $\tilde{O}(AGM(Q))$

Recall:
$\tilde{O}(F)= O(F \log N)$

Example

$Q(x,y,z) = R(x,y), S(y,z), T(z,x),$ $\quad AGM(Q) = N^{3/2}$

$A = \Pi_x(R(x,y)) \cap \Pi_x(T(z,x))$

for a in A **do**

/* compute Q(a,y,z) = R(a,y),S(y,z),T(z,a) */

$B = \Pi_y(R(a,y)) \cap \Pi_y(S(y,z))$

for b in B **do**

/* compute Q(a,b,z) = R(a,b),S(b,z),T(z,a) */

$C = \Pi_z(S(b,z)) \cap \Pi_z(T(z,a))$

for c in C **do**

output (a,b,c)

Proof of the Runtime

- Cauchy-Schwartz:
- $(\sum_i a_i)^{½} (\sum_i b_i)^{½} \geq (\sum_i a_i^{½}) (\sum_i b_i^{½})$
- Generalized Holder: if w1 + w2 + w3 ≥ 1 then

  $(\sum_i a_i)^{w1} (\sum_i b_i)^{w2} (\sum_i c_i)^{w3} \geq (\sum_i a_i^{w1}) (\sum_i b_i^{w2}) (\sum_i b_i^{w3})$

Proof of the Runtime

$Q(x_1,\dots, x_k) = R_1(\dots), R_2(\dots), \dots, R_m(\dots);$

Fix any fractional edge cover $w_1, \dots, w_m$

CLAIM: Runtime = $\tilde{O}(\prod_j N_j^{wj})$

Proof: $J_1 = \{j \mid R_j \text{ contains } x\}$, $J_0 = \{j \mid Rj \text{ does not contain } x\}$

- For $j \in J_1$, $a \in A$, let $N_{j,a} = |R_j[a/x]|$
- By induction: time for $Q[a/x]$ is $(\prod_{j\in J0} N_j^{wj}) (\prod_{j\in J1} N_{j,a}^{wj})$
- Runtime for Q is:

$$(\prod_{j\in J0} N_j^{wj}) \sum_a(\prod_{j\in J1} N_{j,a}^{wj}) \leq (\prod_{j\in J0} N_j^{wj}) (\prod_{j\in J1} (\sum_a N_{j,a})^{wj})$$

$$\leq (\prod_{j\in J0} N_j^{wj}) (\prod_{j\in J1} N_j^{wj})$$

$$= \prod_j N_j^{wj}$$

Holder. Because $\sum_{j\in J1} w_j \geq 1$ (why?)

Because $\sum_a N_{j,a} \leq N_j$ (why)

Comments

- Show in class: computing $A = \cap \{ \Pi_x(R_j) \mid j \in J_1 \}$ takes time $\tilde{O}(AGM(Q))$
  Note: we must compute A in time $\tilde{O}(\min_j \Pi_x(R_j))$
  (why? if N_j is huge and $w_j=0$ then $N_j > AGM(Q)$ and we cannot afford to iterate over $\Pi_x(R_j)$)

- Generic Join is worst case optimal for *any* variable order
- Should we ignore the variable order?

Comparison to Naïve Nested Loop

Naïve nested loop:

```
For x in Domain do
  For y in Domain do
    For z in Domain do
      ...
```

Generic-join

```
A = ∩ domains for x
For x in A do
   B = ∩ domains for y
   For y in B do
      C = ∩ domains for z
      For z in C do
         ...
```

Example

- $Q(x,y,z) = R(x,z), S(y,z)$ AGM(Q) = N^2
- Suppose we pick the worst variable order: x,y,z
- What is the runtime of
  - Naïve nested loop?
  - Generic join?
- On these three databases:
  - R=S= {(i,i) | i=1,N}
  - R={(0,i) | i=1,N}, S = {(i,0) | i=1,N}
  - R={(i,0) | i=1,N}, S = {(0,i) | i=1,N}

Extensions to Keys

- The AGM bound is worst case for *any* input database
- Consider the case when the input database has keys
- E.g. what is the largest output of these queries?
  - $R(x,y),S(\underline{y},z)$
  - $R(\underline{x},y),S(y,z),T(z,x)$

Query Expansion

- Given a query Q, repeat the following:
- If $x \rightarrow y$ holds in some atom R,
  then add the variable y to all atoms S that contain x
- Call Q' the *expanded* query
- FACT1: assuming the inputs to Q satisfy a set of FD's, then $|Q| \leq AGM(Q')$
- FACT2: if all FD's are simple keys, then this bound is tight

Examples

- $Q = R(x,y),S(\underline{y},z)$
- $Q' = R(x,y,z),S(y,z)$; $AGM(Q') = N_1$

- $Q = R(\underline{x},y),S(y,z),T(z,x)$
- $Q' = R(x,y),S(y,z),T(z,x,y)$; $AGM(Q') = \min(N_1N_2, N_3)$