Query Optimization – Homework 1

January, 2026

Submit your answer in a pdf file on Canvas.

- Write your name in the file.
- Use this template `https://www.overleaf.com/project/67f01a081d8c577a12f22353`

Grading is done using credit/partial-credit/no-credit; ignore the points below.
An asterix * indicates that the question may be more challenging.

1 Query Optimization using Dynamic Programming

1. (0 points)

   Consider the DPccp algorithm, which computes an optimal bushy plan by iterating over all connected component pairs (S_1, S_2).

   (a) How large is the size of the dynamic programming table (denoted PlanCost in the lecture notes) in each of the cases below? It suffices to give an answer using the big-O notation, e.g. $O(n^4)$ (not a real answer).

   i. The algorithm runs on a chain query with n relations:
   $Q = R_1(X_0, X_1) \bowtie R_2(X_1, X_2) \bowtie \ldots \bowtie R_n(X_{n-1}, X_n)$

   ii. The algorithm runs on a star query with n relations:
   $Q = R_1(x_2, x_3, \ldots, x_n) \bowtie R_2(x_2, y_2) \bowtie R_3(x_3, y_3) \bowtie \cdots \bowtie R_n(x_n, y_n)$

   (b) The runtime of the algorithm is given by the number of connected component pairs (S_1, S_2) considered. What is the runtime in each of the cases below?

   i. The algorithm runs on a chain query with n relations.

   ii. The algorithm runs on a star query with n relations.

2 Non-Reordable Operators

2. (0 points)

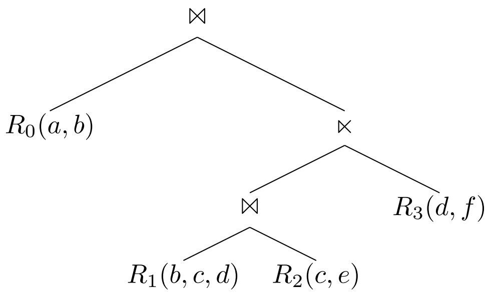
   (a) Prove that $E_1 \equiv E_2$, where

$$\begin{aligned} E_1 =& (R(A, B) ⟕ S(B, C)) ⟕ T(C, D) \\ E_2 =& R(A, B) ⟕ (S(B, C) ⟕ T(C, D)) \end{aligned}$$

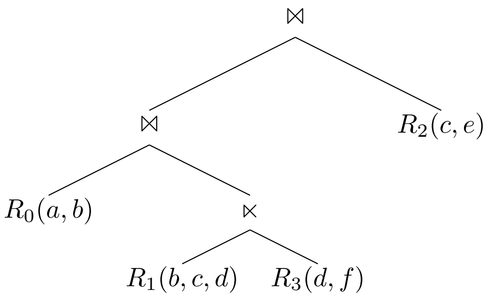
   You may either use identities, or a direct argument, for example by using linearity.

   (b) Consider the operator trees E_1, E_2 below. Prove that $E_1 \equiv E_2$, by rewriting E_1 to E_2 using the identities *commutativity*, *associativity*, *l-asscom*, and *r-asscom*: in class we have denoted this by $E_1 \simeq E_2$. You can only use the identities listed in the lecture notes, which can also be found here [2].

$E_1 =$



$E_2 =$



3 Specialized Algorithms

3. (0 points)

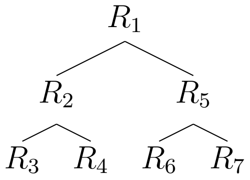
   (a) Consider the cycle query $Q(x, y, z, u) = R(x, y) \bowtie S(y, z) \bowtie T(z, u) \bowtie K(u, x)$ and the following statistics on the database:

$$|R| = 100 \qquad |S| = 300 \qquad |T| = 200 \qquad |K| = 400$$

   Use the Greedy algorithm to compute a bushy query plan without cross products. Assume that, for any join operation $P_1 \bowtie P_2$ the estimated size and cost are:

$$\texttt{Est}(P_1 \bowtie P_2) = 0.02 \cdot \texttt{Est}(P_1) \cdot \texttt{Est}(P_2) \qquad \text{Cost}(P_1 \bowtie P_2) = \texttt{Est}(P_1) \cdot \texttt{Est}(P_2)$$

(b) Consider the following query tree:



Assume the following statistics on the database:

	$\lvert R_i \rvert$	θ_i	$\text{Cost}(R_i) = g_i(\lvert R_i \rvert)$
R_1	700	1	10
R_2	600	0.01	200
R_3	500	0.01	20
R_4	400	0.01	10
R_5	300	0.01	10
R_6	200	0.01	4
R_7	100	0.01	10

Using the IKKBZ algorithm, compute the optimal left linear plan that starts with R_1. (In other words you only need to consider R_1 to be the root of the query tree and find the corresponding optimal plan. No need to try roots $R_2, R_3, \ldots$)

The lecture notes should be sufficient to understand the IKKBZ algorithm. If you want to see more details, refer to [1]; a high-level overview of the algorithm can also be found here [3].

(c) Consider the Cost and Estimation function of the IKKBZ algorithm. If S_1, S_2 are two sequence, then prove the following:

i. If $\text{Rank}(S_1) \leq \text{Rank}(S_2)$ then $\text{Rank}(S_1) \leq \text{Rank}(S_1 S_2) \leq \text{Rank}(S_2)$

ii. If $\text{Rank}(S_1) \geq \text{Rank}(S_2)$ then $\text{Rank}(S_1) \geq \text{Rank}(S_1 S_2) \geq \text{Rank}(S_2)$

In other words, the rank of the concatenated sequence $S_1 S_2$ is between the ranks of S_1 and S_2. Hint: do a direct calculation, using the definition of the Rank function, and the recursive definitions of `Est` and Cost. If you prove the first statement elegantly, then the second statement follows immediately by replacing $\leq$ with $\geq$ and you won't need to prove it again.

References

[1] R. Krishnamurthy, H. Boral, and C. Zaniolo. Optimization of nonrecursive queries. In W. W. Chu, G. Gardarin, S. Ohsuga, and Y. Kambayashi, editors, *VLDB'86 Twelfth International Conference on Very Large Data Bases, August 25-28, 1986, Kyoto, Japan, Proceedings*, pages 128–137. Morgan Kaufmann, 1986.

[2] G. Moerkotte, P. Fender, and M. Eich. On the correct and complete enumeration of the core search space. In K. A. Ross, D. Srivastava, and D. Papadias, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, pages 493–504. ACM, 2013.

[3] T. Neumann and B. Radke. Adaptive optimization of very large join queries. In G. Das, C. M. Jermaine, and P. A. Bernstein, editors, *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, pages 677–692. ACM, 2018.