Solutions for Exercise Sheet 5

Richter, Yannick
MTK 03741982
ge78tup@mytum.de

Rodrigues, Diogo
MTK 03770446
diogo.rodrigues@tum.de

TUM – Query Optimization 2022/23
25th November 2022

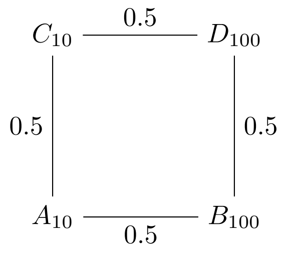
Our solutions for Exercise Sheet 5.

Exercise 1

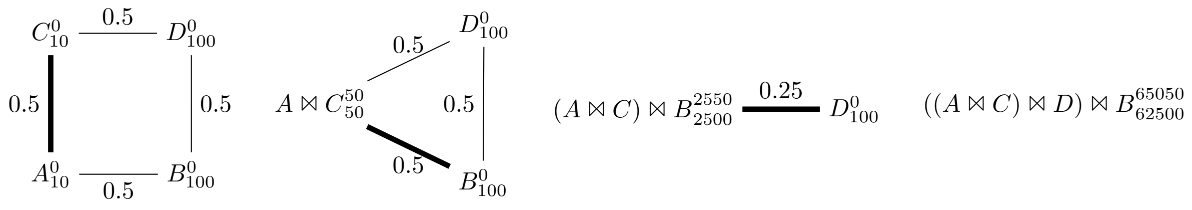
1. Give an example query graph with cardinalities and selectivities for which the GreedyOperatorOrdering (GOO) algorithm does not give the optimal (with regards to C_{out}) join tree. Specify the optimal join tree.
2. Construct the join trees for the example from 1 using GreedyJoinOrdering-1, GreedyJoinOrdering-2 and GreedyJoinOrdering-3. Give the C_{out} cost for the results and compare them with the GOO tree's cost;
   - For GJO-1 use the cardinalities as cost function.
   - For GJO-2 and GJO-3 use the selectivities as a cost function. (For the first relation you can still use the cardinality)

Item 1

Consider a query involving relations A, B, C and D. The cardinality of each relation is subscripted, edges indicate joins between two relations and edge weights indicate the selectivities of each join. To break ties, we prefer the lexicographically smaller relation.

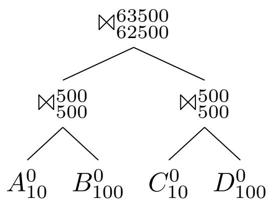


Superscript is the value of C_{out} of the subquery. The trace for GOO is as follows:



So the value of C_{out} for the solution given by GOO is $65\,050$.

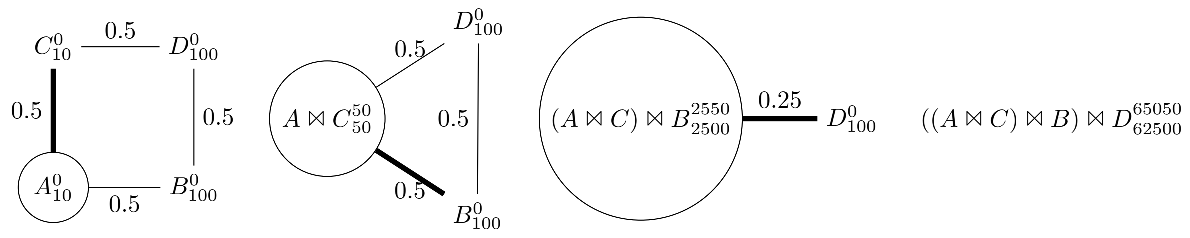
Now consider the following optimal join tree:



The optimal solution has cost $63\,500$, which is less than the value $65\,050$ given by GOO.

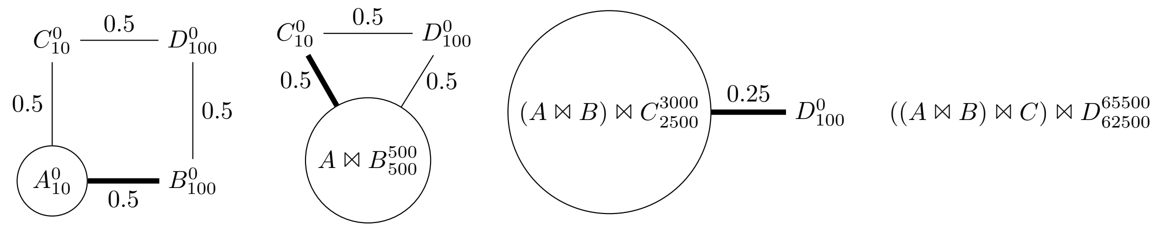
Item 2

GJO-1 works by building a single tree, and at each step it joins the tree with the relation that has the least cost (or cardinality, in this case); we break ties by considering the lexicographically smallest relation. Here is the trace for GJO-1:


C^0_{10}
D^0_{100}
A^0_{10}
B^0_{100}
0.5
$A \bowtie C^{50}_{50}$
$(A \bowtie C) \bowtie B^{2550}_{2500}$
0.25
$((A \bowtie C) \bowtie B) \bowtie D^{65050}_{62500}$

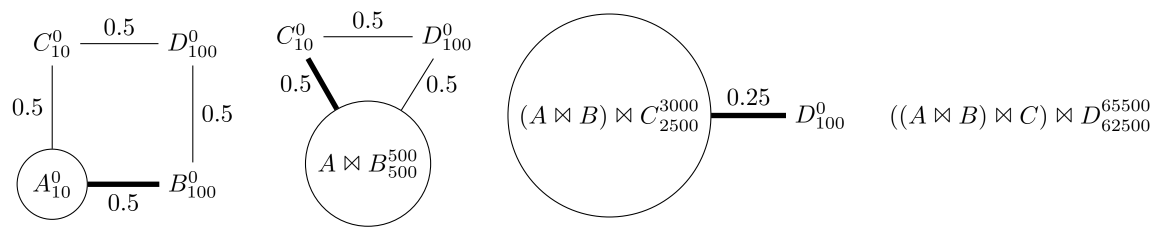
Thus the cost of GJO-1 solution is $65\,050$.

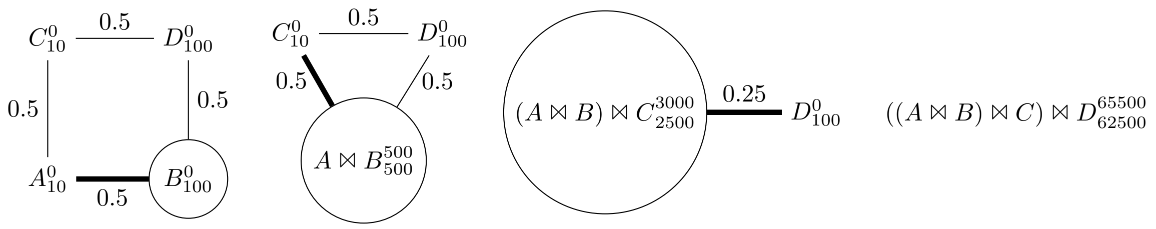
GJO-2 works in the same way as GJO-1, except it picks the next relation based on the least cost function when associated to that specific tree. In this case, our cost function is the selectivity between the tree and the relation. Because all selectivities from our example are the same, GJO-2 could produce any arbitrary tree, but we will break ties by lexicographically smallest relations, so we start with $A - B$.

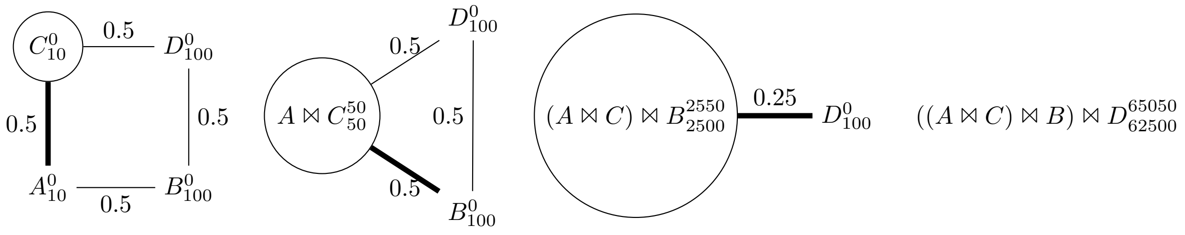

C^0_{10}
D^0_{100}
A^0_{10}
B^0_{100}
0.5
$A \bowtie B^{500}_{500}$
$(A \bowtie B) \bowtie C^{3000}_{2500}$
0.25
$((A \bowtie B) \bowtie C) \bowtie D^{65500}_{62500}$

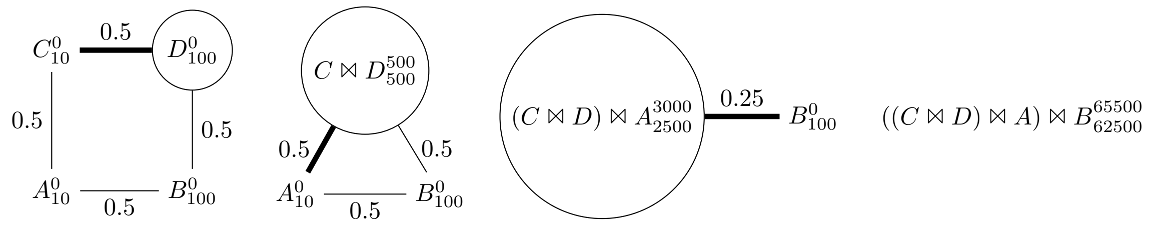
Thus the cost of GJO-2 solution is $65\,500$.

GJO-3 is the same as GJO-2, except it runs GJO-2 for all possible values of the starting relation. Here is the trace for GJO-3, each line corresponding to a different initial relation:


C^0_{10}
D^0_{100}
A^0_{10}
B^0_{100}
0.5
$A \bowtie B^{500}_{500}$
$(A \bowtie B) \bowtie C^{3000}_{2500}$
0.25
$((A \bowtie B) \bowtie C) \bowtie D^{65500}_{62500}$


C^0_{10}
0.5
D^0_{100}
0.5
0.5
A^0_{10}
B^0_{100}
0.5
C^0_{10}
0.5
D^0_{100}
0.5
0.5
$A \bowtie B^{500}_{500}$
$(A \bowtie B) \bowtie C^{3000}_{2500}$
0.25
D^0_{100}
$((A \bowtie B) \bowtie C) \bowtie D^{65500}_{62500}$




C^0_{10}
0.5
D^0_{100}
0.5
0.5
A^0_{10}
0.5
B^0_{100}
$C \bowtie D^{500}_{500}$
0.5
0.5
A^0_{10}
0.5
B^0_{100}
$(C \bowtie D) \bowtie A^{3000}_{2500}$
0.25
B^0_{100}
$((C \bowtie D) \bowtie A) \bowtie B^{65500}_{62500}$

Therefore, the solution given by GJO-3 is $((A \bowtie C) \bowtie B) \bowtie D$, with a cost of $65\,050$.

Strategy	**Cost** (C_{out})
GJO-1	$65\,050$
GJO-2	$65\,500$
GJO-3	$65\,050$
GOO	$65\,050$
Optimal	$63\,500$

The solution of GJO-3 is always better than or equal to the solution of GJO-2, since GJO-3 is defined as GJO-2 but starting at every possible relation.

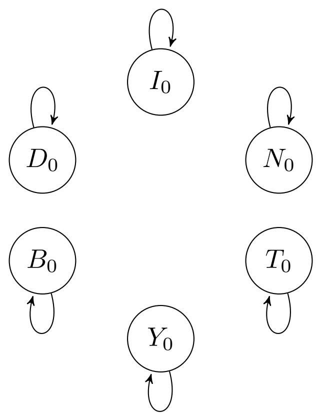
In this case, the solution of GOO is better than GJO-2 and GJO-3, and equal to GJO-1.

Exercise 2

Create a union find data-structure with the letters $\{B, D, I, N, T, Y\}$ as elements. Perform the following operations and give the resulting graph after each step.

1. `union(D,B)`
2. `union(T,I)`
3. `union(I,N)`
4. `union(N,Y)`

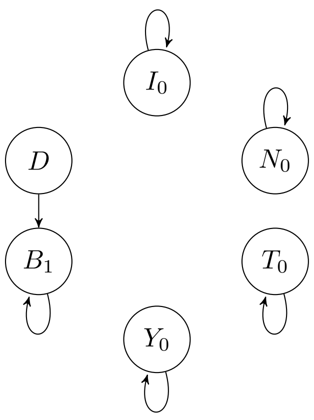
The initial graph should look like this:



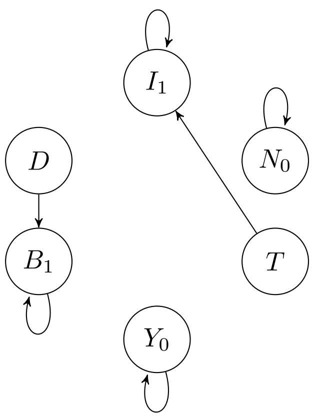
Hint: Do not forget to apply path compression in each step.

---

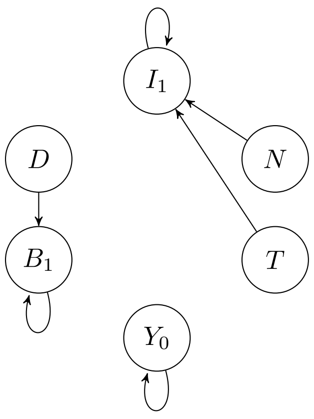
(1) `union(D, B)`



(2) `union(T, I)`



(3) `union(I, N)`



(4) `union(N, Y)`