Solutions for Exercise Sheet 7

Richter, Yannick
MTK 03741982
ge78tup@mytum.de

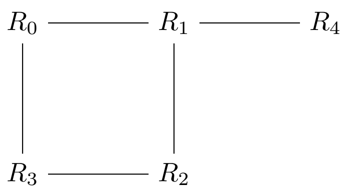
Rodrigues, Diogo
MTK 03770446
diogo.rodrigues@tum.de

TUM – Query Optimization 2022/23
9th December 2022

Our solutions for Exercise Sheet 7.

Exercise 1

Given the following query graph



1. enumerate all pairs of join partners as considered by DPsize without crossproducts
2. enumerate all pairs of join partners as considered by DPsub with crossproducts

---

1. We will use a shorter notation, where for instance 014 means $\{R_0, R_1, R_4\}$. The subsets of each size are:
   - $|S| = 1$: $0, 1, 2, 3, 4$
   - $|S| = 2$: $01, 02, 03, 04, 12, 13, 14, 23, 24, 34$
   - $|S| = 3$: $012, 013, 014, 023, 024, 034, 123, 124, 134, 234$
   - $|S| = 4$: $0123, 0124, 0134, 0234, 1234$
   - $|S| = 5$: 01234

   However, because we only consider trees without crossproducts, the considered subsets are:
   - $|S| = 1$: $0, 1, 2, 3, 4$
   - $|S| = 2$: $01, 03, 12, 14, 23$
   - $|S| = 3$: $012, 013, 014, 023, 123, 124$
   - $|S| = 4$: $0123, 0124, 0134, 1234$
   - $|S| = 5$: 01234

   We consider only one of the join possibilities, so we do not do $S_1 \cup S_2$ and $S_2 \cup S_1$; we consider only the pair where the left operand is shorter, and if they have the same length we consider the pair where the left side is lexicographically smaller.

   The considered joins are as follows, in this order:
   - $s = 2$:

- $S_1 = 0,\ S_2 = 1, 2$
- $S_1 = 1,\ S_2 = 2, 4$
- $S_1 = 2,\ S_2 = 3$

▪ $s = 3$:

- $S_1 = 0,\ S_2 = 12, 14, 23$
- $S_1 = 1,\ S_2 = 03, 23$
- $S_1 = 2,\ S_2 = 01, 03, 14$
- $S_1 = 3,\ S_2 = 01, 12$
- $S_1 = 4,\ S_2 = 01, 12$

▪ $s = 4$:

- $S_1 = 0,\ S_2 = 123, 124$
- $S_1 = 1,\ S_2 = 023$
- $S_1 = 2,\ S_2 = 013, 014$
- $S_1 = 3,\ S_2 = 012, 014, 124$
- $S_1 = 4,\ S_2 = 012, 013, 123$
- $S_1 = 01,\ S_2 = 23$
- $S_1 = 03,\ S_2 = 12, 14$
- $S_1 = 14,\ S_2 = 23$

▪ $s = 5$:

- $S_1 = 0,\ S_2 = 1234$
- $S_1 = 2,\ S_2 = 0134$
- $S_1 = 3,\ S_2 = 0124$
- $S_1 = 4,\ S_2 = 0123$
- $S_1 = 03,\ S_2 = 124$
- $S_1 = 14,\ S_2 = 023$
- $S_1 = 23,\ S_2 = 014$

2.

N	N
1	R_0
10	R_1
11	$R_1 \bowtie R_0$
100	R_2
101	$R_2 \bowtie R_0$
110	$R_2 \bowtie R_1$
111	$R_2 \bowtie R_1 \bowtie R_0$
1000	R_3
1001	$R_3 \bowtie R_0$
1010	$R_3 \bowtie R_1$
1011	$R_3 \bowtie R_1 \bowtie R_0$
1100	$R_3 \bowtie R_2$
1101	$R_3 \bowtie R_2 \bowtie R_0$
1110	$R_3 \bowtie R_2 \bowtie R_1$
1111	$R_3 \bowtie R_2 \bowtie R_1 \bowtie R_0$
10000	R_4
10001	$R_4 \bowtie R_0$
10010	$R_4 \bowtie R_1$
10011	$R_4 \bowtie R_1 \bowtie R_0$
10100	$R_4 \bowtie R_2$
10101	$R_4 \bowtie R_2 \bowtie R_0$
10111	$R_4 \bowtie R_2 \bowtie R_1 \bowtie R_0$
11000	$R_4 \bowtie R_3$
11001	$R_4 \bowtie R_3 \bowtie R_0$
11010	$R_4 \bowtie R_3 \bowtie R_1$
11011	$R_4 \bowtie R_3 \bowtie R_1 \bowtie R_0$
11100	$R_4 \bowtie R_3 \bowtie R_2$
11101	$R_3 \bowtie R_3 \bowtie R_2 \bowtie R_0$
11110	$R_4 \bowtie R_3 \bowtie R_2 \bowtie R_1$
11111	$R_4 \bowtie R_3 \bowtie R_2 \bowtie R_1 \bowtie R_0$

N	N
00001	00001
00010	00010
00011	$00001 \bowtie 00010$
00100	00100
00101	$00001 \bowtie 00100$
00110	$00010 \bowtie 00100$
00111	$00001 \bowtie 00110 \vee 00010 \bowtie 00101 \vee 00011 \bowtie 00100$
01000	01000
01001	$00001 \bowtie 01000$
01010	$00010 \bowtie 01000$
01011	$00001 \bowtie 01010 \vee 00010 \bowtie 01001 \vee 00011 \bowtie 01000$
01100	$00100 \bowtie 01000$
01101	$00001 \bowtie 01100 \vee 00100 \bowtie 01001 \vee 00101 \bowtie 01000$
01110	$00010 \bowtie 01100 \vee 00100 \bowtie 01010 \vee 00110 \bowtie 01000$
01111	$00001 \bowtie 01110 \vee 00010 \bowtie 01101 \vee 00011 \bowtie 01100 \vee 00100 \bowtie 01011$
	$00101 \bowtie 01010 \vee 00110 \bowtie 01001 \vee 00111 \bowtie 01000$
10000	10000
10001	$00001 \bowtie 10000$
10010	$00010 \bowtie 10000$
10011	$00001 \bowtie 10010 \vee 00010 \bowtie 10001 \vee 00011 \bowtie 10000$
10100	$00100 \bowtie 10000$
10101	$00001 \bowtie 10100 \vee 00100 \bowtie 10001 \vee 00101 \bowtie 10000$
10110	$00010 \bowtie 10100 \vee 00100 \bowtie 10010 \vee 00110 \bowtie 10000$
10111	$00001 \bowtie 10110 \vee 00010 \bowtie 10101 \vee 00011 \bowtie 10100 \vee 00100 \bowtie 10011$
	$00101 \bowtie 10010 \vee 00110 \bowtie 10001 \vee 00111 \bowtie 10000$
11000	$01000 \bowtie 10000$
11001	$00001 \bowtie 11000 \vee 01000 \bowtie 10001 \vee 01001 \bowtie 10000$
11010	$00010 \bowtie 11000 \vee 01000 \bowtie 10010 \vee 01010 \bowtie 10000$
11011	$00001 \bowtie 11010 \vee 00010 \bowtie 11001 \vee 00011 \bowtie 11000 \vee 01000 \bowtie 10011$
	$01001 \bowtie 10010 \vee 01010 \bowtie 10001 \vee 01011 \bowtie 10000$
11100	$00100 \bowtie 11000 \vee 01000 \bowtie 10100 \vee 01100 \bowtie 10000$
11101	$00001 \bowtie 11100 \vee 00100 \bowtie 11001 \vee 00101 \bowtie 11000 \vee 01000 \bowtie 10101$
	$01001 \bowtie 10100 \vee 01100 \bowtie 10001 \vee 01101 \bowtie 10000$
11110	$00010 \bowtie 11100 \vee 00100 \bowtie 11010 \vee 00110 \bowtie 11000 \vee 01000 \bowtie 10110$
	$01010 \bowtie 10100 \vee 01100 \bowtie 10010 \vee 01110 \bowtie 10000$
11111	$00001 \bowtie 11110 \vee 00010 \bowtie 11101 \vee 00011 \bowtie 11100 \vee 00100 \bowtie 11011$
	$00101 \bowtie 11010 \vee 00110 \bowtie 11001 \vee 00111 \bowtie 11000 \vee 01000 \bowtie 10111$
	$01001 \bowtie 10110 \vee 01010 \bowtie 10101 \vee 01011 \bowtie 10100 \vee 01100 \bowtie 10011$
	$01101 \bowtie 10010 \vee 01110 \bowtie 10001 \vee 01111 \bowtie 10000$

Exercise 2

Star queries are common in real life. In business workloads, you often need to join a big fact table with a large number of dimension tables to extract data from a normalized data set. You are given a star query with central relation R_0 and relations $R_1, ..., R_n$ directly connected to R_0 with selectivities $s_1, ..., s_n$. The following is a greedy algorithm for building join trees for star queries:

GreedyStar(G)
Input: a star query graph G with relations $R_0, R_1, ..., R_n$
Output: a join tree T
  $T := R_0$
  for $R_i \in \{R_1, ..., R_n\}$ ordered by $|R_i| \cdot s_i$ ascending **do**
    $T := T \bowtie R_i$
  return T

Is this greedy algorithm guaranteed to find the optimal join tree according to C_{out}? Prove it correct or find a counter example. Compare the runtime complexity of this algorithm with DP algorithms.

We will start by stating that, if we do not allow cross-products, then GreedyStar is optimal, and we proceed to prove it. If we allow cross-products, we present a trivial counter-example that shows GreedyStar is not optimal.

We will assume throughout that we are using a symmetric cost function (specifically C_{out}).

Theorem 2.1. *The only possible crossproduct-free join trees for a star are linear trees.*

Proof. All joins involve one leaf and R_0. Therefore, the first join has to be between one leaf and R_0, so R_0 is at the bottom of the tree. After that first step, we still have to join each remaining leaf R_i with the current join tree using join $R_0 - R_i$, so for each following join we are joining a base relation with the tree we have so far. Therefore, we can only have linear trees. ■

We will now consider our *cost function* for GreedyStar to be $c(R_i) = n_i \cdot s_i$, to simplify our formalism. We define $n_i = |R_i|$ to further simplify the equations.

Definition 2.2. *Two functions f and g are **monotonically related** iff $f(x) < f(y) \iff g(x) < g(y)$* [1].

Theorem 2.3. *Consider a precedence graph rooted at R_0, as required by IKKBZ. The rank function r of IKKBZ and the cost function c are monotonically related when evaluated against leaf nodes R_i.*

Proof.

$$r(R_i) = \frac{T(R_i) - 1}{C(R_i)}$$

If we consider the fact R_i is a base relation, $C(R_i) = h_i(n_i) = n_i s_i$ and $T(R_i) = s_i n_i$, so we get

$$r(R_i) = \frac{s_i n_i - 1}{s_i n_i} = 1 - \frac{1}{s_i n_i}$$

Now we have to prove that $r(R_i) = 1 - 1/s_i n_i$ and $c(R_i) = s_i n_i$ are monotonically related. To do that, let us simplify our expressions using $x_i = s_i n_i$, so we now want to prove that $r(R_i) = x_i$ and $c(R_i) = 1 - 1/x_i$ are monotonically related.

We will make the basic assumption that $x_i > 0$, as it is impossible for x_i to be negative, and if $x_i = 0$ then we have the trivial case where one of the joins causes the final result to have no tuples.

$$r(R_i) < r(R_j) \iff 1 - \frac{1}{x_i} < 1 - \frac{1}{x_j} \iff \frac{1}{x_i} > \frac{1}{x_j} \iff x_i < x_j \iff c(R_i) < c(R_j)$$

■

Now we only have to consider some basic facts about IKKBZ:

[1] See https://math.stackexchange.com/questions/1369956/proving-two-functions-are-monotonically-related for the way I came up with this concept.

1. The IKKBZ will give an optimal answer for a star if we only try IKKBZ with the precedence graph rooted at R_0. This comes trivially from theorem 2.1, since we know R_0 will be at the bottom of the join tree, and IKKBZ will give the optimal join tree with R at the bottom if the provided precedence graph is rooted at R. So from this point on we will assume we always run IKKBZ with R_0 as the root of the precedence graph.
2. All of the subtrees of the children of R_0 are chains with one element. Therefore, the whole tree is selected.
3. There are no contradictions, since all chains only have one element.
4. Chains are merged into a single chain by increasing rank $r(R_i)$.

Now consider some basic facts about GreedyStar:

1. GreedyStar joins relations by increasing cost $c(R_i)$.

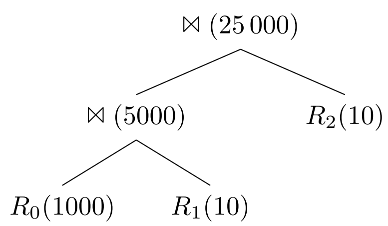
As we have seen before, r and c are monotonically related. If we use the same tie-breaking strategy for IKKBZ and GreedyStar (to break ties when the ranks/costs are equal), IKKBZ and GreedyStar will give the same answer, because, as we have seen above, IKKBZ becomes the same as GreedyStar when applied to a star, since both join base relations by increasing order of r (or c).

If we allow cross-products, here is a counterexample showing that GreedyStar is not optimal.

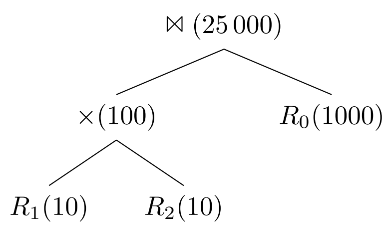
$|R_0| = 1000$, $|R_1| = |R_2| = 10$, $s_1 = s_2 = 0.5$

In the trees, the parenthesis contain the size of the intermediate results.

The best solution that GreedyStar could come up with is the following, with cost $5000 + 25\,000 = 30\,000$:



We now present a better solution using a cross-product, with cost $100 + 25\,000 = 25\,100$



The time complexity of GreedyStar is $O(N \log N)$ where N is the number of tables. This is because the main operations we need to do are calculating $c(R_i) = s_i n_i$ ($O(N)$), sorting leaves by $c(R_i)$ ($O(N \log N)$), and finally, iterating over the sorted leaves and join them to the tree ($O(N)$).

Algorithm	**Complexity (for stars)**
DPsize	$O(4^N)$
DPsub	$O(3^N)$
DPccp	$O(N \cdot 2^N)$
GreedyStar	$O(N \log N)$

In this scenario where we have a star, the complexity of all DP algorithms is higher than GreedyStar. In fact, GreedyStar is polynomial, while all of the DP algorithms are at least exponential time-complexity-wise.

Exercise 3

Using TinyDB, load the TPC H data set such that it is available as `data/tpch` (You can use our snapshot of the data set). Then, execute the following SQL query using the database:

```
select *
    from lineitem l, orders o, customer c
    where l.l_orderkey=o.o_orderkey and
    o.o_custkey=c.c_custkey and
    c.c_name='Customer#000014993'
```

How many rows are there in the result? Please put your answer in the submission pdf. Make sure not to push the data into the git repository!

The result has 34 rows.