# Solutions for Exercise Sheet 6

Richter, Yannick
MTK 03741982
ge78tup@mytum.de

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

TUM – Query Optimization 2022/23
2nd December 2022

Our solutions for Exercise Sheet 6.

## Exercise 1

- Perform the IKKBZ algorithm using the $C_{out}$ cost function on the following query:
  $|A| = 10$; $|B| = 20$; $|C| = 10$; $|D| = 100$
  $f_{AB} = 0.1$; $f_{BC} = 0.2$; $f_{CD} = 0.05$
- Show whether MVP can find a better join (according to $C_{out}$).

---

$$C(\varepsilon) = 0$$
$$C(R_i) = 0 \text{ if } R_i \text{ is root}$$
$$C(R_i) = h_i(n_i) = s_i n_i \text{ otherwise}$$
$$C(S_1 S_2) = C(S_1) + T(S_1) C(S_2)$$

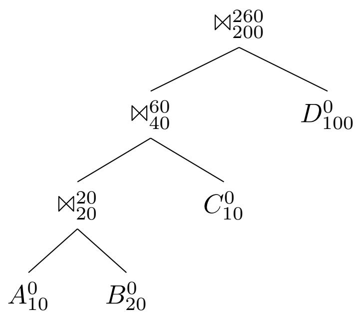
$$T(\varepsilon) = 1$$
$$T(S) = \prod_{R_i \in S} s_i n_i$$

$$rank(S) = \frac{T(S) - 1}{C(S)}$$

With $A$ as a root node. The initial graph is already a chain, so it is also the final step

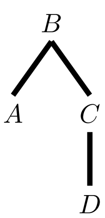
Final graph

Resulting join tree



---

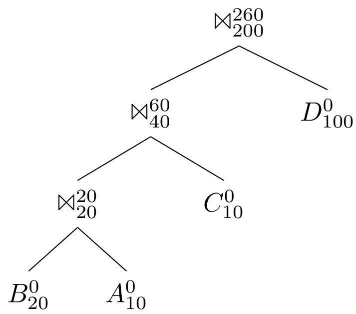
With B as a root node



| | $N$ | $s$ | $C$ | $T$ | rank |
|---|---|---|---|---|---|
| A | 10 | 0.1 | 1 | 1 | 0 |
| C | 10 | 0.2 | 2 | 2 | 1/2 |
| D | 100 | 0.05 | 5 | 5 | 4/5 |

Final graph

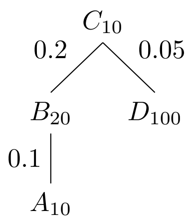

Resulting join tree




---

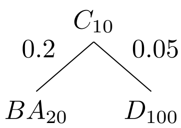
With C as a root node

Precedence graph




| | $N$ | $s$ | $C$ | $T$ | rank |
|---|---|---|---|---|---|
| A | 10 | 0.1 | 1 | 1 | 0 |
| B | 20 | 0.2 | 4 | 4 | 3/2 |
| D | 100 | 0.05 | 5 | 5 | 4/5 |

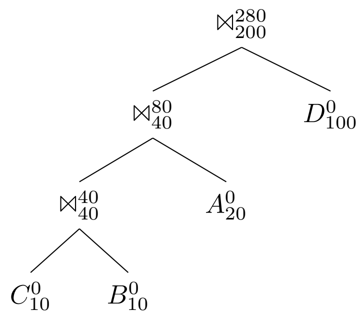
Building compound relations




| | $N$ | $s$ | $C$ | $T$ | rank |
|---|---|---|---|---|---|
| A | 10 | 0.1 | 1 | 1 | 0 |
| B | 20 | 0.2 | 4 | 4 | 3/2 |
| D | 100 | 0.05 | 5 | 5 | 4/5 |
| AB | | | 5 | 5 | 4/5 |

Final graph (Breaking ties in lexicographic order)

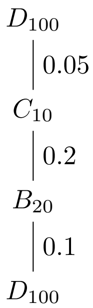

Resulting join tree



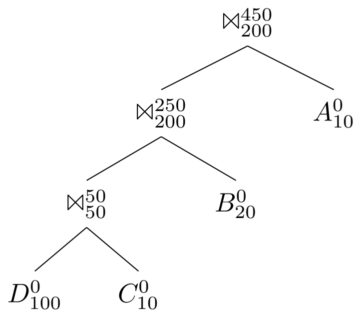

---

With D as a root node

Precedence graph



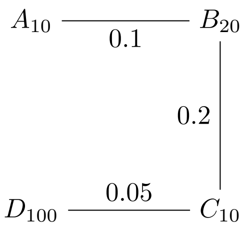

Resulting join tree



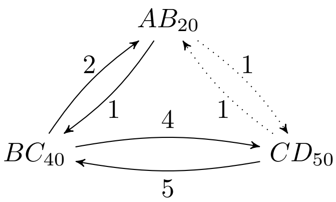

---

$|A| = 10;\ |B| = 20;\ |C| = 10;\ |D| = 100$
$f_{AB} = 0.1;\ f_{BC} = 0.2;\ f_{CD} = 0.05$

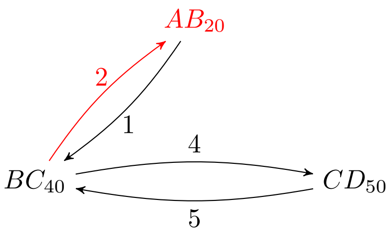
Query Graph

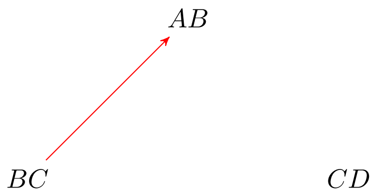



Join Graph

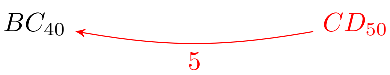


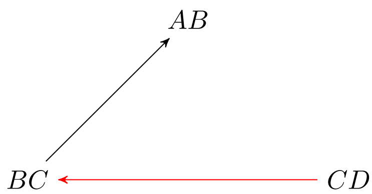

$$
\begin{aligned}
|\bowtie_{AB}| &= 10 * 20 * 0.1 = 20\\
|\bowtie_{BC}| &= 20 * 10 * 0.2 = 40\\
|\bowtie_{CD}| &= 10 * 100 * 0.05 = 50\\
w(AB \to BC) &= \tfrac{20}{20} = 1\\
w(BC \to AB) &= \tfrac{40}{20} = 2\\
w(BC \to CD) &= \tfrac{40}{10} = 4\\
w(CD \to BC) &= \tfrac{50}{10} = 5
\end{aligned}
$$



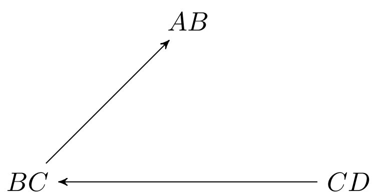

$$AB_{new} = 40 * 10 * 0.1 * 40 = 80$$
$AB_{80}$



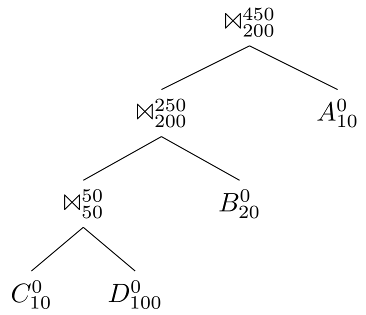




$$BC_{new} = 50 * 10 * 0.2 + 50 = 150$$
$AB_{80}$

$BC_{150}$ $CD_{50}$

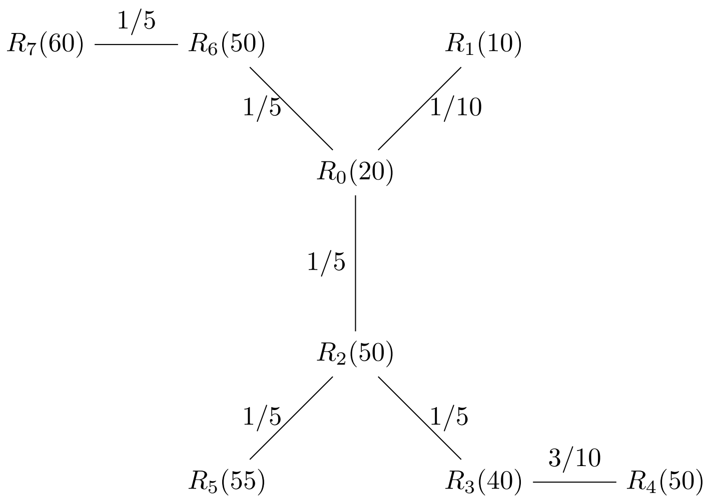


Resulting join tree



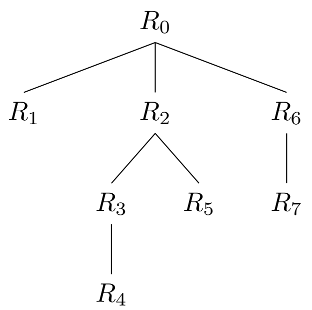
# Exercise 2

Consider the following query graph with selectivities and cardinalities:

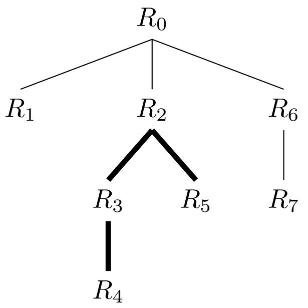



- Give the precedence graph rooted in $R_0$
- Perform the IKKBZ algorithm on this precedence graph using the $C_{out}$ cost function. Give the resulting join tree.

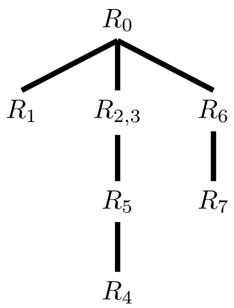
---



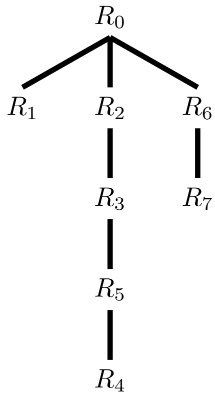

---



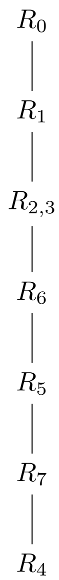

| | $N$ | $s$ | $C$ | $T$ | rank |
|---|---|---|---|---|---|
| 1 | 10 | 0.1 | 1 | 1 | 0 |
| 2 | 50 | 0.2 | 10 | 10 | 9/10 |
| 3 | 40 | 0.2 | 8 | 8 | 7/8 |
| 4 | 50 | 0.3 | 15 | 15 | 14/15 |
| 5 | 55 | 0.2 | 11 | 11 | 10/11 |
| 6 | 50 | 0.2 | 10 | 10 | 9/10 |
| 7 | 60 | 0.2 | 12 | 12 | 11/12 |




| | $N$ | $s$ | $C$ | $T$ | rank |
|---|---|---|---|---|---|
| 1 | 10 | 0.1 | 1 | 1 | 0 |
| 2,3 | 400 | 0.2 | 90 | 80 | $79/90 \approx 0.7887$ |
| 4 | 50 | 0.3 | 15 | 15 | $14/15 \approx 0.9333$ |
| 5 | 55 | 0.2 | 11 | 11 | $10/11 \approx 0.9091$ |
| 6 | 50 | 0.2 | 10 | 10 | $9/10 \approx 0.9000$ |
| 7 | 60 | 0.2 | 12 | 12 | $11/12 \approx 0.9167$ |

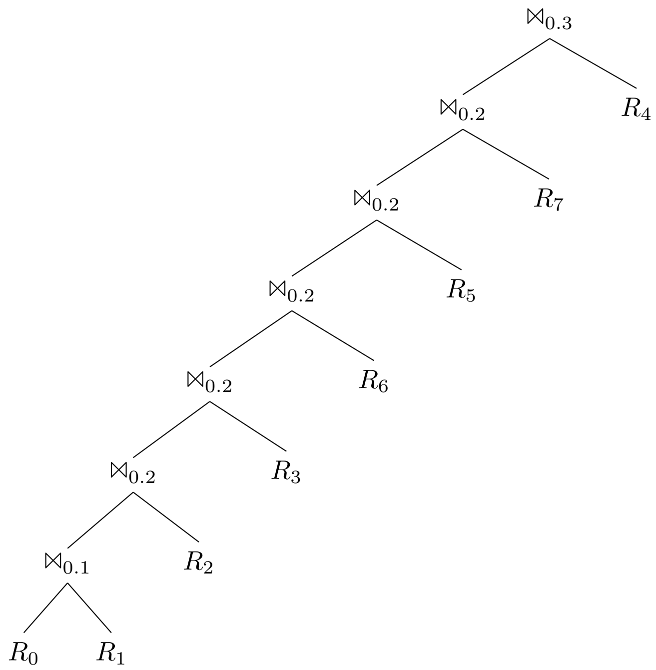



| | $N$ | $s$ | $C$ | $T$ | rank |
|---|---|---|---|---|---|
| 1 | 10 | 0.1 | 1 | 1 | 0 |
| 2 | 50 | 0.2 | 10 | 10 | 9/10 |
| 3 | 40 | 0.2 | 8 | 8 | 7/8 |
| 4 | 50 | 0.3 | 15 | 15 | 14/15 |
| 5 | 55 | 0.2 | 11 | 11 | 10/11 |
| 6 | 50 | 0.2 | 10 | 10 | 9/10 |
| 7 | 60 | 0.2 | 12 | 12 | 11/12 |



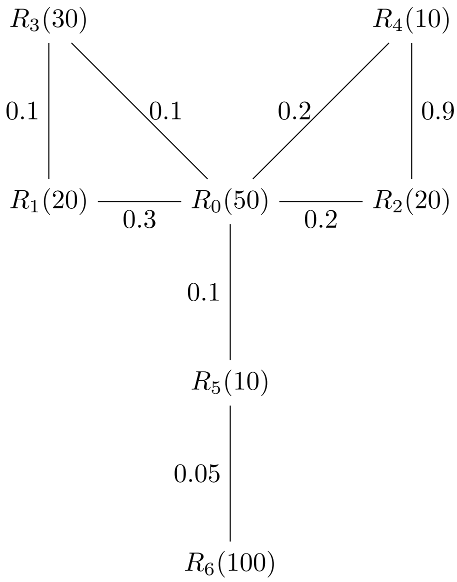
Final graph


Resulting join tree



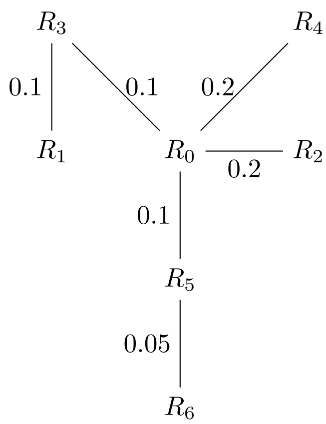
## Exercise 3

Consider the following query graph with selectivities and cardinalities:

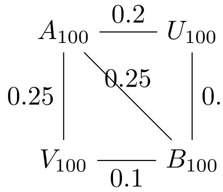


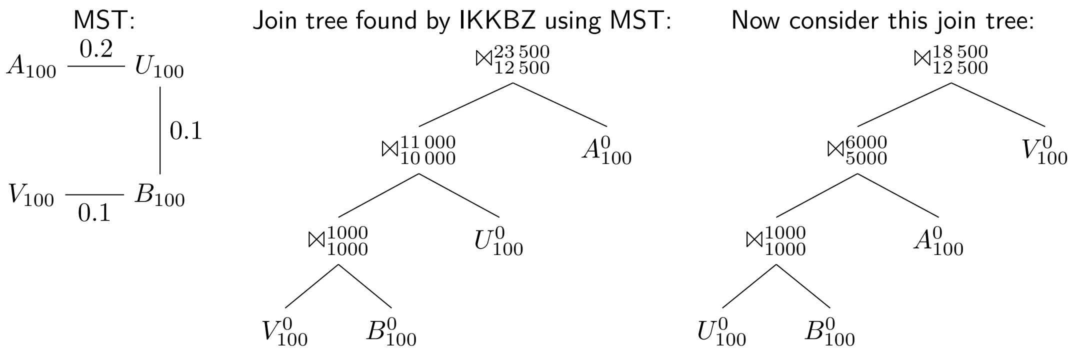

- Give the minimum spanning tree (MST) for this query graph.
- Answer the following questions briefly (1-2 sentences each):
  - What guarantees can you give for the resulting join graph if you use IKKBZ on the MST?
  - Why does it make sense to use a MST instead of any other spanning tree?

---




- **What guarantees can you give for the resulting join graph if you use IKKBZ on the MST?** IKKBZ gives an optimal left-deep join tree for the MST, but that is the only guarantee we have. We can't even guarantee it is the best possible ST that IKKBZ could be provided, because of this counterexample:






MST and then IKKBZ finds a join tree with cost $23\,500$. However, the actual optimal tree for this acyclic query graph is the one on the right, with cost $18\,500$. So we can't even guarantee that the MST is the best MST that we can provide to the IKKBZ so that IKKBZ will give us the best join tree.

But it is a good heuristic. Also, the sparsest the original query graph is, the closer the MST is to the best ST that we could provide to IKKBZ.

- **Why does it make sense to use a MST instead of any other spanning tree?** A: IKKBZ can only work with trees, so we must build a spanning tree. Aside from that, the MST is a good spanning tree that we can supply to IKKBZ because we want to minimize the sizes of intermediate results, so we prefer to join using the most selective joins (aka, edges with least selectivity).