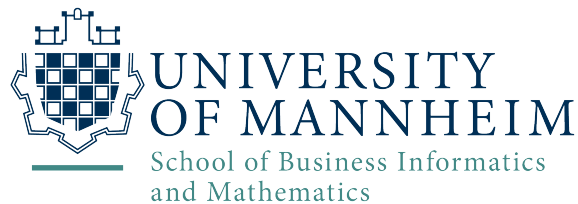
Database Systems II – Exercise #8

Sheet #8: Operator Trees and Virtual Machines
Sheet #9: Typical Exam Questions, CSB$^+$ Trees, Boolean Expressions

Daniel Flachs

Chair of Practical Computer Science III:
Database Management Systems

08/05/2019



Created May 8, 2019

Contents

Contents

Task 1

Building an operator tree is one possibility to interpret a program. Every operator is encapsulated in a class and all operators share a common interface. In our case, this is the `eval` function.

In the files provided on the website, you can find a simple implementation of operators to build an operator tree for expressions in Boolean algebra. You can also find a small example.

So far, the following operations exist:

- `Literal`, which just creates a literal like `a := true`,
- `Not`, which negates an expression, e.g. `Not(a)`,
- `Or`, the logical or operator, to evaluate disjunctions,
- `Implication`, which evaluates an implication, like $a \Rightarrow b$.

Task 1

a) Implement the logical `And` operator.

b) Construct and evaluate the following expression:

$$(a \wedge b) \vee c$$

You can choose values for $a, b, c \in \{\text{true}, \text{false}\}$.

Task 2

Another possibility to interprete a program is to run it in a virtual machine. In the files provided on the website, you can find a simple implementation of a virtual machine that supports only integer operations.

b) Give C++ code that is equivalent to the following instructions.

```
1  PUSH, 111,
2  DUPLICATETOP,
3  PRINTTOP,                  // JMP jumps here
4  DUPLICATETOP,
5  PUSH, 122,
6  SUB,
7  JMPIFFALSE, 16,
8  PUSH, 1,
9  ADD,
10 JMP, 3,
11 POP,                       // JMPIFFALSE jumps here
12 HALT
```

Task 2

b) Give C++ code that is equivalent to the following instructions.

```
1  PUSH, 111,
2  DUPLICATETOP,
3  PRINTTOP,                  // JMP jumps here
4  DUPLICATETOP,
5  PUSH, 122,
6  SUB,
7  JMPIFFALSE, 16,
8  PUSH, 1,
9  ADD,
10 JMP, 3,
11 POP,                       // JMPIFFALSE jumps here
12 HALT
```

Solution

```
1 for (int i = 111; i <= 122; ++i) {
2   std::cout << i << std::endl;
3 }
```

Task 2

c) (i) Implement a multiplication operation `MUL` that takes the last two elements from the stack, removes them, and pushes their product onto the stack.

(ii) Implement an integer division operation `DIV`, that takes the last two elements from the stack, removes them, and pushes their quotient onto the stack. Make sure that the order of the operands is consistent with how the SUB operations handles its operands.

(iii) Also implement a conditional jump operation `JMPIFTRUE` that removes the last element from the stack and jumps to a specified location in the program if the element evaluates to `true`, i. e., a number unequal to zero.

d) Replace `program` with a program that computes $(5 + 3 \cdot 4) \div 2$ and outputs the result.

Contents

Task 1

Find questions that you would consider appropriate to be asked in a DBS II exam. Also assign points to your task and give an outline of what is expected as the solution. A typical exam covers questions of different types like

- *name/list n different properties of Y*,
- *discuss advantages and disadvantages of X*,
- *implement a function that does X*,
- *given the following piece of code, complete the code such that it does X*,
- *given the following piece of code/figure/graph/architecture, answer the following questions: ...*,
- *apply algorithm/method/technique X known from the lecture/exercise to the following problem Y*,
- or other types of questions.

Send your results to daniel.flachs@uni-mannheim.de. Appropriate questions (or variants thereof) might appear in the exam.

Task 2

Read (at least) Sections 1 to 3 of the paper
Making B^+-Trees Cache Conscious in Main Memory
by Rao and Ross.

Task 3

Assume you are given the following query:

```
SELECT *
FROM R
WHERE Age > 27 and Income > 30.000 and Weight < 75;
```

We refer to the predicates in the query by the set

$$P = \{Age > 27, Income > 30\,000, Weight < 75\} = \{p_1, p_2, p_3\}$$

Task 3

Furthermore, assume you are given the following sample taken from R:

ID	Age	Income	Weight
1	28	40,000	80
2	30	55,000	50
3	27	37,000	75
4	40	60,000	60
5	42	62,000	85
6	22	15,000	55
7	70	20,000	67
8	50	80,000	57
9	55	85,000	86
10	33	42,000	58

Task 3a

Which tuples fulfill which predicates?

$$P = \{p_1, p_2, p_3\} = \{Age > 27, Income > 30\,000, Weight < 75\}$$

ID	Age	Income	Weight	p_1	p_2	p_3
1	28	40,000	80	✓	✓	✗
2	30	55,000	50	✓	✓	✓
3	27	37,000	75	✗	✓	✗
4	40	60,000	60	✓	✓	✓
5	42	62,000	85	✓	✓	✗
6	22	15,000	55	✗	✗	✓
7	70	20,000	67	✓	✗	✓
8	50	80,000	57	✓	✓	✓
9	55	85,000	86	✓	✓	✗
10	33	42,000	58	✓	✓	✓

Task 3a

Let $P' \subseteq P$.

$$F_\beta(P') = \bigwedge_{p_i \in P'} p_i \qquad F_\gamma(P') = \bigwedge_{p_i \in P'} p_i \wedge \bigwedge_{p_i \notin P'} \neg p_i$$

$$\gamma = \begin{pmatrix} 0.0 \\ 0.0 \\ 0.1 \\ 0.3 \\ 0.1 \\ 0.1 \\ 0.0 \\ 0.4 \end{pmatrix} \qquad \begin{matrix} 000 \\ 100 \\ 010 \\ 110 \\ 001 \\ 101 \\ 011 \\ 111 \end{matrix} \qquad \begin{matrix} \neg p_1 \wedge \neg p_2 \wedge \neg p_3 \\ p_1 \wedge \neg p_2 \wedge \neg p_3 \\ \neg p_1 \wedge p_2 \wedge \neg p_3 \\ p_1 \wedge p_2 \wedge \neg p_3 \\ \neg p_1 \wedge \neg p_2 \wedge p_3 \\ p_1 \wedge \neg p_2 \wedge p_3 \\ \neg p_1 \wedge p_2 \wedge p_3 \\ p_1 \wedge p_2 \wedge p_3 \end{matrix}$$

It holds: $\sum_{P' \subseteq P} \gamma(P') = 1$ ✓

Task 3b

Give the complete design matrix C that is associated with $|P| = 3$.

The complete design matrix can be found recursively by

$$C_{|P|} = \begin{pmatrix} C_{|P|-1} & C_{|P|-1} \\ 0 & C_{|P|-1} \end{pmatrix} \quad \text{and} \quad C_0 = (1)\,.$$

$$C_3 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Task 3c

Compute $C\gamma$.

$$\beta = C\gamma = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0.0 \\ 0.0 \\ 0.1 \\ 0.3 \\ 0.1 \\ 0.1 \\ 0.0 \\ 0.4 \end{pmatrix} = \begin{pmatrix} 1.0 \\ 0.8 \\ 0.8 \\ 0.7 \\ 0.6 \\ 0.5 \\ 0.4 \\ 0.4 \end{pmatrix} \widehat{=} \begin{pmatrix} \emptyset \\ p_1 \\ p_2 \\ p_1 \wedge p_2 \\ p_3 \\ p_1 \wedge p_3 \\ p_2 \wedge p_3 \\ p_1 \wedge p_2 \wedge p_3 \end{pmatrix}$$