Chair of applied computer science III
Prof. Dr. Guido Moerkotte
Email: moerkotte@uni-mannheim.de
UNIVERSITY OF MANNHEIM

Query Optimization — Exercise sheet 5

Exercise 1

Exercise 1 a)

Classify the join ordering algorithmns discussed in the lecture. Fill in the following table:

Name	Query Graph	Join-Tree	Cross products	cost functions	complexity	optimal	remarks

Solution

Name	Query Graph	Join-Tree	Cross products	Cost functions	Complexity	optimal	Remarks
DP-Linear-1	arbitrary	linear	optional	arbitrary	$O(2^n)$	yes	
DP-Linear-2	arbitrary	linear	yes	arbitrary	$O(3^n)$	yes	
DP-Bushy	arbitrary	bushy	yes	arbitrary	$O(3^n)$	yes	
DPSize	connected	bushy	no	arbitrary	$O(n^4) \dots O(4^n/\sqrt{n})$	yes	complexity depends on the query graph
DPSub	connected	bushy	no	arbitrary	$O(2^n) \dots O(3^n)$	yes	complexity depends on the query graph
DPccp	connected	bushy	no	arbitrary	$O(n^3) \dots O(3^n)$	yes	complexity depends on the query graph

Exercise 1 b)

Consider the relations

$$|R_0| = 10, |R_1| = 2, |R_2| = 100$$

and the join selectivities

$$f_{R_0R_1} = 0.5, f_{R_1R_2} = 0.1$$

Note down the corresponding DP-tables for `DPsub` and `DPsize`.
Keep track of all plans written to the DP-table and mark the final plan.
Use C_{out} as the cost function.

Solution

Size = 1	R_0	0	*	R_1	0	*	R_2	0	*		
Size = 2	$R_0 \bowtie R_1$	10	*	$R_0 \bowtie R_2$	not connected		$R_1 \bowtie R_2$	20	*		
	$R_1 \bowtie R_0$	10		$R_2 \bowtie R_0$	not connected		$R_2 \bowtie R_1$	20			

Size = 3	$(R_0 \bowtie R_1) \bowtie R_2$	110	*
	$R_2 \bowtie (R_0 \bowtie R_1)$	110	
	$(R_1 \bowtie R_2) \bowtie R_0$	120	
	$R_0 \bowtie (R_1 \bowtie R_2)$	120	
	$(R_0 \bowtie R_2) \bowtie R_1$	not connected	
	$R_1 \bowtie (R_0 \bowtie R_2)$	not connected	

Exercise 1 c)

Now, note down the DP-tables corresponding to the above query graph for `DPccp`. The subscripts denote the order in which BFS (breadth-first search) visits the nodes.

Solution

Procedure	S_1	Procedure	S_2	Plan	C_{out}
ECsg	$\{R_2\}$	ECmp	$\emptyset$		
ECsg	$\{R_1\}$	ECmp	$\{R_2\}$	$R_1 \bowtie R_2$	20 *
				$R_2 \bowtie R_1$	20
ECsgRec	$\{R_1, R_2\}$	ECmp	$\emptyset$		
ECsg	$\{R_0\}$	ECmp	$\{R_1\}$	$R_0 \bowtie R_1$	10 *
				$R_1 \bowtie R_0$	10
		ECsgRec	$\{R_1, R_2\}$	$\{R_1, R_2\} \bowtie R_0$	120
				$R_0 \bowtie \{R_1, R_2\}$	120
ECsgRec	$\{R_0, R_1\}$	ECmp	$\{R_2\}$	$\{R_0, R_1\} \bowtie R_2$	110 *
				$R_2 \bowtie \{R_0, R_1\}$	110
ECsgRec	$\{R_0, R_1, R_2\}$	ECmp	$\emptyset$		

Exercise 2

Exercise 2 a)

Implement either `DP-Linear-1`, `DP-Linear-2` or `DP-Bushy`. You may use the helper classes provided in the solution code.

Solution

See code.

Exercise 2 b)

Among how many join trees do `DP-Linear-1` (with cross products) and `DP-Bushy` select the optimal one?

Solution

DP-Linear-1: $n!$
DP-Bushy: $n!C_{n-1}$
where n is the number of relations.

Exercise 2 c)

Bonus
Compare the number of plans generated by the enumerator from the first exercise with the number generated partial trees (denoted by variable name CurrTree in the pseudocode) in `DP-bushy`.

Solution

Find the solution yourself!

Exercise 3

In the algorithms discussed in the lecture, We have already seen several enumeration techniques in the algorithms discussed in the lecture. This exercise is to recall these techniques.

Exercise 3 a)

How to enumerate all sizes for the *two sides of a plan* (i.e., all unordered pairs) for all plan sizes? Write down the order in which values created.

Solution

für 4 Relationen:
len: 2; left: 1; right: 1
len: 3; left: 1; right: 2
len: 4; left: 1; right: 3
len: 4; left: 2; right: 2

Exercise 3 b)

How to enumerate all possible values of a bitvector of size n in ascending order? Write down the order in which values created.

Solution

für 4 Relationen
plan: 0001
plan: 0010
plan: 0011
plan: 0100
plan: 0101

plan: 0110
plan: 0111
plan: 1000
plan: 1001
plan: 1010
plan: 1011
plan: 1100
plan: 1101
plan: 1110
plan: 1111

Exercise 3 c)

How to enumerate all subsets of a set of n values? Assume that a set is implemented as a bitvector. Write down the order in which values created.

Solution

für 4 Relationen
bits: 0001
bits: 0010
bits: 0011
bits: 0100
bits: 0101
bits: 0110
bits: 0111
bits: 1000
bits: 1001
bits: 1010
bits: 1011
bits: 1100
bits: 1101
bits: 1110
bits: 1111

Solution

code

```
package samples;

public class DP_Enumeration {

        static void enumByLength(int numRel) {
                for (int len = 2; len <= numRel; ++len) {
                        for (int l = 1; l <= len/2; ++l) {
                                int r = len - l;
                                /* enumerate disjoint subsets of S with length l and r */
```

```
                        System.out.println("len: " + len +
                                        "; left: " + l +
                                            "; right: " + r);
                    }
                }
            }

            static void enumByNumber(int numRel) {
                for (int i = 1; i < (1 << numRel); ++i) {
                    System.out.println("plan: " + Integer.toBinaryString(i));
                }
            }

            static void enumSubset(int set) {
                for (int bits=set&(-set);bits!=0;bits=set&(bits-set)) {
                System.out.println("bits: " + Integer.toBinaryString(bits));
            }
            }

            public static void main(String[] args) {
                final int numRel = 4;

                System.out.println("enumerate by length");
                enumByLength(numRel);
                System.out.println("enumerate by number");
                enumByNumber(numRel);

                // initialize bits
                int set = 0;
                for(int i = 0; i < numRel; ++i) {
                    set |= (1 << i);
                }
                System.out.println("enumerate by subsets of " + Integer.toBinaryString(set));
                enumSubset(set);
            }
        }
```