

Prof. Dr. Guido Moerkotte

Email: moer@pi3.informatik.uni-mannheim.de

Pit Fender

B6, 29, Raum C0.05

68131 Mannheim

Telefon: (0621) 181-2517

Email: pfender@pi3.informatik.uni-mannheim.de

Anfrageoptimierung
Herbst-/Wintersemester 08

5. Übungsblatt
22. Oktober 2008

Aufgabe 1

Verdeutlichen Sie sich, z.B. anhand von Codefragmenten, die Möglichkeiten zum Aufzählen der Alternativen beim dynamischen Programmieren. Geben Sie für jede Alternative an, in welcher Reihenfolge die Teilpläne generiert werden, z.B. für 4 Relationen.

Aufgabe 1 a)

Aufzählen der Alternativen nach der Anzahl der Relationen im Teilplan.

Lösung

für 4 Relationen:

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

Aufgabe 1 b)

Aufzählen der Planalternativen durch Zählen und Interpretation als Bitvector.

Lösung

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

Aufgabe 1 c)

Aufzählen der Teilmengen einer Menge.

Lösung

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

Lösung

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; ++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;

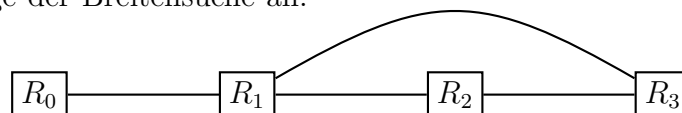
    System.out.println("enumerate by length");
    enumByLength(numRel);
    System.out.println("enumerate by number");
    enumByNumber(numRel);

    // 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);
}
}

```

Aufgabe 1 d)

Geben Sie die Sequenz von erzeugten Teilgraphen an, die die Funktion `EnumerateCsg` für den folgenden Anfragegraphen erzeugt. Erstellen Sie dazu eine Tabelle analog zu dem Beispiel aus der Vorlesung. Markieren Sie auch die rekursiven Aufrufe der Funktion `EnumerateCsgRec`. Die Subskripte der Relationennamen geben die Position in der Besuchsreihenfolge der Breitensuche an.



Lösung

S	EnumerateCsg		emit
	X	$N = \mathcal{N}(S) \setminus X$	
	outer ($i = 3$)		$\{ 3 \}$
x $\{ 3 \}$	$\{ 0,1,2,3 \}$	\emptyset	-
	outer ($i = 2$)		$\{ 2 \}$
x $\{ 2 \}$	$\{ 0,1,2 \}$	$\{ 3 \}$	$\{ 2,3 \}$
			$\{ 2,3 \}$
xx $\{ 2,3 \}$	$\{ 0,1,2,3 \}$	\emptyset	-
	outer ($i = 1$)		$\{ 1 \}$
x $\{ 1 \}$	$\{ 0,1 \}$	$\{ 2,3 \}$	$\{ 1,2 \}$
			$\{ 1,3 \}$
			$\{ 1,2,3 \}$
xx $\{ 1,2 \}$	$\{ 0,1,2,3 \}$	\emptyset	-
xx $\{ 1,3 \}$	$\{ 0,1,2,3 \}$	\emptyset	-
xx $\{ 1,2,3 \}$	$\{ 0,1,2,3 \}$	\emptyset	-
	outer ($i = 0$)		$\{ 0 \}$
x $\{ 0 \}$	$\{ 0 \}$	$\{ 1 \}$	$\{ 0,1 \}$
			$\{ 0,1 \}$
xx $\{ 0,1 \}$	$\{ 0,1 \}$	$\{ 2,3 \}$	-
			$\{ 0,1,2 \}$
			$\{ 0,1,3 \}$
			$\{ 0,1,2,3 \}$
xxx $\{ 0,1,2 \}$	$\{ 0,1,2,3 \}$	\emptyset	-
xxx $\{ 0,1,3 \}$	$\{ 0,1,2,3 \}$	\emptyset	-
xxx $\{ 0,1,2,3 \}$	$\{ 0,1,2,3 \}$	\emptyset	-

Aufgabe 1 e)

Geben Sie für den Anfragegraphen aus der vorherigen Aufgabe alle Teilgraphen an, die von der Funktion `EnumerateCmp` für $S_1 = \{1\}$ generiert werden. Erstellen Sie dazu eine Tabelle wie im vorherigen Beispiel.

Lösung

S	EnumerateCsg		emit
	X	N	
(outer)	$\{ 0,1 \}$	$\{ 2,3 \}$	$\{ 3 \}$
x $\{ 3 \}$	$\{ 0,1,2,3 \}$	\emptyset	-
(outer cont.)			$\{ 2 \}$
x $\{ 2 \}$	$\{ 0,1,2 \}$	$\{ 3 \}$	$\{ 2,3 \}$
xx $\{ 2,3 \}$	$\{ 0,1,2,3 \}$	\emptyset	-

Die Erzeugung der Teilmengen S_1 und S_2 im Überblick

Prozedur	S_1	Prozedur	S_2
ECsg	{2}	ECmp	{3}
	{1}	ECmp	{3}
	{1}		{2}
	{1}	ECsgRec	{2,3}
ECsgRec	{1,2}		{3}
	{1,3}		{2}
ECsg	{0}	ECmp	{1}
	{0}	ECsgRec	{1,2}
	{0}	ECsgRec	{1,3}
	{0}	ECsgRec	{1,2,3}
ECsgRec	{0,1}	ECmp	{3}
	{0,1}	ECmp	{2}
	{0,1}	ECsgRec	{2,3}
→ ECsgRec	{0,1,2}	ECmp	{3}
→ ECsgRec	{0,1,3}	ECmp	{2}

Aufgabe 2

Aufgabe 2 a)

Klassifizieren Sie alle besprochenen DP-Algorithmen aus der Vorlesung basierend auf der Tabelle von Übungsblatt 3.

Lösung

Name	Query Graph	Join-Tree	Kreuzprodukte	Kostenfunktion	Komplexität	optimal	Anmerkungen
DP-Linear-1	beliebig	linear	optional	beliebig	$O(2^n)$	ja	
DP-Linear-2	beliebig	linear	ja	beliebig	$O(3^n)$	ja	
DP-Bushy	beliebig	bushy	ja	beliebig	$O(3^n)$	ja	
DPSize	connected	bushy	nein	beliebig	$O(n^4) \dots O(4^n)$	ja	Komplexität abhängig von query graph
DPSub	connected	bushy	nein	beliebig	$O(2^n) \dots O(3^n)$	ja	Komplexität abhängig von query graph
DPccp	connected	bushy	nein	beliebig	$O(n^3) \dots O(3^n)$	ja	Komplexität abhängig von query graph

Aufgabe 2 b)

Erzeugen Sie (von Hand) die DP-Tabelle, die durch **DPSub** oder **DPSize** für die Relationen R_0, R_1, R_2 mit den Kardinalitäten $|R_0| = 10, |R_1| = 2, |R_2| = 100$ und den Selektivitäten $f_{R_0 R_1} = 0.5$ und $f_{R_1 R_2} = 0.1$ (Kostenfunktion C_{out}) berechnet wird. Berücksichtigen Sie nur Pläne ohne Kreuzprodukte. Markieren Sie den endgültigen Tabelleneintrag, aber geben Sie auch verworfene und ungültige Einträge an.

Lösung

Size = 1	R_0	0	*	R_1	0	*	R_2	0	*
Size = 2	$R_0 \bowtie R_1$	10	*	$R_0 \bowtie R_2$	nicht verbunden		$R_1 \bowtie R_2$	20	*
	$R_1 \bowtie R_0$	10		$R_2 \bowtie R_0$	nicht verbunden		$R_2 \bowtie R_1$	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							
	$(R_0 \bowtie R_2) \bowtie R_1$	nicht verbunden							
	$R_1 \bowtie (R_0 \bowtie R_2)$	nicht verbunden							

Aufgabe 3

Erweitern Sie den dynamischen Programmieralgorithmus `DPSub` vom letzten Übungsblatt, so daß nur noch Pläne ohne Kreuzprodukte erzeugt werden.

Lösung

siehe `DPSub.java`