

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

4. Übungsblatt  
15. Oktober 2007

---

---

### Aufgabe 1

---

Einige Algorithmen wie z.B. IKKBZ benötigen einen azyklischen Joingraph. Wie könnten sie (sinnvoll) auch für zyklische Joingraphen verwendet werden? Skizzieren Sie einen optimalen Algorithmus und ggf. eine gute Heuristik.

#### Lösung

Optimal: Alle Spannbäume ausprobieren, die entfernten Kanten als zuzustliches Join-Prdikat an entsprechender Stelle hinzufügen. Die Kardinalität beim Einfügen der letzten Relation des aufgetrennten Zyklus zum Plan berechnet sich dann aus dem Produkt zweier Filterfaktoren (der noch vorhandenen Kante und der gelöschten).

Die Anzahl der zu betrachtenden precedence Graphen entspricht nicht mehr der Anzahl der Relation im Query Graph, sondern dem Produkt der Anzahl der Relationen und der Kanten im Zyklus.

Heuristik: z.B. minimaler Spannbaum, der durch das Löschen der teuersten Kante je Zyklus entsteht.

---

### Aufgabe 2

---

Modifizieren Sie den Pseudo-Code von DP-Linear-1 aus der Vorlesung so, daß er bushy Join-Trees erzeugt.

#### Lösung

(entspricht Algorithmus DPSize)

DP-BUSHY-1( $R_1 \dots R_n$ )

▷ Input: a set of relations to be joined

▷ Output: a optimal bushy join tree

```

1  for  $i \leftarrow 1 \dots n$ 
2      do BESTTREE( $R_i$ )  $\leftarrow R_i$ 
3  for  $i \leftarrow 2 \dots n$ 
4      do for  $l \leftarrow 1 \dots i - 1$ 
5          do  $r \leftarrow i - l$ 
6              for all  $L \subset S, |L| = l, R \subset S, |R| = r, L \cap R = \emptyset$ 
7                  do if NOCROSSPRODUCTS
8                      then if ( $\neg$  CONNECTED( $L$ )  $\vee$ 
9                           $\neg$  CONNECTED( $R$ )  $\vee$ 
10                              $\neg$  CONNECTED( $L, R$ ))
11                              then continue
12                       $CurrTree \leftarrow$ 
13                          CREATEJOINTREE(BESTTREE( $L$ ), BESTTREE( $R$ ))
14                      if (BESTTREE( $S$ ) = NULL  $\vee$ 
15                          COST(BESTTREE( $S$ )) > COST( $CurrTree$ ))
16                          then BESTTREE( $S$ )  $\leftarrow CurrTree$ 
17  return BESTTREE( $R_1 \dots R_n$ )

```

### Aufgabe 3

#### Aufgabe 3 a)

Klassifizieren Sie alle Greedy-Heuristiken und den IKKBZ-Algorithmus aus der Vorlesung basierend auf der Tabelle des vorangegangenen Aufgabenblattes.

#### Lösung

Name	Query Graph	Join-Tree	Kreuzprodukte	Kostenfunktion	Komplexität	optimal	Anmerkungen
Greedy-1	beliebig	left-deep	ja	beliebig	$O(n \lg n)$	nein	
Greedy-2	beliebig	left-deep	ja	beliebig	$O(n \lg n)$	nein	
Greedy-3	beliebig	left-deep	ja	beliebig	$O(n^2 \lg n)$	nein	
GOO	beliebig	bushy	ja	beliebig	$O(n^2)$	nein	
IKKBZ	azyklisch	left-deep	nein	ASI	$O(n \lg n)$	ja	1 Kostenfunktion je Pädikat

#### Aufgabe 3 b)

Geben Sie ein Beispiel an, bei dem GreedyHeuristic-1 nicht die optimale Lösung findet, IKKBZ aber schon.

#### Lösung

$$|R_1| = 100, |R_2| = 200, |R_3| = 300, f_{12} = \frac{1}{10}, f_{23} = \frac{1}{100}$$

Bei  $R_1$  oder  $R_3$  als Wurzel  $\rightarrow$  Precedence Graph ist eine Kette, kein Berechnung des Ranks nötig!

Wird  $R_2$  als Wurzel ausgewählt  $\rightarrow$  Rank von  $R_1: \frac{9}{10}$ , Rank von  $R_3: \frac{2}{3}$

Plan	$C_{out}$
Greedy: $(R_1 \bowtie R_2) \bowtie R_3$	8000
IKKBZ: z.B. $R_1 \bowtie (R_2 \bowtie R_3)$	6600

#### Aufgabe 4

##### Aufgabe 4 a)

Welche Voraussetzungen müssen erfüllt sein, damit ein Problem sinnvoll mit dynamischer Programmierung gelöst werden kann?

##### Lösung

- Problem muß durch optimale Lösungen von Teilproblemen lösbar sein
- damit DP effizienter ist, als vollständige Enumeration aller Alternativen, muß es wiederkehrende Teilprobleme geben.

##### Aufgabe 4 b)

Was heisst das für das Join-Ordering Problem bzw. insbesondere für die Kostenfunktion?

##### Lösung

- Kostenfunktion muß monoton steigen mit zunehmender Problem gröÙe.
- Properties (z.B. Sortierung, Duplikate, gemeinsame Teilausdrücke) bereiten z.B. Probleme, weil höhere Kosten von Teilproblemen zu niedrigeren Kosten insgesamt führen können.

##### Aufgabe 4 c)

Welche Voraussetzungen müssen beim Aufzählen der Planalternativen beim dynamischen Programmieren erfüllt sein?

##### Lösung

Alle Teilprobleme eines Problems müssen generiert werden, bevor das Problem gelöst werden kann.

---

### Aufgabe 5

---

Erweitern Sie das Programm vom vorherigen Übungsblatt so, dass eine möglichst günstige Joinreihenfolge erzeugt wird. Verwenden Sie z.B. `GreedyHeuristic-1`.

#### Lösung

siehe `GreedyJoin.java`

---

### Aufgabe 6

---

Implementieren Sie den `DPSub` (d.h. `DP-Bushy`) Algorithmus aus der Vorlesung. Die erzeugten Pläne dürfen Kreuzprodukte enthalten.

#### Lösung

siehe `DPJoin.java`