

Aufgabe 1

Modifizieren Sie den Memoization Algorithmus, so dass er keine Kreuzprodukte mehr berücksichtigt!

Lösung

MEMOIZATION(V)

▷ **Input:** a set of relations $V = \cup_i \{R_i\}$
▷ **Output:** an optimal join tree for V

- 1 **for** $i \leftarrow 1$ **to** n
- 2 **do** $BestTree(\{R_i\}) \leftarrow R_i$
- 3 **return** MEMOIZATIONSUB(R)

MEMOIZATIONSUB(S)

▷ **Input:** a (sub-) set of relations S
▷ **Output:** an optimal join tree for S

- 1 **if** $BestTree(S) \neq \text{NULL}$
- 2 **then return** $BestTree(S)$
- 3 **for all** $S_1 \subset S$ and $S_1 \neq \emptyset$
- 4 **do if** ISCONNECTED(S_1)
- 5 **then continue**
- 6 $S_2 \leftarrow S - S_1$
- 7 **if** ISCONNECTED(S_2)
- 8 **then continue**
- 9 $CurrTree \leftarrow createTree(\text{MEMOIZATIONSUB}(S_1), \text{MEMOIZATIONSUB}(S_2))$
- 10 **if** $BestTree(S) = \text{NULL}$ or $cost(BestTree(S)) > cost(CurrTree)$
- 11 **then** $BestTree(S) \leftarrow CurrTree$
- 12 **return** $BestTree(S)$

Aufgabe 1 a)

Was fällt Ihnen bezüglich der Anzahl der Connection-Tests auf? Vergleichen sie diese mit DBSize aus eine der letzten Übungen! Begründen Sie!

Lösung

Der Connection-Test zwischen S_1 und S_2 entfällt. Dies folgt aus der Tatsache, dass S ein verbundener Teilgraph sein muss. Wenn S_1 und S_2 je verbundene Teilgraphen sind, dann muss auch $S_1 \cup S_2$ verbunden sein!

Aufgabe 1 b)

Worin liegen die Vor- und Nachteile gegenüber DP?

Lösung

Nachteil: rekursiv, kein bekannter Algorithmus vergleichbar zu DPcsgCmp! Generell langsamer als DP.

Vorteil: Cost pruning ist möglich.

Aufgabe 2

Skizzieren Sie einen Algorithmus, um den optimalen join tree für chain queries zu berechnen (beliebige Kostenfunktion). Welche Komplexität hat Ihr Algorithmus? Als Übung: Implementieren Sie den Algorithmus.

Lösung

```
1  for i=1 to n
2      do table[i,1]=scan relation i
3  for i=2 to n
4      do for j=1 to n-i+1
5          do best=null
6              for k=1 to i-1
7                  do test=table[j,k]  $\bowtie$  table[j+k,i-k]
8                      if cost(test) < cost(best)
9                          then best=test
10         table[j,i]=best
11 return table[1,n]
```

Komplexität: $O(n^3)$

Dieser Algorithmus baut ebenso auf DP auf! Das lässt sich daran erkennen, dass in Zeile 10 immer nur der beste Plan und nicht alle Pläne abgespeichert werden. Um den Algorithmus nicht auf dynamischen Programmieren basieren zu lassen, müssen nicht nur alle Pläne gespeichert werden, sondern diese auch durchprobiert werden. Soetwas lässt sich ähnlich wie beim transformationsbasiertem Ansatz am einfachsten über rekursive

Aufrufe bewerkstelligen. Anders hier ist die Speicherung der Pläne, die nicht wie bisher über Bitvektoren geschieht!