

Kürzester Weg

Entwicklung
eines Programms
zum
Gallenbacherbeispiel

Datenstruktur und Zugriffsfunktion

Kürzester Weg

- Wahl der Datenstruktur, Alternativen sind
 - Kantenliste
 - Knotenliste
- Hier gewählt: Kantenliste

Kürzester Weg

```
(define
  kanten
  ' ( (A B 69) (A D 36) (A M 36) (A N 22)
      (B A 69) (B D 64) (B Z 32) (B H 30) (B I 34) (B X 26)
      (C I 40) (C M 31) (C X 23)
      (D A 36) (D B 64) (D L 95) (D N 20)
      (E F 79) (E G 60) (E K 31) (E O 102)
      (F E 79) (F K 29) (F O 14) (F P 57)
      (G E 60) (G K 58) (G L 58) (G Y 30)
      (H B 30) (H I 65) (H K 31) (H L 106) (H P 34) (H Y 29)
      (I B 34) (I C 40) (I H 65) (I M 43) (I P 55)
      (K E 31) (K F 29) (K G 58) (K H 31) (K P 25) (K Y 20)
      (L D 95) (L G 58) (L H 106) (L Z 40)
      (M A 36) (M C 31) (M I 43) (M X 12)
      (N A 22) (N D 20) (N X 29) (N Z 30)
      (O E 102) (O F 14) (O P 91) (P F 57)
      (P H 34) (P I 55) (P K 25) (P O 91)
      (X B 26) (X C 23) (X M 12) (X N 29)
      (Y G 30) (Y H 29) (Y K 20) (Y Z 23)
      (Z B 32) (Z L 40) (Z N 30) (Z Y 23))
  )
```

Kürzester Weg

- Wahl der Datenstruktur, Alternativen sind
 - Kantenliste
 - Knotenliste
- Hier gewählt: Kantenliste
- Eine Aufteilung der Aufgabe in Teilaufgaben führt zu der hier behandelten Teilaufgabe:
Bestimmung der Nachfolgekanten zu einem Knoten

Kürzester Weg

Die Funktion zur Bestimmung aller Nachfolgekanten der aktuellen Stadt im Graphen muss die aktuelle Stadt und alle Kanten des Graphen kennen

Schritt 1: Funktionskopf

```
(define  
  (nachfolge-kanten aktuelle-stadt kanten)  
  'undefiniert)
```

Kürzester Weg

Funktion zur Bestimmung aller Nachfolgekanten der aktuellen Stadt im Graphen

Schritt 1*: Entscheidung für eine innere Funktion, um endrekursiv arbeiten zu können

```
(define
  (nachfolge-kanten aktuelle-stadt kanten)
  (define
    (intern aktuelle-stadt kanten akku)
    'undefiniert)
  (intern aktuelle-stadt kanten ' ()))
)
```

Kürzester Weg

- Wir betrachten nur diese innere Funktion weiter
- Die Rekursion erfolgt über alle Kanten des Graphen
- Ein Abbruchfall ist daher einzuplanen für eine leere Liste von Kanten

Kürzester Weg

Schritt 2: Abbruch der Rekursion

(reverse ist nicht zwingend, es sorgt dafür, dass die Reihenfolge erhalten bleibt)

```
(define
  (intern aktuelle-stadt kanten akku)
  (cond
    ((null? kanten) (reverse akku))
    (else
     'undefiniert)
  )
```


Kürzester Weg

Schritt 3: Erfolgsfall, am Kopf befindet sich eine Nachfolgekante

Wir können eine Hilfsfunktion einsetzen:

- Die Hilfsfunktion `ist-nachfolge-kante?` vergleicht `aktuelle-stadt` mit der ersten Stadt in der betrachteten Kante
- Gibt `#t` (wahr) oder `#f` (falsch) zurück

```
(define  
  (ist-nachfolge-kante? aktuelle-stadt kante)  
  (equal? aktuelle-stadt (first kante)))
```

Kürzester Weg

Schritt 3: Erfolgsfall, am Kopf befindet sich eine Nachfolgekante

```
(define
  (intern aktuelle-stadt kanten akku)
  (cond
    ((null? kanten) (reverse akku))
    ((ist-nachfolge-kante?
      aktuelle-stadt (first kanten))
     'kante-verwenden)
    (else 'undefiniert)
  )
)
```

Kürzester Weg

Schritt 4: Misserfolgsfall, am Kopf befindet sich keine Nachfolgekante

- Es gibt keine weiteren Fälle -> else-Fall

```
(define
  (nachfolge-kanten aktuelle-stadt kanten)
  (cond
    ((null? kanten) '())
    ((ist-nachfolge-kante?
      aktuelle-stadt (first kanten))
     'kante-verwenden)
    (else 'kante-verwerfen)
  )
)
```

Kürzester Weg

- Bisher ist "verwenden" und "verwerfen" nur eine Meldung, wird aber nicht umgesetzt.
- Ziel ist, eine Liste zu bekommen, die genau die Kanten enthält, die als zu "verwenden" gemeldet worden sind.
- Vorgehen: Arbeite
 - im Erfolgsfall weiter mit dem Rest der kanten und der Kante am Kopf des akku
 - sonst mit dem Rest der Kanten ohne die Kante dem akku hinzuzufügen

Kürzester Weg

Schritt 5: Rückgaben aufbauen

```
(define
  (intern aktuelle-stadt kanten akku)
  (cond
    ((null? kanten) '())
    ((ist-nachfolge-kante?
      aktuelle-stadt (first kanten))
     (intern
      aktuelle-stadt
      (rest kanten)
      (cons (first kanten) akku)))
    (else
     (intern aktuelle-stadt (rest kanten) akku)))
  )
)
```

Kürzester Weg

Die folgenden Aufrufe liefern die gewünschten Nachfolgekanten:

```
(nachfolge-kanten 'A kanten)
```

```
→ ((A B 69) (A D 36) (A M 36) (A N 22))
```

```
(nachfolge-kanten 'I kanten)
```

```
→ ((I B 34) (I C 40) (I H 65) (I M 43) (I P  
55))
```

Kürzester Weg

- Die endrekursive Version mit Aufrufhülle:

```
(define
  (nachfolge-kanten aktuelle-stadt kanten)
  (define
    (intern aktuelle-stadt kanten akku)
    (cond
      ((null? kanten) (reverse akku))
      ((ist-nachfolge-kante?
        aktuelle-stadt (first kanten))
       (intern
        aktuelle-stadt
        (rest kanten)
        (cons (first kanten) akku)))
      (else
       (intern aktuelle-stadt (rest kanten) akku))))
  (intern aktuelle-stadt kanten ' ()))
```