Prim Daten & Programm

Verfolgen der
Bearbeitung der Daten
im Programm zum
Algorithmus von Prim

Prim Daten & Programm

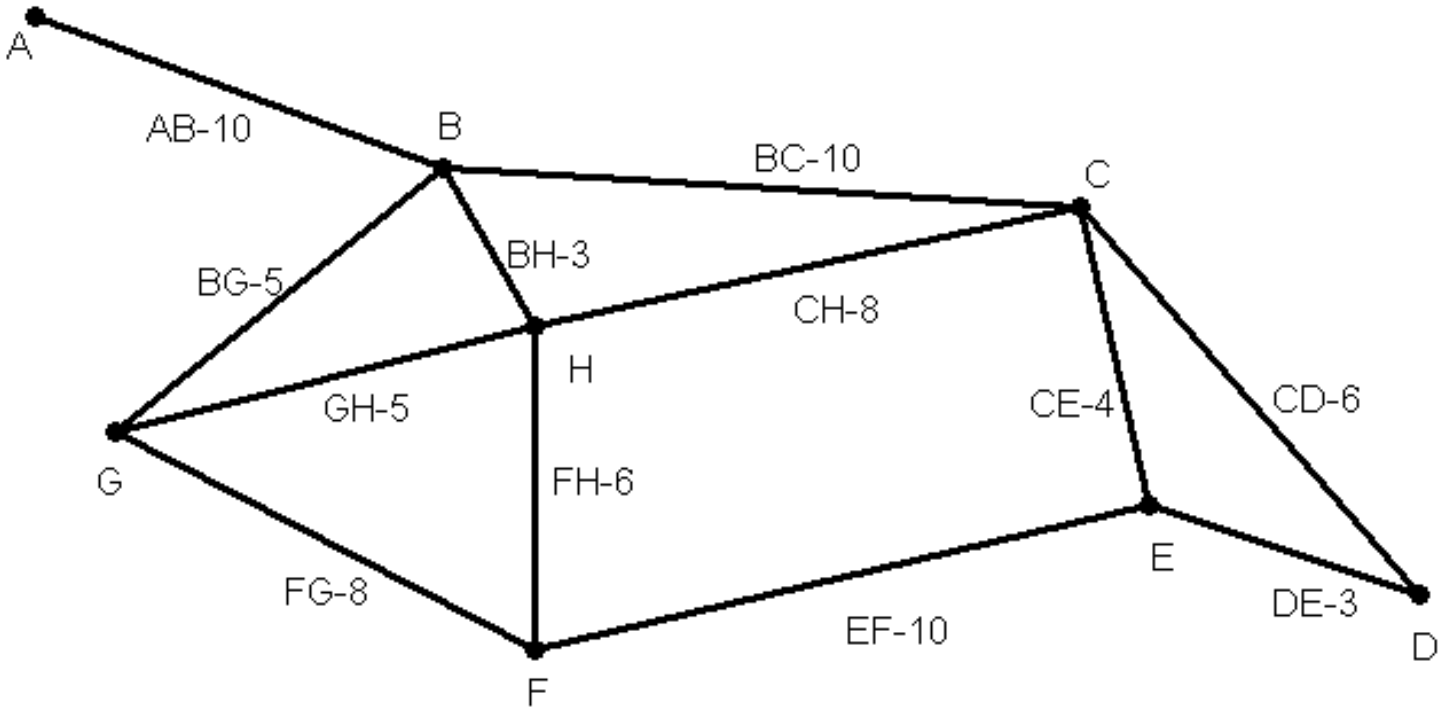
- Eingebunden über load werden

  (load "graph-kanten.scm")

  (load "prioritaetswarteschlange.scm")

  mit den Daten *kanten* (Kantenliste) und der Funktionen *alle-nachfolge-kanten* bzw. *fuege-alle-ein*, auf die zugegriffen wird.

Prim Daten & Programm



Prim Daten & Programm

```
(define
  kanten
  '((A B 10)
    (B A 10) (B C 10) (B G 5) (B H 3)
    (C B 10) (C D 6) (C E 4) (C H 8)
    (D C 6) (D E 3)
    (E C 4) (E D 3) (E F 10)
    (F E 10) (F G 8) (F H 6)
    (G B 5) (G F 8) (G H 5)
    (H B 3) (H C 8) (H F 6) (H G 5)
    )
  )
```

Prim Daten & Programm

- Funktionsköpfe (mit interner Funktion):

```
(prim start-knoten kanten) =>
(prim-intern besucht anzahl kuerzeste priows)
```

- "Beginne mit einem Baum, der allein aus einem (beliebigen) Knoten besteht."
  - besucht ← `'(D)`
  - *`anzahl`* wird vorher bestimmt
  - *`kuerzeste`* ist noch leer
  - In *`priows`* werden beim Aufruf der Hüllfunktion die Nachfolgekanten von D eingefügt

Prim Daten & Programm

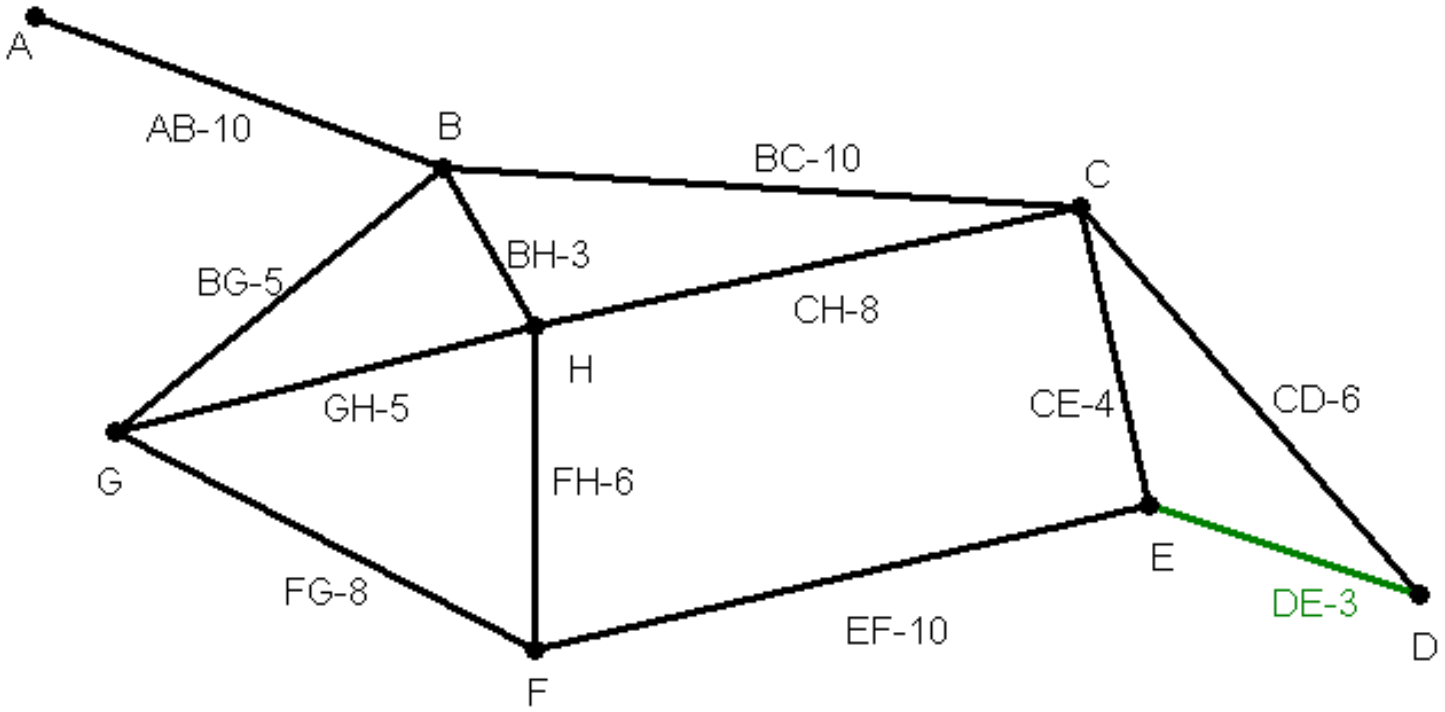
```
(prim-intern
  (list start-knoten)
  (anzahl-knoten kanten) ; Hilfsfunktion
  '()
  (fuege-alle-ein          ; Start-PrioWS
   (nachfolge-kanten start-knoten kanten)
   vor?
   '()))
```

Prim Daten & Programm

Schritt 1

- Da in *`prim`* die Tests auf *alle Knoten enthalten*, *leere PrioWS* und *Zyklus* für die erste Kante in der PrioWS alle nicht erfüllt sind, wird else bearbeitet:
- *`prim`* ruft sich selbst auf mit
  - dem freien Knoten hinzugefügt zu *`besucht`*
  - der anzahl
  - der ersten Kante hinzugefügt zu *`kuerzeste`*
  - den Nachfolgekanten des freien Knoten eingefügt in die PrioWS

Prim Daten & Programm



Prim Daten & Programm

```
besucht : '(E D)

kuerzeste : '((D E 3))

priows :
  '((E D 3) (E C 4) (D C 6) (E F 10))
```

Prim Daten & Programm

```
 ((define (zyklus? kante besucht)
    (member (second kante) besucht))
;;; in prim:
 ((zyklus? (first priows) besucht)
  (prim
    besucht
    anzahl
    kuerzeste
    (rest priows)))
```

Prim Daten & Programm

- Da in *prim* nun der Test auf einen Zyklus für die erste Kante in der PrioWS erfüllt ist (Rückkante), wird *prim* erneut, aber ohne dieses erste Element aufgerufen.

```
priows :
  '((E C 4) (D C 6) (E F 10))
```

- Nun wird die erste Kante akzeptiert und wie beim vorigen else-Schritt aufgenommen.

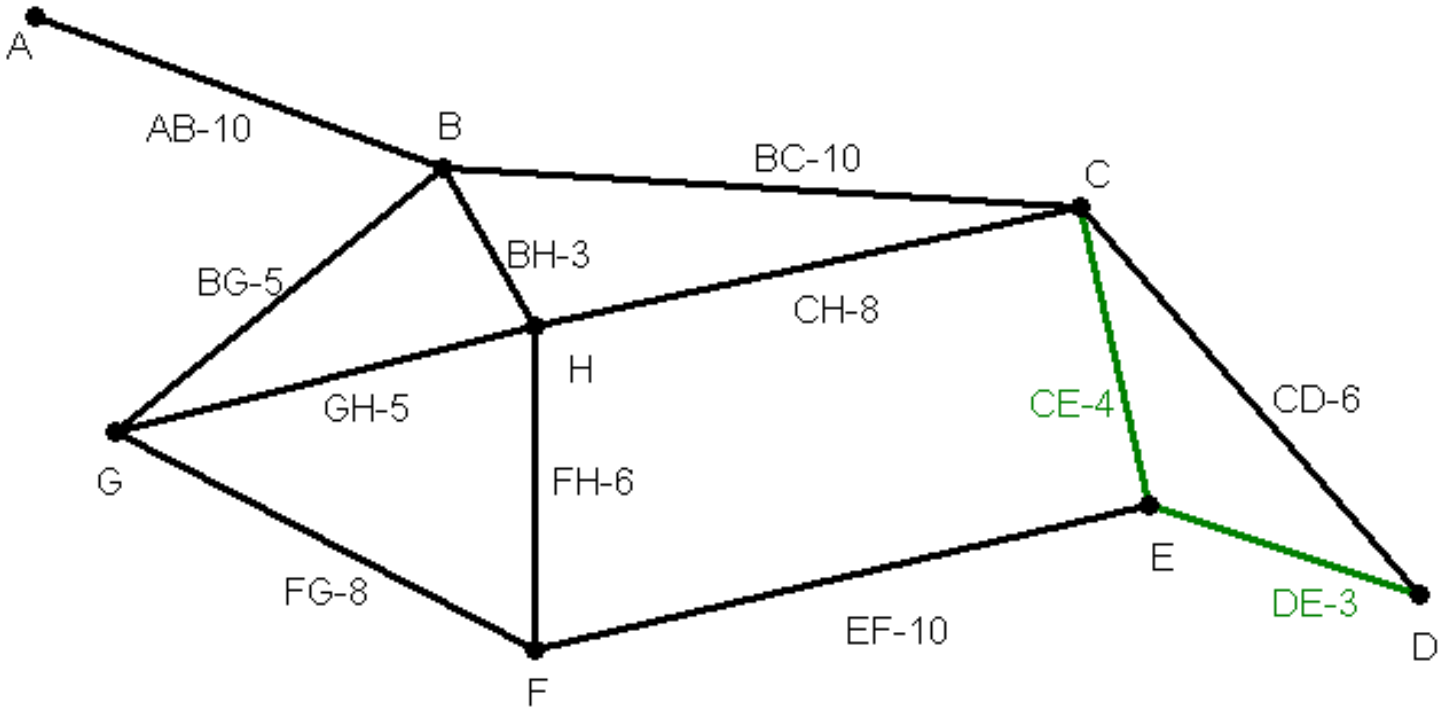
Prim Daten & Programm

```
besucht : '(C E D)

kuerzeste : '((E C 4) (D E 3))

priows :
  '((C E 4) (D C 6) (C D 6) (C H 8)
    (E F 10) (C B 10))
```

Prim Daten & Programm



Prim Daten & Programm

- Rückkante entfernen

```
besucht : '(C E D)

kuerzeste : '((E C 4) (D E 3))

priows :
  '((D C 6) (C D 6) (C H 8) (E F 10)
    (C B 10))
```

Prim Daten & Programm

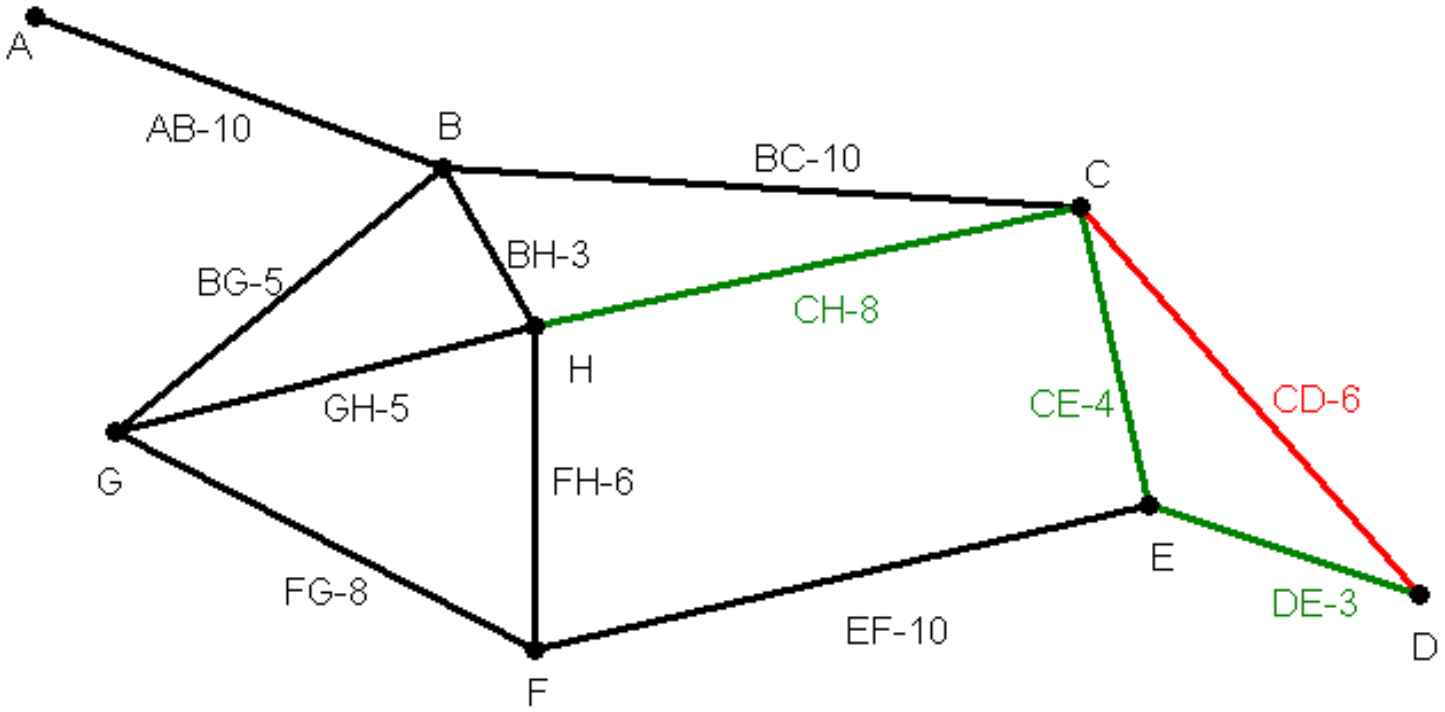
- Kante von D zu C als Zyklus erkennen und ebenso wie ihre Rückkante entfernen

```
besucht : '(C E D)

kuerzeste : '((E C 4) (D E 3))

priows :
 '((C H 8) (E F 10) (C B 10))
```

Prim Daten & Programm



Prim Daten & Programm

- Kante von C zu H entwickeln

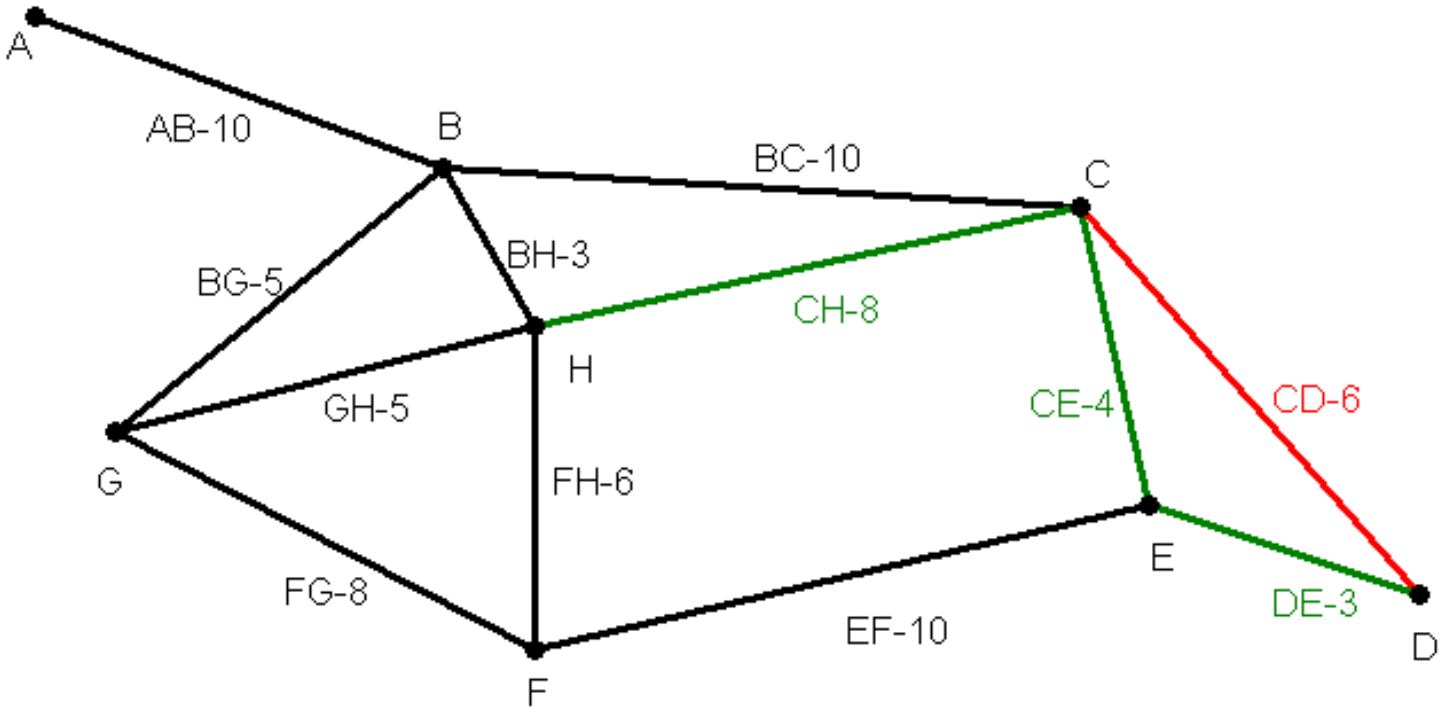
```
besucht : '(H C E D)

kuerzeste : '((C H 8) (E C 4) (D E 3))

priows :
  '((H B 3) (H G 5) (H F 6) (H C 8)
    (E F 10) (C B 10))
```

- *Ein Hinweis: Die Rückkante steht nun nicht mehr vorn in der PrioWS, kann also erst später entfernt werden.*

Prim Daten & Programm



Prim Daten & Programm

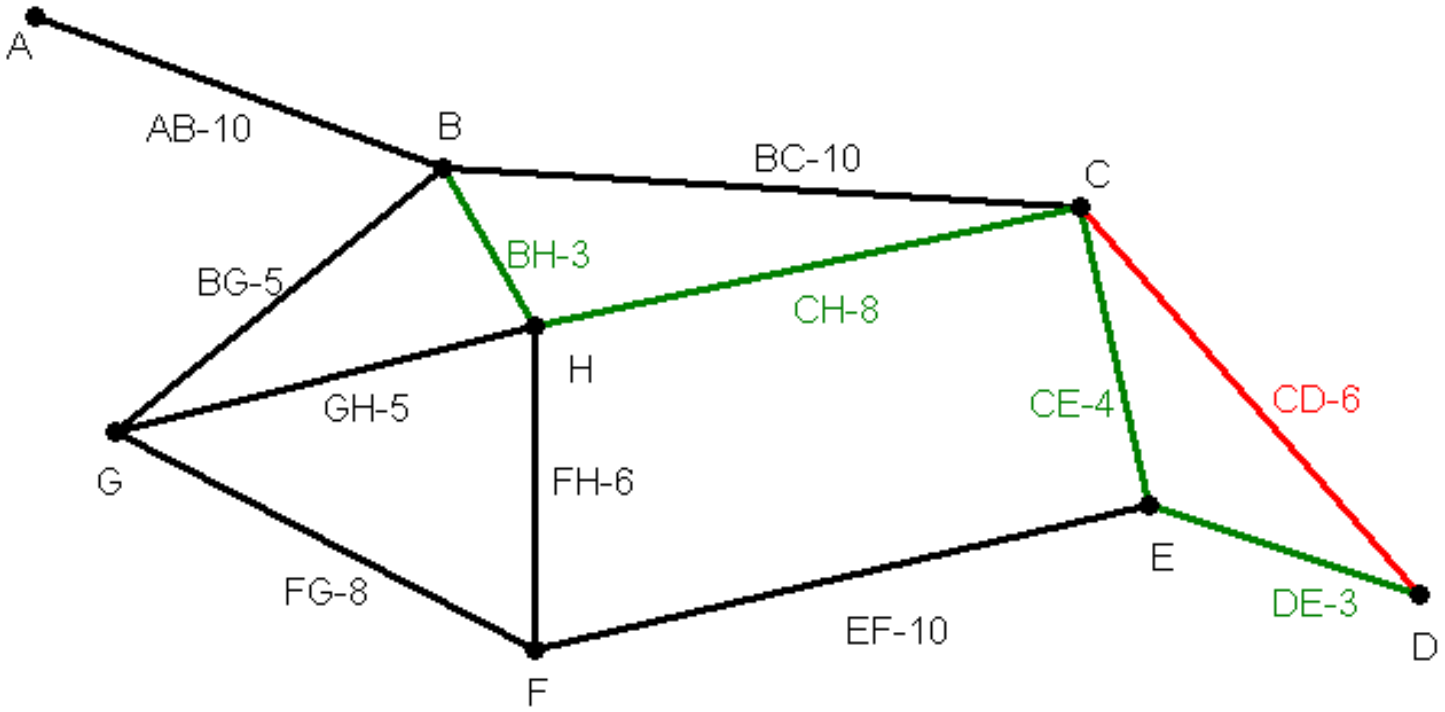
- Kante von H zu B entwickeln

```
besucht : '(B H C E D)

kuerzeste : '((H B 3) (C H 8) (E C 4)
        (D E 3))

priows :
 '((B H 3) (H G 5) (B G 5) (H F 6)
    (H C 8) (E F 10) (C B 10) (B A 10)
    (B C 10))
```

Prim Daten & Programm



Prim Daten & Programm

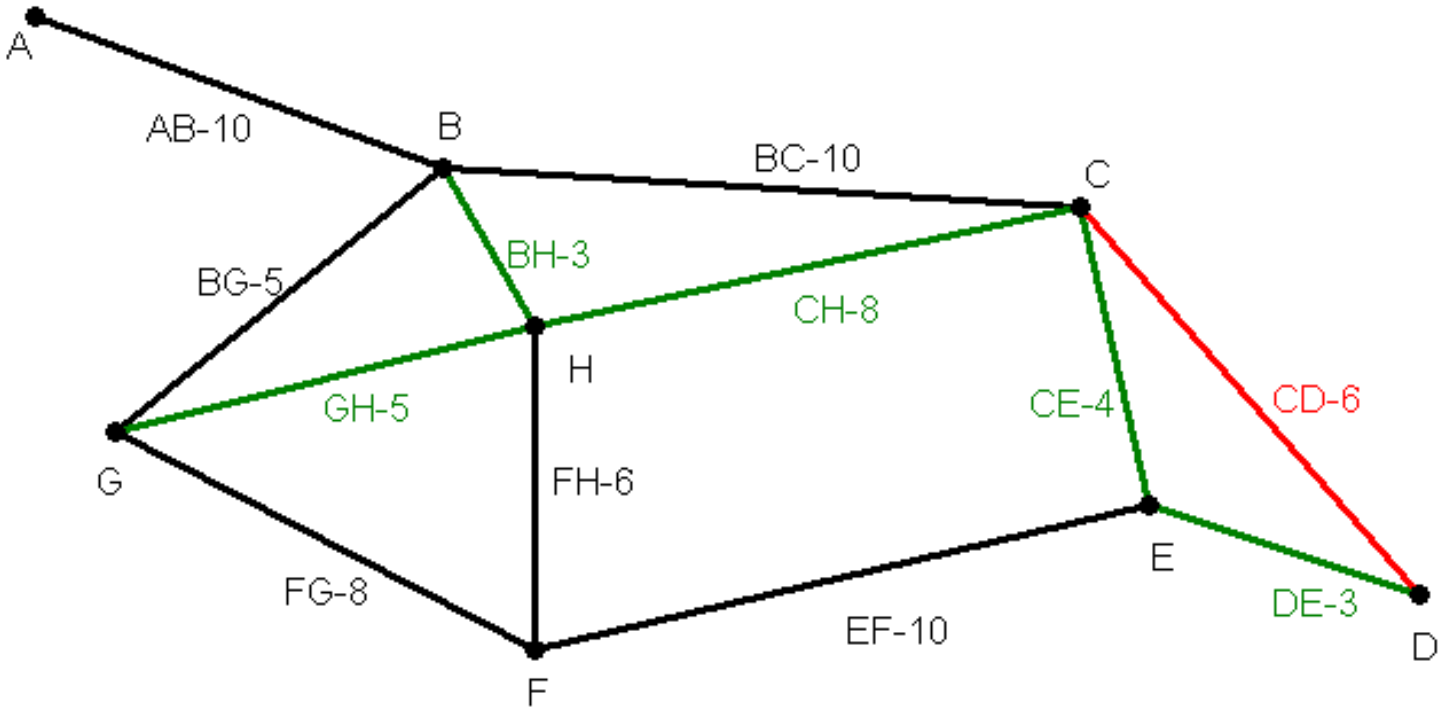
- Kante von H zu G entwickeln

```
besucht : '(G B H C E D)

kuerzeste : '((H G 5) (H B 3) (C H 8)
        (E C 4) (D E 3))

priows :
 '((B G 5) (G B 5) (G H 5) (H F 6)
   (H C 8) (G F 8) (E F 10) (C B 10)
   (B A 10) (B C 10))
```

Prim Daten & Programm



Prim Daten & Programm

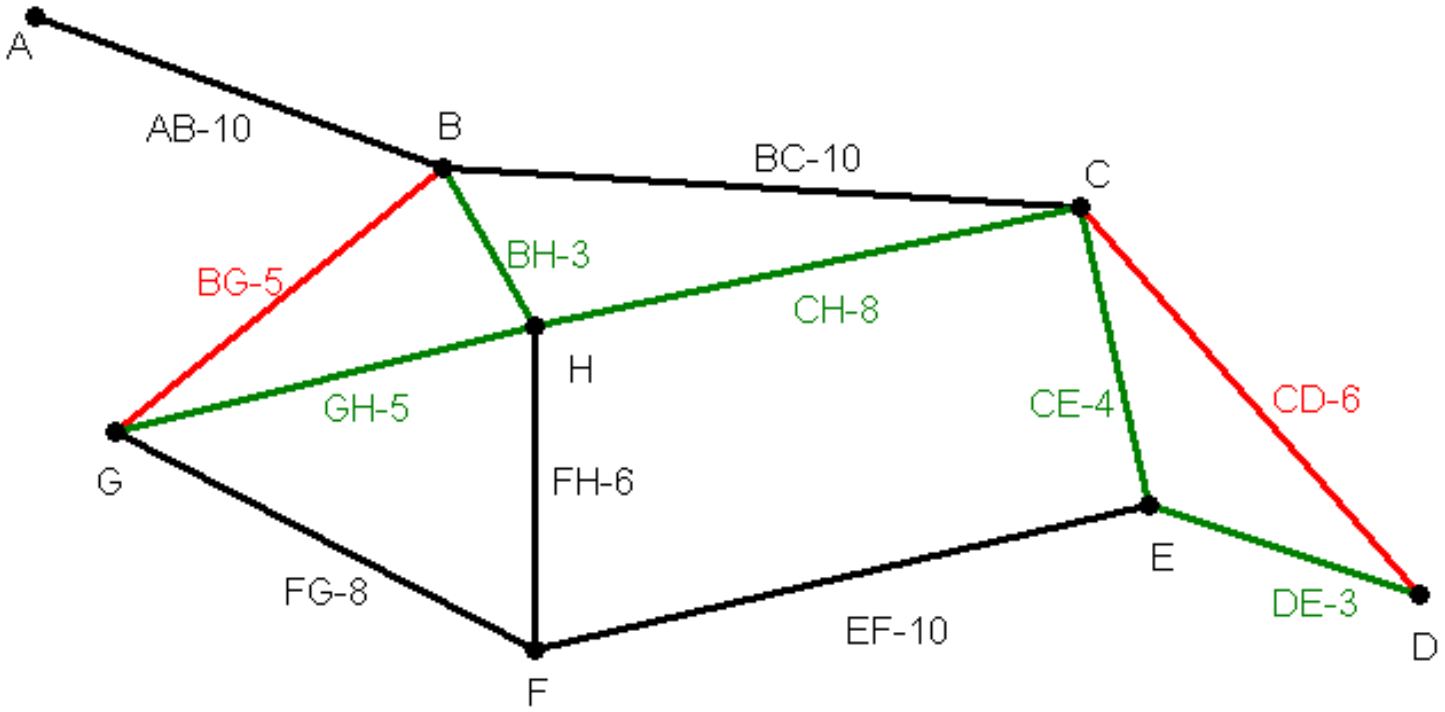
- Zyklus B G entfernen & Rückkanten

```
besucht : '(G B H C E D)

kuerzeste : '((H G 5) (H B 3) (C H 8)
        (E C 4) (D E 3))

priows :
 '((H F 6) (H C 8) (G F 8) (E F 10)
   (C B 10) (B A 10) (B C 10))
```

Prim Daten & Programm



Prim Daten & Programm

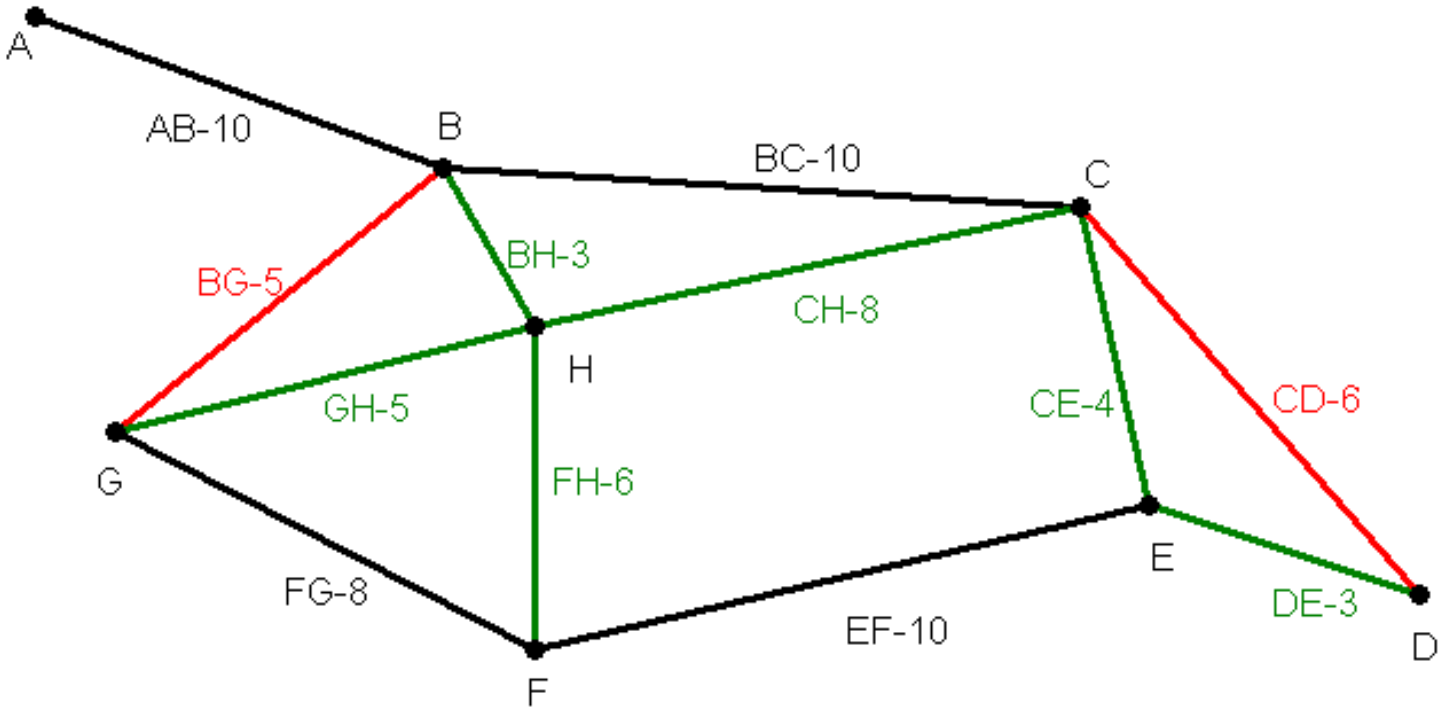
- Kante von H zu F entwickeln

```
besucht : '(F G B H C E D)

kuerzeste : '((H F 6) (H G 5) (H B 3)
        (C H 8) (E C 4) (D E 3))

priows :
 '((H C 8) (G F 8) (E F 10)
   (C B 10) (B A 10) (B C 10))
```

Prim Daten & Programm



Prim Daten & Programm

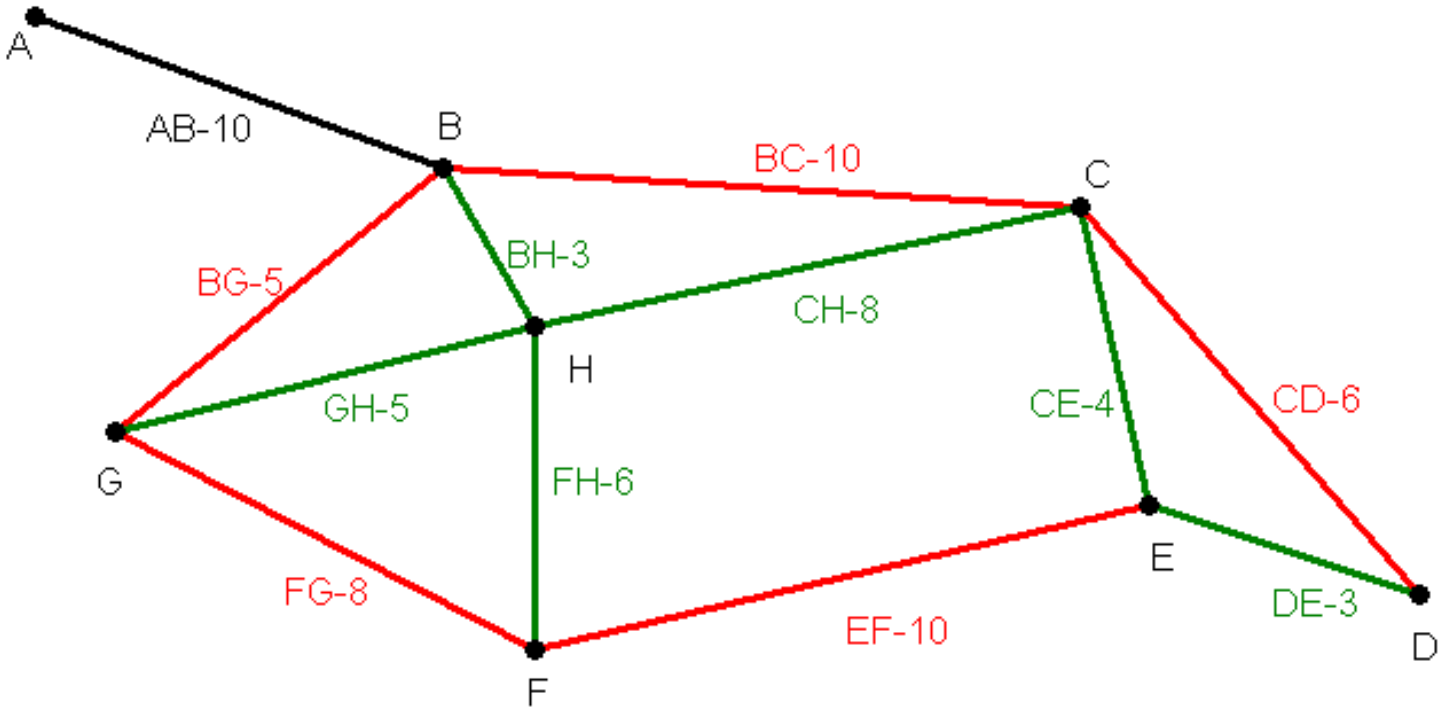
- Rückkanten und Zyklen entfernen

```
besucht : '(F G B H C E D)

kuerzeste : '((H F 6) (H G 5) (H B 3)
        (C H 8) (E C 4) (D E 3))

priows :
 '((B A 10) (B C 10))
```

Prim Daten & Programm



Prim Daten & Programm

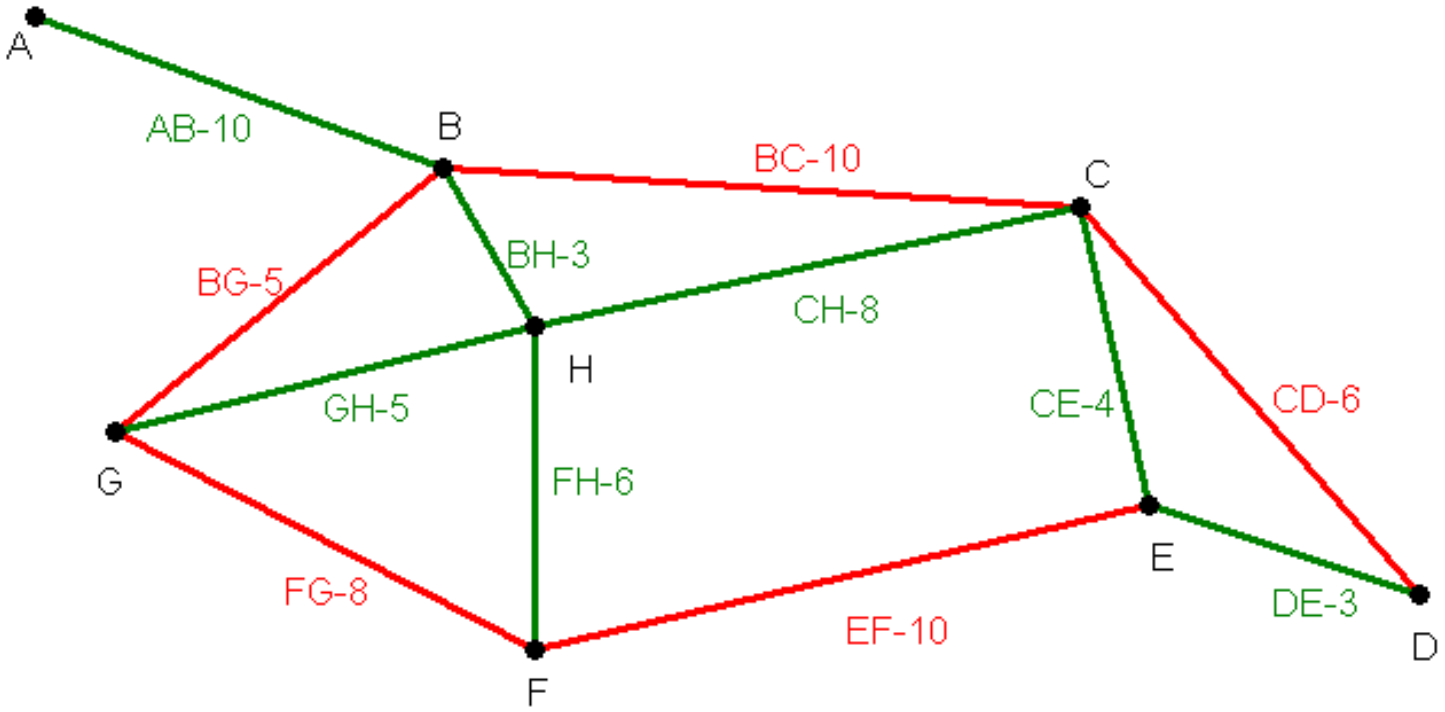
- Kante von B zu A entwickeln

```
besucht : '(A F G B H C E D)

kuerzeste : '((B A 10) (H F 6) (H G 5)
        (H B 3) (C H 8) (E C 4) (D E 3))

priows :
 '((B C 10))
```

Prim Daten & Programm



Prim Daten & Programm

- Alle Knoten enthalten!

```
((equal? (length besucht) anzahl)
 kuerzeste)

besucht : '(A F G B H C E D)
kuerzeste : '((B A 10) (H F 6) (H G 5)
        (H B 3) (C H 8) (E C 4) (D E 3))
priows :
 '((B C 10))
```

- *kuerzeste* als Lösung ausgeben!