Algorithmus von Kruskal

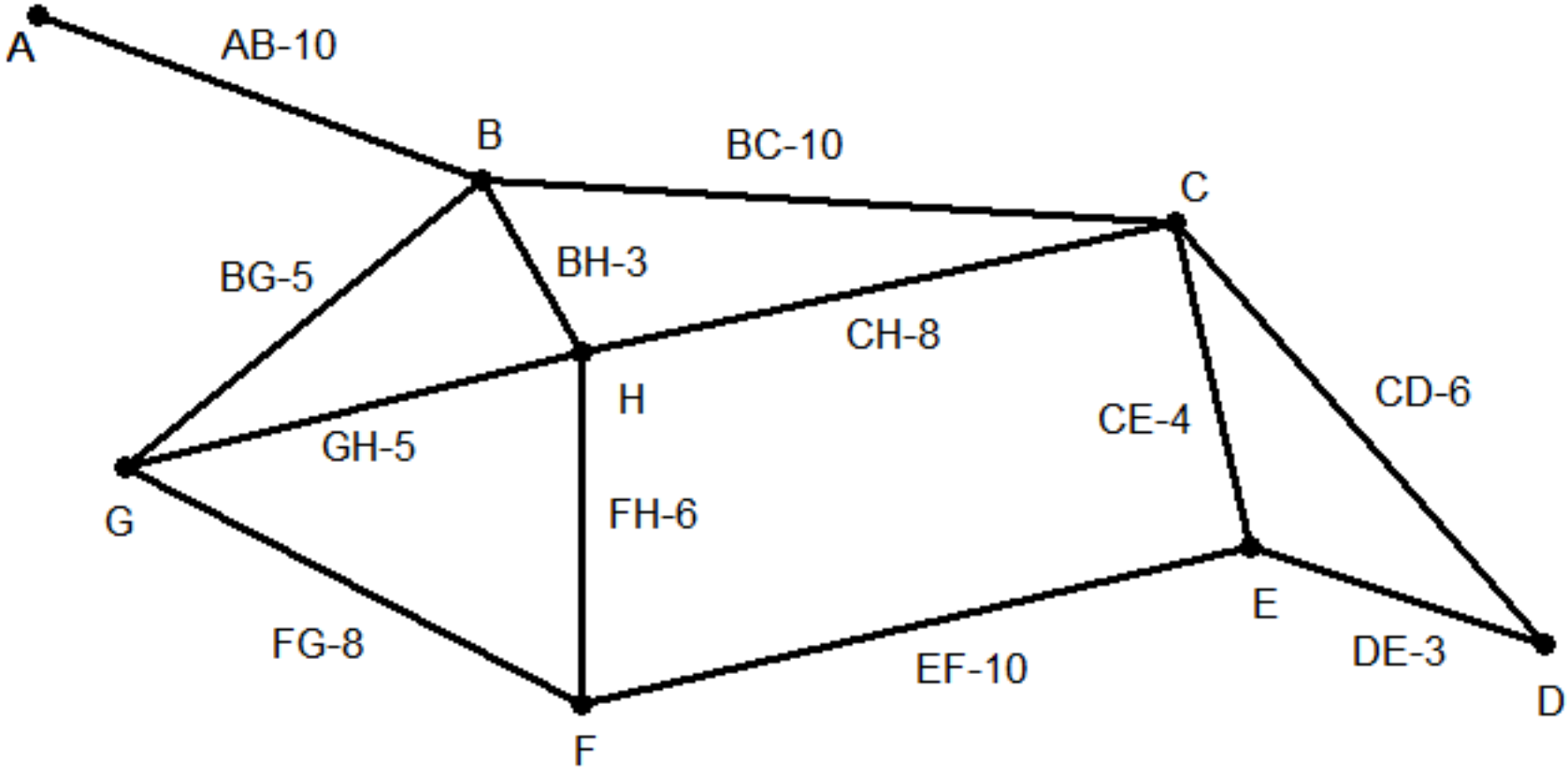
Algorithmus von Kruskal

ein Algorithmus zur Bestimmung eines minimalen aufspannenden Baums

(minimal spanning tree)

Algorithmus von Kruskal

Ein Graph



Algorithmus von Kruskal

Beschreibung des Algorithmus

- Ordne alle Kanten nach ihren Kosten.
- Beginne mit einer Liste der Teilbäume, die von allen Knoten jeweils genau einen enthalten.
- Füge dann jeweils immer die nächste Kante mit den geringsten Kosten hinzu, die genau zwei Teilbäume verbindet.
- Lasse Kanten weg, die Zyklen bilden.
- Beende die Suche, wenn es nur noch einen Teilbaum gibt.

Voraussetzung: Der Ausgangsgraph muss zusammenhängend sein.

Algorithmus von Kruskal

Aufgaben:

- Vor dem Start: Ordnen der Kanten nach ihren Bewertungen
- Prüfen, ob beide Knoten einer neuen Kante im selben Teilbaum enthalten sind (→ *Zyklus*)
- Verbinden zweier Teilbäume mit der neuen Kante
- Die eigentliche Steuerung des Algorithmus

Algorithmus von Kruskal

Ordnen der Kanten

wird durch das Einbinden
der Funktionen für die
Prioritätswarteschlange
realisiert

*(PrioWS: Hier wird allein die Sortierung genutzt,
sie wird bei Kruskal also nur einmalig aufgebaut)*

Algorithmus von Kruskal

Hinweis

- Die ***Prioritätswarteschlange*** selbst wird in der funktionalen Modellierung allein vom zugreifenden Programm als Liste gehalten.
- Die Eigenschaft dieser Liste, Prioritäts-warteschlange zu sein, ergibt sich aus den Zugriffsfunktionen, die aber identisch sind mit den Funktionen für die Aufgabe ***„Sortieren durch Einfügen“.***

Algorithmus von Kruskal

Die Prioritätswarteschlange benötigt das Prädikat zum Einfügen

```
(define
  (vor? kante-1 kante-2)
  (< (third kante-1) (third kante-2)))
```

Die Kanten sind also gespeichert in der Form:
(<Knoten-1> <Knoten-2> <Bewertung>)

Algorithmus von Kruskal

Erzeugen der Teilbäume

Algorithmus von Kruskal

Datenstruktur der Teilbäume
(es gäbe auch Alternativen)

- Eine Liste von zwei Teillisten aus
  - den enthaltenen Knoten
  - den enthaltenen Kanten
- Anfangszustand zum Beispielgraph:

(erzeuge-teilbaeume kanten) →

```
(((H) ()) ((G) ()) ((F) ()) ((E) ())
 ((D) ()) ((C) ()) ((B) ()) ((A) ()))
```

Algorithmus von Kruskal

```
(define
  (erzeuge-teilbaeume kanten)
  (define
    (erzeuge-intern knoten teilbaeume kanten)
    ...
      ))
  (erzeuge-intern '() '() kanten)
  )
```

Algorithmus von Kruskal

```
(define
 (erzeuge-intern knoten teilbaeume kanten)
 (cond
  ((null? kanten)   ; alle bearbeitet
   teilbaeume)
  ((member (first (first kanten)) knoten)
   (erzeuge-intern knoten teilbaeume (rest kanten)))
  (else
   (erzeuge-intern
    (cons (first (first kanten)) knoten)
    (cons
     (list (list (first (first kanten))) '())
     teilbaeume)
    (rest kanten)))
  ))
```

Algorithmus von Kruskal

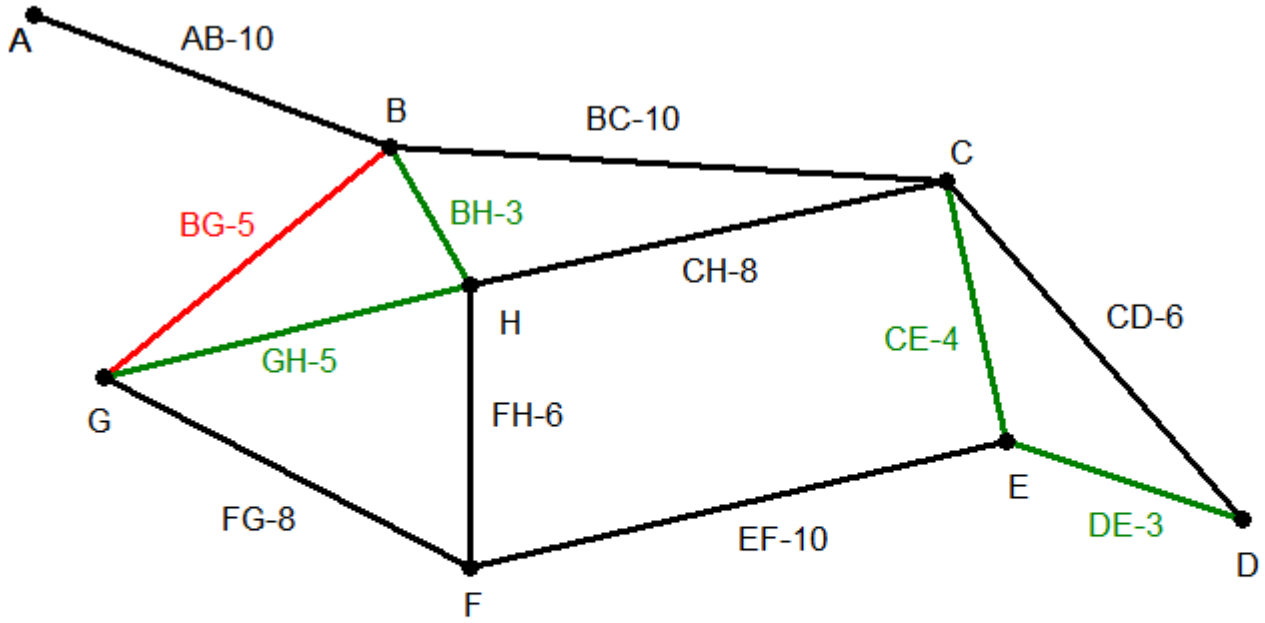
```
(define
 (erzeuge-intern knoten teilbaeume kanten)
 (cond
  ((null? kanten)
   teilbaeume)
  (((member (first (first kanten)) knoten) ; hab ich
   (erzeuge-intern knoten teilbaeume (rest kanten)))
  (else
   (erzeuge-intern
    (cons (first (first kanten)) knoten)
    (cons
     (list (list (first (first kanten))) '())
     teilbaeume)
    (rest kanten)))
  ))
```

Algorithmus von Kruskal

```
(define
 (erzeuge-intern knoten teilbaeume kanten)
 (cond
  ((null? kanten)
   teilbaeume)
  ((member (first (first kanten)) knoten)
   (erzeuge-intern knoten teilbaeume (rest kanten)))
  (else                ; sonst aufnehmen
   (erzeuge-intern
    (cons (first (first kanten)) knoten)
    (cons
     (list (list (first (first kanten))) '())
     teilbaeume)
    (rest kanten)))
  ))
```

Algorithmus von Kruskal

Prüfen,
ob beide Knoten einer Kante
im selben Teilbaum enthalten sind
(→ Zyklus)



Algorithmus von Kruskal

```
(define
  (zyklus? kante teilbaeume)
  (cond
    ((null? teilbaeume)
     #f)
    ((beide-knoten-im-teilbaum?
       kante (first teilbaeume))
     #t)
    (else
     (zyklus?
       kante
       (rest teilbaeume)))))
```

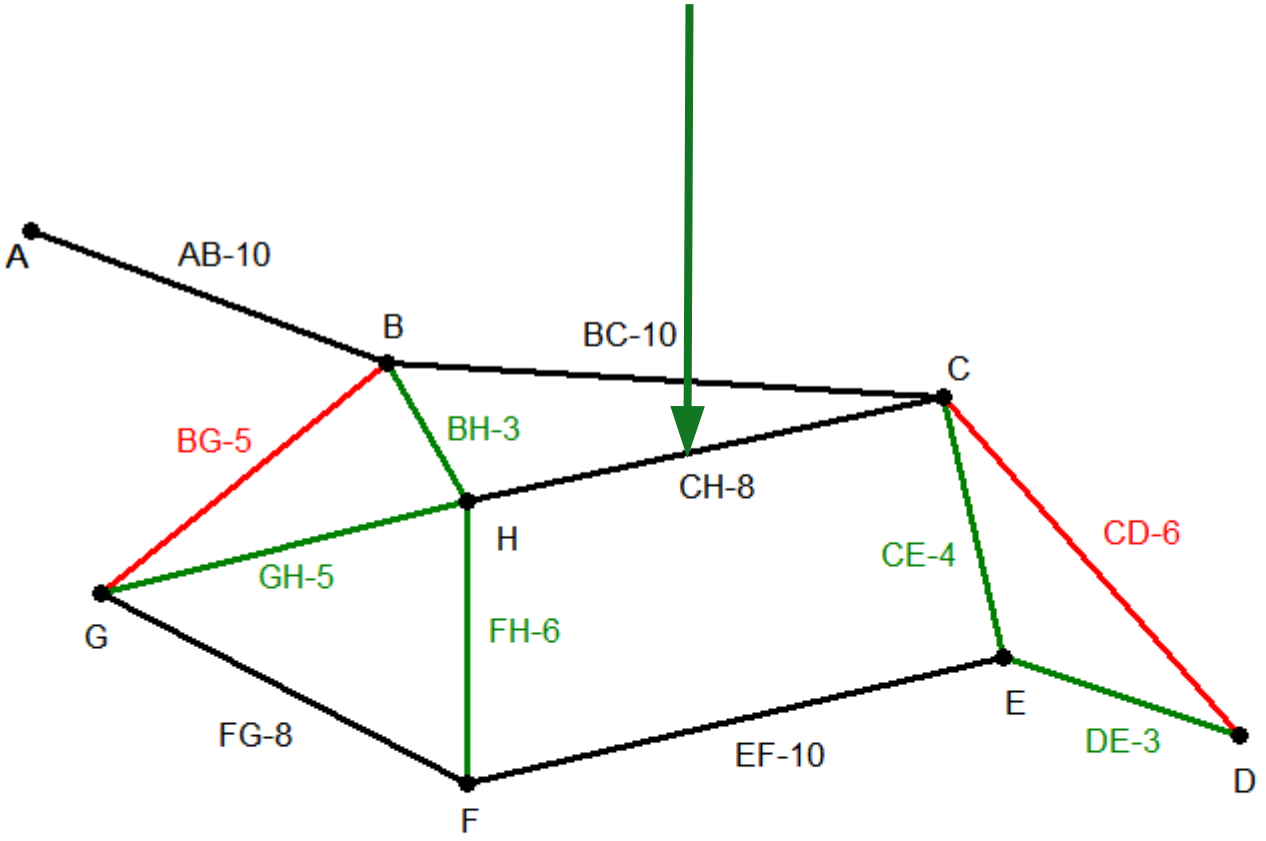
Algorithmus von Kruskal

```
(define
  (knoten-im-teilbaum? knoten teilbaum)
  (member knoten (first teilbaum)))

(define
  (beide-knoten-im-teilbaum? kante teilbaum)
  (and
   (knoten-im-teilbaum?
     (first kante) teilbaum)
   (knoten-im-teilbaum?
     (second kante) teilbaum)))
```

Algorithmus von Kruskal

Verbinden zweier Teilbäume



Algorithmus von Kruskal

Verbinden zweier Teilbäume

- Bei dieser Funktion werden zwei Teilbäume durch die neue Kante verbunden.
- Um die Bearbeitung zu erleichtern, werden die Teilbäume so umsortiert verwendet, dass die beiden zu verbindenden Teilbäume vorn in der Liste stehen.

Algorithmus von Kruskal

```
(define
  (beide-nach-vorn kante teilbaeume)
  (define
    (beide-intern beide bearbeitet teilbaeume)
    ...
    ))
  (beide-intern '() '() teilbaeume))
```

Algorithmus von Kruskal

```
(define
 (beide-intern beide bearbeitet teilbaeume)
 (cond
  ((= 2 (length beide))          ; gefunden
   (append beide bearbeitet teilbaeume))

  ...

  ))
```

Algorithmus von Kruskal

```
(define
 (beide-intern beide bearbeitet teilbaeume)
 (cond
  ...
  ((knoten-im-teilbaum?          ; einer
     (first kante) (first teilbaeume))
   (beide-intern
    (cons (first teilbaeume) beide)
    bearbeitet
    (rest teilbaeume)))

    ...

  ))
```

Algorithmus von Kruskal

```
(define
 (beide-intern beide bearbeitet teilbaeume)
 (cond

   ...

  ((knoten-im-teilbaum?       ; der andere
     (second kante) (first teilbaeume))
   (beide-intern
    (cons (first teilbaeume) beide)
    bearbeitet
    (rest teilbaeume)))

   …

  ))
```

Algorithmus von Kruskal

```
(define
 (beide-intern beide bearbeitet teilbaeume)
 (cond

   ...

  (else                    ; weiter suchen
    (beide-intern
     beide
     (cons (first teilbaeume) bearbeitet)
     (rest teilbaeume)))
  ))
```

Algorithmus von Kruskal

```
(define
 (verbinde-teilbaeume-mit-kante
                                  kante teilbaeume)
 (define
  (verbinde-intern kante teilbaeume)
  (cons
   (verbinde-zwei
    kante
    (first teilbaeume)
    (second teilbaeume))
    (rest (rest teilbaeume))))

 (verbinde-intern
   kante (beide-nach-vorn kante teilbaeume)))
```

Algorithmus von Kruskal

```
(define
  (verbinde-zwei kante ersten zweiten)
  (list
   ; knoten:
   (append (first ersten) (first zweiten))
   ;kanten:
   (cons kante
    (append
     (second ersten)
     (second zweiten)))))
```

Algorithmus von Kruskal

Die Funktion **`kruskal`** wird aufgeteilt

- in eine Aufrufhülle und
- die eigentliche Schrittfunktion

Algorithmus von Kruskal

```
(define
  (kruskal kanten)
  (define
    (kruskal-intern kanten teilbaeume)

    ...

  (kruskal-intern
   (fuege-alle-ein kanten vor? '())
   (erzeuge-teilbaeume kanten))
  )
```

Algorithmus von Kruskal

```
(define
    (kruskal-intern kanten teilbaeume)
    (cond
      ((null? (rest teilbaeume))   ; fertig!
       (first teilbaeume))
      ((null? kanten)
       "Der Graph ist nicht zusammenhaengend!")
      ((zyklus? (first kanten) teilbaeume)
       (kruskal-intern
         (rest kanten) teilbaeume))
      (else
       (kruskal-intern
        (rest kanten)
        (verbinde-teilbaeume-mit-kante
          (first kanten) teilbaeume)))
      ))
```

Algorithmus von Kruskal

```
(define
    (kruskal-intern kanten teilbaeume)
    (cond
      ((null? (rest teilbaeume))
       (first teilbaeume))
      ((null? kanten)                    ; Fehler !
       "Der Graph ist nicht zusammenhaengend!")
      ((zyklus? (first kanten) teilbaeume)
       (kruskal-intern
         (rest kanten) teilbaeume))
      (else
       (kruskal-intern
        (rest kanten)
        (verbinde-teilbaeume-mit-kante
          (first kanten) teilbaeume)))
      ))
```

Algorithmus von Kruskal

```
(define
    (kruskal-intern kanten teilbaeume)
    (cond
      ((null? (rest teilbaeume))
       (first teilbaeume))
      ((null? kanten)
       "Der Graph ist nicht zusammenhaengend!")
      ((zyklus? (first kanten) teilbaeume)
       (kruskal-intern  ; weglassen
         (rest kanten) teilbaeume))
      (else
       (kruskal-intern
        (rest kanten)
        (verbinde-teilbaeume-mit-kante
          (first kanten) teilbaeume)))
      ))
```

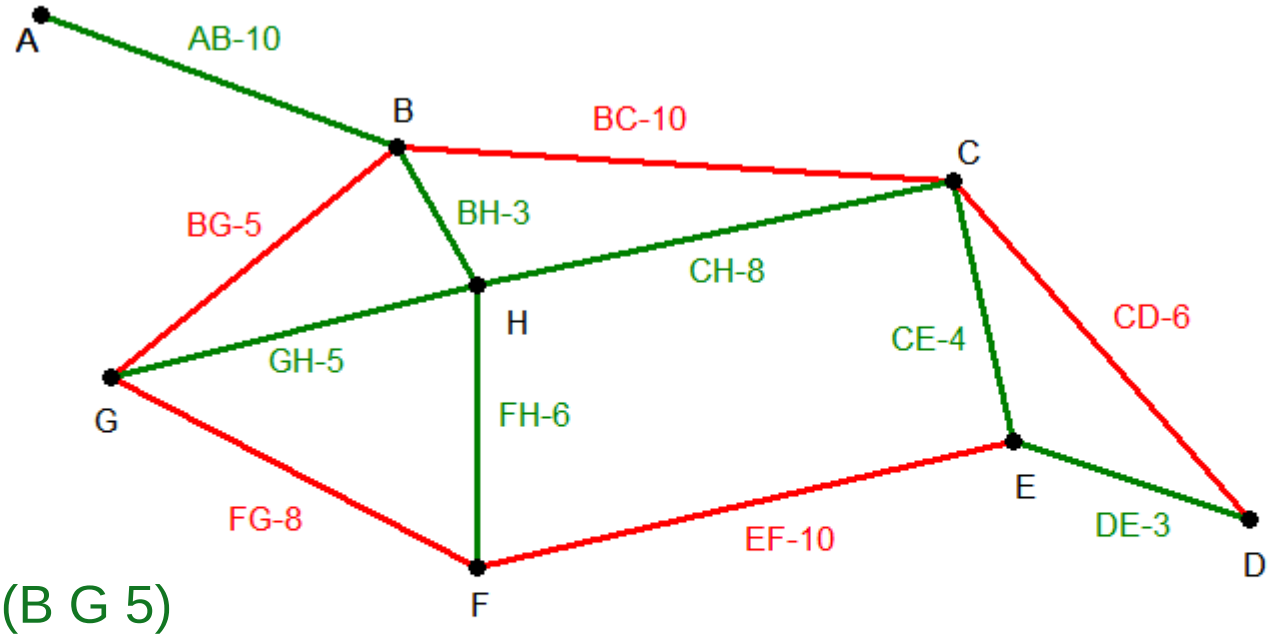
Algorithmus von Kruskal

```
(define
    (kruskal-intern kanten teilbaeume)
    (cond
      ((null? (rest teilbaeume))
       (first teilbaeume))
      ((null? kanten)
       "Der Graph ist nicht zusammenhaengend!")
      ((zyklus? (first kanten) teilbaeume)
       (kruskal-intern
         (rest kanten) teilbaeume))
      (else                              ; weiter
       (kruskal-intern
        (rest kanten)
        (verbinde-teilbaeume-mit-kante
          (first kanten) teilbaeume)))
      ))
```

Algorithmus von Kruskal

Ein Aufruf für den angegebenen Graphen ergibt

```
((A C E D F G B H) ; die Knoten
  ((A B 10) (C H 8) (C E 4) (D E 3)
   (F H 6) (G H 5) (B H 3)) ; die Kanten *)
)
```



*) alternativ (B G 5)