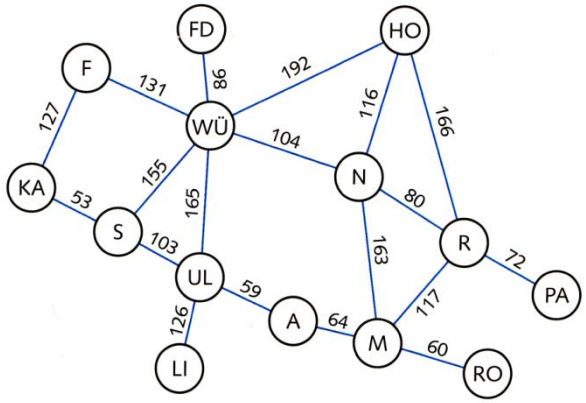
# *Breitensuche*

Die Breitensuche besucht von einem Startknoten aus systematisch in konzentrischen Bereichen um den Startknoten alle (erreichbaren) Knoten eines Graphen.

| **Prinzip der Breitensuche** | | |
|---|---|---|
| | | 1 |
| 1<br>2 | 1<br>2 3 | 1<br>2 3 4 |
| 1<br>2 3 4<br>5 | 1<br>2 3 4<br>5 6 | 1<br>2 3 4<br>5 6 7 |
| 1<br>2 3 4<br>5 6 7 8 | 1<br>2 3 4<br>5 6 7 8<br>9 | 1<br>2 3 4<br>5 6 7 8<br>9 10 |

Es ist folgender Graph gegeben:




Startknoten = N

Wenn mehrere Wege möglich sind, dann wird zuerst der Ort mit der kleinsten Gewichtung besucht.

**Warteliste (Schlange):**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| N | | | | | | | |
| N | R | | | | | | |
| N | R | WÜ | | | | | |
| N | R | WÜ | HO | | | | |
| N | R | WÜ | HO | M | | | |
| R | WÜ | HO | M | | | | |
| R | WÜ | HO | M | PA | | | |
| WÜ | HO | M | PA | | | | |
| WÜ | HO | M | PA | FD | | | |
| WÜ | HO | M | PA | FD | F | | |
| WÜ | HO | M | PA | FD | F | S | |
| WÜ | HO | M | PA | FD | F | S | UL |
| HO | M | PA | FD | F | S | UL | |
| M | PA | FD | F | S | UL | | |
| M | PA | FD | F | S | UL | RO | |
| M | PA | FD | F | S | UL | RO | A |
| PA | FD | F | S | UL | RO | A | |
| FD | F | S | UL | RO | A | | |
| F | S | UL | RO | A | | | |
| F | S | UL | RO | A | KA | | |
| S | UL | RO | A | KA | | | |
| UL | RO | A | KA | | | | |
| UL | RO | A | KA | LI | | | |
| RO | A | KA | LI | | | | |
| A | KA | LI | | | | | |
| KA | LI | | | | | | |
| LI | | | | | | | |

# *Algorithmus für die Breitensuche*

public void ***breitensuche***(String ***startKnoten***)

| | |
|---|---|
| int startNummer = getKnotenNummer(startKnoten); | *(bestimme die Knotennummer des Startknotens und speichere diese unter startNummer ab)* |
| int aktuellerKnoten = startNummer; | *(setze den aktuellen Knoten auf die startNummer)* |
| anzahlWarteliste = 0; | *(setze die Anzahl der Knoten in der Warteliste auf 0)* |
| anzahlBesucht = 0; | *(setze die Anzahl der besuchten Knoten auf 0)* |
| setze die Warteliste zurück | *(alle Indixe im Feld warteliste erhalten den Wert -1)* |
| setze die Besuchtliste zurück | *(alle Indixe im Feld besucht erhalten den Wert false)* |
| warteliste[0] = aktuellerKnoten; | *(der aktuelle Knoten (= Startknoten) wird in die leere Warteliste an erster Stelle eingefügt)* |
| anzahlWarteliste++; | *(die Anzahl der in der Warteliste eingefügten Knoten wird um 1 erhöht)* |
| wiederhole solange *die Warteliste nicht leer ist* | |
| zähle i von 0 bis anzahlKnoten - 1 | |
| matrix[aktuellerKnoten][i] > 0 *(es existiert eine Kante zwischen aktuellerKnoten und i)* **und** der Knoten mit der Knotennummer i wurde noch nicht besucht **und** der Knoten mit der Knotennummer i ist noch nicht in der Warteliste | |
| **wahr** | **falsch** |
| warteliste[anzahlWarteliste] = i;<br>*(Knoten mit Knotennummer i wird in die Warteliste hinten eingefügt)* | |
| anzahlWarteliste++; | |
| besucht[aktuellerKnoten] = true; | *(der Knoten mit der Knotennummer aktuellerKnoten wird als besucht gespeichert)* |
| anzahlBesucht++; | *(die Anzahl der besuchten Knoten wird um 1 erhöht)* |
| ausWartelisteEntfernen(); | *(der bereits besuchte Knoten wird aus der Warteliste vorne entfernt)* |
| aktuellerKnoten = warteliste[0]; | *(die erste Knoten in der Warteliste wird zum neuen akutellenKnoten)* |

Mit der Breitensuche kann geprüft werden, ob ein Graph zusammenhängend ist.

Es kann auch ein bestimmter Knoten eines Graphen gesucht werden. Dieser Knoten wird auf dem Pfad mit der geringsten Kantenanzahl gefunden.