

Hilfestellung zur Programmierung mit dem Graphentester



Hier findest du die vollständige Dokumentation für die Methoden, die der Graphentester mitbringt.

Aktion auf allen Knoten/Kanten

Soll der Algorithmus mit allen Knoten des Graphen etwas Bestimmtes machen, dann verwendet man die Methode `getAllKnoten()` bzw. `getAllKanten()` des Objekts `g` (Klasse `Graph`):

```
import java.util.List;

// Eine Liste aller Knoten anfordern
List<Knoten> alleKnoten = g.getAllKnoten();

// Schleife über alle Knoten der Liste
for (Knoten aktuellerKnoten: alleKnoten) {
    // mache etwas mit dem aktuellen Knoten, z.B. markieren
    aktuellerKnoten.setMarkiert(true);
    // Ausführung unterbrechen
    step();
}
```

Aktion auf bestimmten Knoten/Kanten

Soll der Algorithmus mit bestimmten Knoten/Kanten des Graphen etwas bestimmtes machen, dann holt man sich eine Liste aller Knoten/Kanten, die einer Bedingung genügen. Diese Bedingung kann als Prädikat, d.h. einem Lambda-Ausdruck, der `true` oder `false` zurück liefert, angegeben werden.

```
import java.util.List;

// Eine Liste aller markierten Knoten anfordern
List<Knoten> markierteKnoten = g.getAllKnoten(k->k.isMarkiert());

// Schleife über alle Knoten der Liste
for (Knoten aktuellerKnoten: markierteKnoten ) {
    // mache etwas mit dem aktuellen Knoten, z.B. Markierung löschen
    aktuellerKnoten.setMarkiert(false);
    // Ausführung unterbrechen
    step();
}
```

Diese Beschränkung auf eine Teilmenge mittels eines Lambda-Ausdrucks funktioniert auch bei den Methoden:

- `getAlleKanten(...)`
- `getAusgehendeKanten(...)`
- `getEingehendeKanten(...)`
- `getNachbarknoten(...)`

Knoten/Kanten sortieren oder Aktion mit Knoten/Kante mit kleinstem/größtem Wert

Möchte man eine Knoten- oder Kantenliste sortieren oder den größten/kleinsten wert ermitteln, kann man folgendermaßen vorgehen:

Zunächst holt man sich eine Liste der Knoten/Kanten. Mit `Collections.sort(...)` kann man die Kanten nach ihrem Gewicht und die Knoten nach ihrem Wert aufsteigend sortieren lassen. Das erste Element der Liste (Index=0) ist dann das mit dem kleinsten Wert, das letzte das mit dem größten Wert (Index=(Länge der Liste -1)).

```
import java.util.List;
import java.util.Collections;

// Liste aller Kanten holen
List<Kante> kanten = g.getAlleKanten();
// Kanten nach Gewicht aufsteigend sortieren
Collections.sort(kanten);
// Kante mit minimalem bzw. maximalem Gewicht ermitteln
Kante min = kanten.get(0);
Kante max = kanten.get(kanten.size()-1);
```

Arbeiten mit einer ToDo-Liste

In vielen Fällen arbeitet man die Knoten in einer vorher nicht feststehenden Reihenfolge ab. Die Knoten werden während des Algorithmus einer ToDo-Liste hinzugefügt und dann der Reihe nach abgearbeitet, bis die ToDo-Liste leer ist.

```
import java.util.ArrayList;

// ToDo-Liste erzeugen und mit Startknoten füllen
ArrayList<Knoten> toDo = new ArrayList<Knoten>();
toDo.add(getStartKnoten());

// Solange die ToDo-Liste nicht leer ist...
while(toDo.size()>0) {
    // ersten Knoten aus der Liste herausnehmen
    Knoten k = toDo.remove(0);

    // markiere ihn als besucht
    k.setBesucht(true);
}
```

```
// füge alle nicht besuchten Nachbarknoten der Liste hinzu
for(Knoten Nachbar : g.getNachbarKnoten(k)) {
    if(!Nachbar.isMarkiert() && !todo.contains(Nachbar)) {
        todo.add(0, n); //füge am Anfang der Liste hinzu (Tiefensuche)
        // oder
        // todo.add(n) //füge am Ende der Liste hinzu (Breitensuche)
    }
}
step(); // Ausführung unterbrechen
}
```

From:
<https://info-bw.de/> -

Permanent link:
<https://info-bw.de/faecher:informatik:oberstufe:graphen:zpg:hilfekarten:start?rev=1668444560>

Last update: **14.11.2022 16:49**

