Problemstellung

Warenverkehr bei einer Aufbausimulation (alle Bilder Pixabay-Lizenz - kein Nachweis notwendig)

Bei einem Aufbausimulationsspiel kann man verschiedene Gebäude bauen. Jedes Gebäude produziert eine bestimmte Ware. Damit das Gebäude arbeiten kann, benötigt es bestimmte Waren:

Die Minen müssen mit Essen versorgt werden, das entweder Fleisch vom Metzger oder Brot vom Bäcker sein kann. Sie produzieren Kohle bzw. Eisenerz. Die Eisenschmelze stellt daraus das vom Werkzeugmacher benötigte Eisen her. Der Werkzeugmacher heizt seine Schmiede mit Kohle und erzeugt aus dem Eisen Sensen für die Getreidefarm, Fleischermesser für den Metzger und Pickel für die Minen. Mit dem Getreide der Farm werden die Schweine gefüttert und in der Mühle das Mehl für den Bäcker gemahlen. Die Schweine werden vom Metzger geschlachtet.

Diese Abhängigkeiten machen es schwer zu entscheiden, in welcher Reihenfolge man die Gebäude bauen sollte, damit sie nicht ungenutzt herumstehen.

1. *Gib (wenn möglich) eine sinnvolle Baureihenfolge der Gebäude an.*

Modellierung

Die Problematik der gegenseitigen Abhängigkeiten tritt in vielen Situationen auf. Daher sollte es möglich sein, automatisch mit einem Algorithmus zu entscheiden, ob überhaupt eine sinnvolle Reihenfolge gefunden werden kann und wenn ja welche. Diese Reihenfolge nennt man topologische Sortierung. Dazu soll die Ausgangssituation zunächst als gerichteter Graph modelliert werden.

2. *Entscheide, welche der folgenden Informationen wichtig für die Festlegung der Reihenfolge sind:*
   - Name der Gebäude
   - Welchen Rohstoff produzieren die Gebäude?
   - Welchen Rohstoff benötigen die Gebäude?
   - Welches Gebäude produziert den benötigten Rohstoff für ein Gebäude?
   - Anzahl der Gebäude

Modellierung

Knoten:

Kanten:

Topologische Sortierung

Weiterführende Fragen

Im Computerspiel gibt es zwei Möglichkeiten, Verklemmungen zu lösen.

- Weitere Gebäude werden eingeführt: z.B. eine Fischerhütte, die ohne Voraussetzung Essen (Fisch) produziert.
- Man erhält eine Anzahl von Anfangsgegenständen, mit denen man die Produktion in Gang bringen kann: z.B. 20 Pickel.

Aufgaben

3. *Gib an, welche Kanten des ursprünglichen Graphen (zumindest vorübergehend) entfernt werden können, wenn man die Fischerhütte hinzufügt und 20 Pickel zur Verfügung stehen. Zeige, dass so die Verklemmung gelöst wird und eine topologische Sortierung nun möglich ist.*
4. *Analysiere, welche Gegenstände am Spielanfang zur Verfügung gestellt werden könnten, um die Verklemmung ohne ein neues Gebäude zu lösen.*
5. *Entscheide, ob man den Algorithmus für die topologische Sortierung auch auf einen ungerichteten Graph anwenden kann.*
6. *Analysiere, unter welcher Voraussetzung es nicht möglich ist, eine topologische Sortierung zu finden.*

Beschreibung des Algorithmus, um eine topologische Sortierung zu bestimmen

Bei jedem Knoten wird als Wert die Anzahl der eingehenden Kanten (Eingangsgrad) eingetragen. Das ist die Anzahl der noch unerfüllten Voraussetzungen.

Danach wiederholt man folgendes:

- Suche einen noch unmarkierten Knoten, der den Wert 0 hat (= alle Voraussetzungen sind erfüllt). Füge diesen der topologischen Sortierung hinzu und markiere ihn als bearbeitet. Reduziere den Wert aller Knoten um 1, zu denen eine Kante vom aktuellen Knoten führt.
- Gibt es keinen Knoten mit dem Wert 0 und es sind noch nicht alle Knoten abgearbeitet, dann ist es nicht möglich, eine topologische Sortierung zu finden.

Pseudocode des Algorithmus, um eine topologische Sortierung zu bestimmen

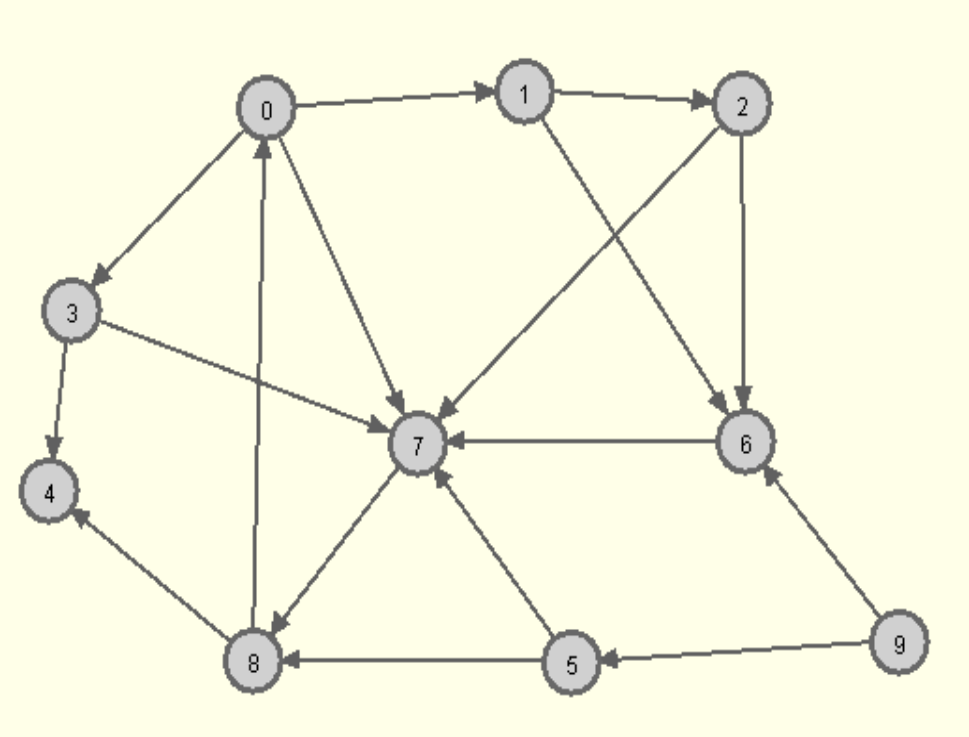
Topologische Sortierung:

```
Für jeden Knoten k des Graphen:
   setze den Wert des Knoten auf seinen Eingangsgrad

Setze topologische_Sortierung auf leer

Wiederhole solange es einen unmarkierten Knoten mit Wert 0 gibt
   Nimm einen unmarkierten Knoten k mit Wert 0
   Markiere k

   Füge k der topologische_Sortierung hinzu

   Für jeden Nachbarknoten n von k:
      reduziere den Wert des Knoten n um 1
Ende Wiederhole

Falls noch unmarkierte Knoten übrig sind:
  => es gibt keine topologische Sortierung
sonst:
  => topologische_Sortierung ist gefunden
```

Quelltext des Algorithmus, um eine topologische Sortierung zu finden

```
if (g.getAnzahlKnoten()==0) {
   return;
}

ArrayList<Knoten> knoten = g.getAlleKnoten();
for(Knoten k: knoten) {
   k.setWert(g.getEingehendeKanten(k).size());
}

String reihenfolge = "";

while(knoten.size()>0) {
   Collections.sort(knoten);
   Knoten k = knoten.get(0);
   k.setMarkiert(true);

   if(k.getIntWert() != 0) {
      melde("Fehler: Wert ist nicht 0 - Zyklus vorhanden");
      return;
   } else {
      reihenfolge += " "+g.getKnoteninfo(k, false);
      knoten.remove(k);
      for(Knoten k2 : g.getNachbarKnoten(k)) {
         k2.setWert(k2.getIntWert()-1);
      }
   }
}
melde("Topologische Sortierung: "+reihenfolge);
```

Beispiel 1



Beispiel 2