Programmierkurs Python II – SS 2011

Übung 3

---

1 Wikipedia-Graph mit Gewichten (2 Punkte)

Die Datei `wikipedia.txt` enthält eine kleine Sammlung von Wikipedia-Links. Die Links sind als Listen von String-Paaren gespeichert, wobei jeder String ein Seitentitel (im Wikipedia-Titel-Format) ist. Die Strings sind mit Tabulatoren getrennt; der erste markiert die Quelle, der zweite das Ziel eines Links.

Implementiere eine Methode `parse(dateiname)`, die einen Dateinamen als Parameter nimmt und einen gerichteten Graphen zurückgibt, dessen Knoten mit den Wikipedia-Seitentiteln markiert sind und dessen Kanten Wikipedia-Links symbolisieren. Du kannst hierfür die Graph-Infrastruktur aus Übung 2 benutzen.

Die Kanten des Graphen sollen gewichtet werden. Dafür musst Du die Datenstruktur erweitern. Implementiere eine Methode `weight(node,node)`, die das Gewicht einer Kante zwischen zwei Knoten zurück gibt. Wenn zwischen den beiden Knoten keine Kante existiert, soll die Methode 0 zurück geben.

Im Wikipedia-Graph soll jede Kante symbolisch ein Gewicht bekommen, das anzeigt, wie „wahrscheinlich “ es ist, dass der Link benutzt wird. Benutze für eine Kante `u,v` das Gewicht `100*indeg(v)/outdeg(u)`, also Ein- und Ausgangsgrad der beiden Knoten (auf `int` gerundet). Implementiere hierfür eine Methode `computeWeights()`, die in `__init__` aufgerufen wird (ggf. optional), aber auch getrennt aufgerufen werden kann.

2 Dijkstra (3 Punkte)

1. Der Dikstra-Algorithmus funktioniert nicht mit negativen Kantengewichten. Erkläre wieso. **(1 Punkt)**

2. Implementiere den Dijkstra-Algorithmus in einer Methode `dijkstra(node)` in der Graphklasse, die den Dijkstra-Algorithmus mit `node` als Startknoten ausführt. Du kannst ihn auf dem Wikipedia-Graphen testen, z.B. mit „Apple“ als Startknoten. Rückgabe des Algorithmus soll ein Dictionary sein: Schlüssel des Dictionaries sind alle Knoten des Graphen, Werte sind Paare aus 1. der Distanz zum Startknoten und 2. dem Vorgängerknoten bei der Berechnung dieser Distanz. **(2 Punkte)**

3 Topologische Sortierung (3 Punkte)

Implementiere topologische Sortierung für Graphen in einer Methode `tsort()` In der Graphklasse. Parse die Datei `frikadellen.txt` zu einem Testgraphen (gleiches Format wie `wikipedia.txt`).

Füge der Graph-Klasse eine Methode `tsort` hinzu, die die Knoten des Graphen topologisch sortiert in einer Liste zurück gibt. Du kannst die Ergebnisse Deiner Methode mit dem Output des UNIX-Tools `tsort` vergleichen, das Listen wie die in `frikadellen.txt` als Graph betrachtet und topologisch sortiert:

```
dhcp104-206: Michaela$ tsort < frikadellen.txt
```

(Beachte hierbei, dass es mehrere mögliche topologische Sortierungen gibt.)

4 Cliquen (Bonusaufgabe, 3 Punkte)

Implementiere den Basis-Algorithmus zur Extraktion von Cliquen aus einem Graphen. Teste ihn auf dem Wikipedia-Graphen. Pseudocode für eine rekursive Funktion ist unten angegeben. (G = Graph, CS = aktuelle Cliquenmenge, CA = Kandidatenmenge, NOT = Visited, $CLIQUES$ = gefundene Cliquen)

Procedure `ExtractCliques`(*G, CS, CA, NOT, CLIQUES*)

```
1  if CA = ∅ AND NOT = ∅ then
2      CLIQUES ← {CS}∪ CLIQUES
3  else
4      if ∃a ∈ NOTs.t.∀c ∈ CA : (a, c) ∈ E then
5          return
6  end
7  if CA = ∅ then
8      return
9  else
10     forall ca ∈ CA do
11         if ca ∉ NOT then
12             CS' ← {ca} ∪ CS
13             CA ← CA \ {ca}
14             NOT' ← ∅
15             CA' ← ∅
16             forall a ∈ NOT do if (a, ca) ∈ E then
17                 NOT' ← NOT' ∪ {CA}
18             forall c ∈ CA do if (c, ca) ∈ E then
19                 CA' ← CA' ∪ {ca}
20             NOT = NOT ∪ {ca}
21             ExtractCliques(G, CS', CA', NOT', CLIQUES)
22     end
23 end
```

Abgabe bis Donnerstag, 12.05.2011, 08:30 Uhr