Script generated by TTT

Title: Meixner: test2 (31.01.2013)

Date: Thu Jan 31 10:44:18 CET 2013

Duration: 71:30 min

Pages: 21

4.4 Greedy-Algorithmus

Sei $M = (S, U)$ ein Matroid, $w : S \to R$ eine Gewichtsfunktion.

```
algorithm greedy(S,U,w)
  B := ∅
  while (|B|<r(M)) do
    sei x ∈ {y ∈ S \ B; B ∪ {y} ∈ U} mit
      minimalem Gewicht
    B := B ∪ {x}
  od
end
```


4.3 Matroide

Definition 305

Sei S eine endliche Menge, $U \subseteq 2^S$ eine Teilmenge der Potenzmenge von S. Dann heißt $M = (S, U)$ ein Matroid und jedes $A \in U$ heißt unabhängige Menge, falls gilt:

1. $\emptyset \in U$
2. $A \in U,\ B \subseteq A \implies B \in U$
3. $$A, B \in U,\ |B| = |A| + 1 \implies (\exists x \in B \setminus A)\Big[(A \cup \{x\}) \in U\Big]$$

Jede bezüglich $\subseteq$ maximale Menge in U heißt Basis.
Nach 3. haben je zwei Basen gleiche Kardinalität. Diese heißt der Rang $\mathrm{r}(M)$ des Matroids.

4.4 Greedy-Algorithmus

Sei $M = (S, U)$ ein Matroid, $w : S \to R$ eine Gewichtsfunktion.

```
algorithm greedy(S,U,w)
  B := ∅
  while (|B|<r(M)) do
    sei x ∈ {y ∈ S \ B; B ∪ {y} ∈ U} mit
      minimalem Gewicht
    B := B ∪ {x}
  od
end
```

Beweis:

Aus der Definition des Matroids (1.) folgt, dass die leere Menge $\emptyset$ eine unabhängige Menge ist.

Aus 3. folgt, dass in der `while`-Schleife wiederum nur unabhängige Mengen generiert werden.

Daher ist B am Ende des Algorithmus eine Basis (da inklusionsmaximal). Es bleibt zu zeigen, dass die gefundene Basis minimales Gewicht besitzt.

Sei also $B = \{b_1, \ldots, b_r\}$ die vom Algorithmus gelieferte Basis. Sei $b_1, \ldots, b_r$ die Reihenfolge der Elemente, in der sie der Greedy-Algorithmus ausgewählt hat. Dann gilt

$$w(b_1) \leq w(b_2) \leq \ldots \leq w(b_r).$$

Beweis (Forts.):

Sei weiter $B' = \{b'_1, \ldots, b'_r\}$ eine minimale Basis, und es gelte o. B. d. A.

$$w(b'_1) \leq w(b'_2) \leq \ldots \leq w(b'_r) .$$

Sei $i \in \{1, \ldots, r\}$. Gemäß Eigenschaft 3 für Matroide folgt, dass es ein $b' \in \{b'_1, \ldots, b'_i\}$ gibt, so dass $\{b_1, \ldots, b_{i-1}, b'\} \in U$.

Damit ist $w(b_i) \leq w(b'_i)$ (für alle i), und daher wegen der Minimalität von B'

$$w(b_i) = w(b'_i) \qquad \text{für alle } i .$$

$\square$

4.5 Minimale Spannbäume

Satz 309

Sei $G = (V, E)$ ein zusammenhängender, ungerichteter Graph, $F \subseteq 2^E$ die Menge der kreisfreien Teilmengen von E. Dann ist $M = (E, F)$ ein Matroid mit Rang $|V| - 1$.

Beweis:

Es sind die drei Eigenschaften eines Matroids zu zeigen.

Beweis (Forts.):

3. Sind A und B kreisfrei, $|B| = |A| + 1$, dann existiert ein $b \in B$, so dass $A \cup \{b\}$ kreisfrei ist:

   Wir betrachten die Wälder (V, A) (mit $|A|$ Kanten und $|V| - |A|$ Zusammenhangskomponenten) und (V, B) (mit $|B|$ Kanten und $|V| - |B|$ Zusammenhangskomponenten). Diese Bedingungen lassen zwei Möglichkeiten zu:

$\square$

Beweis (Forts.):

3. Sind A und B kreisfrei, $|B| = |A| + 1$, dann existiert ein $b \in B$, so dass $A \cup \{b\}$ kreisfrei ist:

   Wir betrachten die Wälder (V, A) (mit $|A|$ Kanten und $|V| - |A|$ Zusammenhangskomponenten) und (V, B) (mit $|B|$ Kanten und $|V| - |B|$ Zusammenhangskomponenten). Diese Bedingungen lassen zwei Möglichkeiten zu:

   1. Es existiert eine Kante e in B, die zwei Zusammenhangskomponenten in (V, A) verbindet. Damit ist $A \cup \{e\}$ kreisfrei.
   2. Alle Kanten in B verlaufen innerhalb der Zusammenhangskomponenten in (V, A). (V, A) besitzt jedoch eine Zusammenhangskomponente mehr als (V, B). Daher muss es eine Zusammenhangskomponente in (V, A) geben, deren Knoten nicht in (V, B) auftauchen, was einen Widerspruch darstellt.

□

Kruskals Algorithmus:

```
algorithm kruskal
  sortiere E aufsteigend: w(e_1) ≤ ... ≤ w(e_m).
  F := ∅
  i := 0
  while |F|<|V|-1 do
    i++
    if F ∪ {e_i} kreisfrei then
      F := F ∪ {e_i}
    fi
  od
end
```


Satz 310

Kruskals Algorithmus bestimmt (bei geeigneter Implementierung) einen minimalen Spannbaum für $G = (V, E)$ *in Zeit* $O\Big(|E| \cdot \log(|V|)\Big)$.

Beweis:
Die Korrektheit folgt aus Satz 309.
Zur Laufzeit:
Die Sortierung von E nach aufsteigendem Gewicht benötigt

$$O\Big(|E| \cdot \log(|E|)\Big),$$

z. B. mit Heapsort oder Mergesort.
Da $|E| \le (|V|)^2$, gilt auch

$$O\Big(|E| \cdot \log(|V|)\Big)$$

als Zeitbedarf für das Sortieren.

Implementierung des Tests auf Kreisfreiheit:

Repräsentation der Zusammenhangskomponenten:
Feld Z: $Z[i]$ ist die Zusammenhangskomponente des Knoten i.
Feld N: $N[j]$ ist die Anzahl der Knoten in der Zusammenhangskomponente j.
Feld M: $M[j]$ ist eine Liste mit den Knoten in der Zusammenhangskomponente j.

```
co Initialisierung oc
for all i ∈ V do
  Z[i] := i
  N[i] := 1
  M[i] := (i)
od
co Test auf Kreisfreiheit oc
sei e := {i,j}
```

Fortsetzung

```
co F ∪ {e} kreisfrei ⇔ Z[i]≠Z[j] oc
if Z[i]≠Z[j] then
  if N[Z[i]] <= N[Z[j]] then
    BigSet := Z[j]
    SmallSet := Z[i]
  else
    BigSet := Z[i]
    SmallSet := Z[j]
  fi
  N[BigSet] := N[BigSet] + N[SmallSet]
  for all k ∈ M[SmallSet] do
    Z[k] := BigSet
  od
  hänge M[SmallSet] an M[BigSet] an
fi
```


Beweis (Forts.):

Zeitbedarf für den Test: $O(1)$ für jede Abfrage, damit dafür insgesamt

$$O(|E|).$$

Zeitbedarf für das Umbenennen der Zusammenhangskomponenten: Nach jedem Umbenennen befindet sich ein Knoten in einer mindestens doppelt so großen Zusammenhangskomponente. Daher ist die Anzahl der Umbenennungen je Knoten $\leq \log(|V|)$. Für das Umbenennen aller Knoten benötigt man dann

$$O\Big(|V| \cdot \log(|V|)\Big).$$

$\square$

Bemerkung:
Es gibt Algorithmen für minimale Spannbäume der Komplexität $O(m + n \cdot \log n)$ und, für dünnbesetzte Graphen, der Komplexität $O(m \cdot \log^* n)$, wobei

$$\log^* x = \min_{n \in \mathbb{N}} \Big\{ n \colon \underbrace{\log\Big(\log(\cdots\log(x)\cdots)\Big)}_{n} < 1 \Big\}.$$

5. Spezielle Pfade

5.1 Eulersche Pfade und Kreise

Definition 311

Ein Pfad bzw. Kreis in einem Graphen (Digraphen) heißt eulersch, wenn er jede Kante des Graphen genau einmal enthält.
Ein Graph (Digraph) heißt eulersch, wenn er einen eulerschen Kreis enthält.

Satz 312

Ein Graph besitzt genau dann einen eulerschen Kreis (Pfad), wenn er zusammenhängend ist und alle (alle bis auf zwei) Knoten geraden Grad haben.

Beweis:

„⇒“

Ein eulerscher Graph muss notwendigerweise zusammenhängend sein. Die Knotengrade müssen gerade sein, da für jede zu einem Knoten (auf dem eulerschen Kreis) hinführende Kante auch eine von diesem Knoten weiterführende Kante existieren muss, da sonst der eulersche Kreis nicht fortgeführt werden kann.

Beweis (Forts.):

„⇐“

Konstruktion des eulerschen Kreises: Man suche einen beliebigen Kreis im Graphen (muss aufgrund der Voraussetzungen existieren). Sind noch Kanten unberücksichtigt, suche man auf dem Kreis einen Knoten, der zu noch nicht verwendeten Kanten inzident ist.

Nach Voraussetzung muss sich wieder ein Kreis finden lassen, der vollständig aus noch nicht berücksichtigten Kanten besteht. Diesen füge man zum bereits gefundenen Kreis hinzu, worauf sich ein neuer Kreis ergibt.

Dieses Verfahren läßt sich fortführen, bis keine Kanten mehr unberücksichtigt sind und damit ein eulerscher Kreis gefunden ist.

□

Satz 313

Ein Digraph besitzt genau dann einen eulerschen Kreis (Pfad), wenn er stark zusammenhängend ist und für alle Knoten der In-Grad gleich dem Aus-Grad ist (wenn für einen Knoten In-Grad = Aus-Grad − 1, für einen weiteren Knoten In-Grad = Aus-Grad + 1 gilt und für alle anderen Knoten der In-Grad gleich dem Aus-Grad ist).

Beweis:

Der Beweis ist analog zum Beweis des vorhergehenden Satzes. □

5.2 Hamiltonsche Pfade

Ein Pfad (Kreis) in einem Graphen (Digraphen) heißt hamiltonsch, wenn er jeden Knoten genau einmal enthält.

Ein Graph (Digraph) heißt hamiltonsch, wenn er einen hamiltonschen Kreis enthält.

Beispiel 314 (Das Königsberger Brückenproblem)

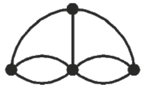
Dieser Graph besitzt einen hamiltonschen Kreis, aber weder einen eulerschen Kreis noch einen eulerschen Pfad.

Die Aufgabe, einen hamiltonschen Kreis zu finden, ist wesentlich schwerer als einen eulerschen Kreis zu finden; es ist ein $\mathcal{NP}$-vollständiges Problem.

5.2 Hamiltonsche Pfade

Ein Pfad (Kreis) in einem Graphen (Digraphen) heißt hamiltonsch, wenn er jeden Knoten genau einmal enthält.
Ein Graph (Digraph) heißt hamiltonsch, wenn er einen hamiltonschen Kreis enthält.

Beispiel 314 (Das Königsberger Brückenproblem)



Dieser Graph besitzt einen hamiltonschen Kreis, aber weder einen eulerschen Kreis noch einen eulerschen Pfad.

Die Aufgabe, einen hamiltonschen Kreis zu finden, ist wesentlich schwerer als einen eulerschen Kreis zu finden; es ist ein $\mathcal{NP}$-vollständiges Problem.

6. Kürzeste Wege

Gegeben sind ein (Di)Graph $G = (V, E)$ und eine Gewichtsfunktion $w : E \to \mathbb{R}^+ \cup \{+\infty\}$. O. B. d. A. sei G vollständig, damit auch zusammenhängend.

Sei $u = v_0, v_1, v_2, \ldots, v_n = v$ ein Pfad in G. Die Länge dieses Pfades ist

$$\sum_{i=0}^{n-1} w(v_i, v_{i+1}).$$

$d(u, v)$ sei die Länge eines kürzesten Pfades von u nach v.