WS 2003/04

Diskrete Strukturen I

Ernst W. Mayr

mayr@in.tum.de
Institut für Informatik
Technische Universität München

01-27-2004

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

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

Beweis

Die Korrektheit folgt aus Satz *9. Zur Laufzeit:*
Die Sortierung von E nach aufsteigendem Gewicht benötigt

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

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

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

als Zeitbedarf für das Sortieren.

Satz

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

Beweis

Die Korrektheit folgt aus Satz *9*. *Zur Laufzeit: Die Sortierung von E nach aufsteigendem Gewicht benötigt*

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

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

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

als Zeitbedarf für das Sortieren.

Satz

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

Beweis

Die Korrektheit folgt aus Satz *9. Zur Laufzeit:*
Die Sortierung von E nach aufsteigendem Gewicht benötigt

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

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

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

als Zeitbedarf für das Sortieren.

Satz

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

Beweis

Die Korrektheit folgt aus Satz *9. Zur Laufzeit:*
Die Sortierung von E *nach aufsteigendem Gewicht benötigt*

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

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

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

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}
```

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
```

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

$$O\big(|E|\big).$$

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

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

q. e. d.

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

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

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

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

Spezielle Pfade

Eulersche Pfade und Kreise

Definition

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

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

Spezielle Pfade

Eulersche Pfade und Kreise

Definition

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

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

Beweis

„$\Rightarrow$“

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.

„$\Leftarrow$“

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.

q. e. d.

Satz

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. □

Satz

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. □

Algorithmus zum Finden eines eulerschen Kreises:

```
algorithm Eulerian_Circle(V,E)
  EC := ∅
  select v=v_0 ∈ V
  do
    C := ∅
    while N(v)≠∅ do
      select w ∈ N(v)
      E := E\{v,w}
      C := C ∪ {v,w}
      if N(v)≠∅ then
        Q.add(v)
      fi
      v := w
    od
    EC := EC ∪ C co Neuer Kreis oc
```

Fortsetzung

```
        if not empty(Q) then
          v := Q.remove()
        fi
      until E = ∅
    end
```

Laufzeit des Algorithmus: $O(E)$.
Laufzeit der `while`-Schleife: $O(E)$, der `do`-`until`-Schleife ohne Durchlaufen der `while`-Schleife: $O(|V|)$ und damit ebenfalls $O(|E|)$, da der Graph zusammenhängend ist.

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 (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.

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 (Das Königsberger Brückenproblem)

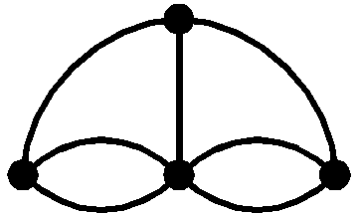
Dieser Graph besitzt einen hamiltonschen Kreis, aber weder einen eulerschen Kreis noch einen eulerschen Pfad.

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

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 (Das Königsberger Brückenproblem)



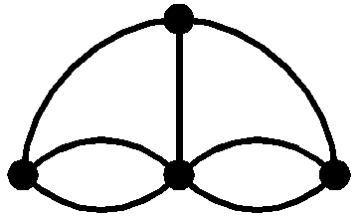
Dieser Graph besitzt einen hamiltonschen Kreis, aber weder einen eulerschen Kreis noch einen eulerschen Pfad.

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

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 (Das Königsberger Brückenproblem)



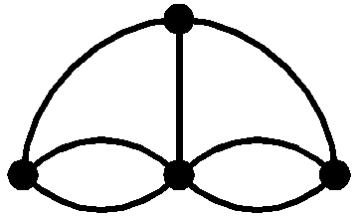
Dieser Graph besitzt einen hamiltonschen Kreis, aber weder einen eulerschen Kreis noch einen eulerschen Pfad.

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

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 (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.

Kürzeste Wege

Gegeben sind ein 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

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

$\mathsf{d}(u, v)$ ist die Länge eines kürzesten Pfades von u nach v.

Kürzeste Wege

Gegeben sind ein 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

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

$\mathsf{d}(u, v)$ ist die Länge eines kürzesten Pfades von u nach v.

Problemstellungen:

1. Gegeben $u, v \in V$, berechne $\mathsf{d}(u, v)$.
2. Gegeben $u \in V$, berechne für alle $v \in V$ die Länge $\mathsf{d}(u, v)$ eines kürzesten Pfades von u nach v (*sssp, single source shortest path).*
3. *Berechne für alle $(u, v) \in V^2$ die kürzeste Entfernung $\mathsf{d}(u, v)$ (apsp, all pairs shortest path).*

Problemstellungen:

1. Gegeben $u, v \in V$, berechne $\mathsf{d}(u, v)$.
2. Gegeben $u \in V$, berechne für alle $v \in V$ die Länge $\mathsf{d}(u, v)$ eines kürzesten Pfades von u nach v (*sssp, single source shortest path).*
3. *Berechne für alle $(u, v) \in V^2$ die kürzeste Entfernung $\mathsf{d}(u, v)$ (apsp, all pairs shortest path).*

Problemstellungen:

1. Gegeben $u, v \in V$, berechne $\mathsf{d}(u, v)$.
2. Gegeben $u \in V$, berechne für alle $v \in V$ die Länge $\mathsf{d}(u, v)$ eines kürzesten Pfades von u nach v (*sssp, single source shortest path).*
3. *Berechne für alle $(u, v) \in V^2$ die kürzeste Entfernung $d(u, v)$ (apsp, all pairs shortest path).*