Algorithmen für Planare Graphen

24. Juni 2021, Übung 4

Lars Gottesbüren

INSTITUT FÜR THEORETISCHE INFORMATIK

Umfang

Der *Umfang* (engl. girth) eines Graphen G ist die Länge eines kürzesten Kreises in G. Enthält G keinen Kreis, so ist der Umfang ∞.

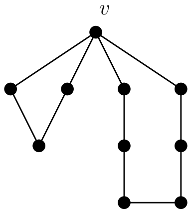
Geben Sie einen Algorithmus an, der für einen Knoten v von G entweder

- die Länge des kürzesten Kreises berechnet auf dem v liegt, oder
- entscheidet, dass v nicht auf einem Kreis liegt.

Umfang

SHORTCIRCLE(v)

- Breitensuche beginnend bei v
- Wird ein Knoten zum zweiten Mal besucht, ist ein Kreis gefunden.
- Ein kürzester Kreis wird zuerst gefunden.



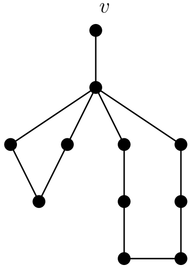
Umfang

SHORTCIRCLE(v)

- Breitensuche beginnend bei v
- Wird ein Knoten zum zweiten Mal besucht, ist ein Kreis gefunden.
- Ein kürzester Kreis wird zuerst gefunden.

Aber: v liegt nicht notwendigerweise auf diesem Kreis. Deshalb:

- Nummeriere ("labele") Knoten auf dem ersten Level (Nachbarn von v)
- Vererbe Label auf neu gefundene Knoten
- Wenn sich unterschiedliche Label treffen ist das ein Kreis, der v enthält.



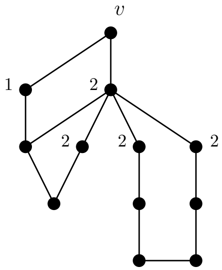
Umfang

SHORTCIRCLE(v)

- Breitensuche beginnend bei v
- Wird ein Knoten zum zweiten Mal besucht, ist ein Kreis gefunden.
- Ein kürzester Kreis wird zuerst gefunden.

Aber: v liegt nicht notwendigerweise auf diesem Kreis. Deshalb:

- Nummeriere ("labele") Knoten auf dem ersten Level (Nachbarn von v)
- Vererbe Label auf neu gefundene Knoten
- Wenn sich unterschiedliche Label treffen ist das ein Kreis, der v enthält.



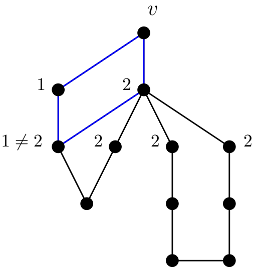
Umfang

SHORTCIRCLE(v)

- Breitensuche beginnend bei v
- Wird ein Knoten zum zweiten Mal besucht, ist ein Kreis gefunden.
- Ein kürzester Kreis wird zuerst gefunden.

Aber: v liegt nicht notwendigerweise auf diesem Kreis. Deshalb:

- Nummeriere ("labele") Knoten auf dem ersten Level (Nachbarn von v)
- Vererbe Label auf neu gefundene Knoten
- Wenn sich unterschiedliche Label treffen ist das ein Kreis, der v enthält.



Umfang

Verwenden Sie das Verfahren aus Aufgabenteil 6.1, um für einen beliebigen Graphen den Umfang zu berechnen. Welche Laufzeit erhalten Sie?

- Breitensuche für eine Zusammenhangskomponente liegt in $\mathcal{O}(|V| + |E|) = \mathcal{O}(|E|)$.
- Wiederhole für jeden Knoten und gib den kleinsten Kreis aus: $\mathcal{O}(|V||E|)$.

Umfang

Verwenden Sie das Verfahren aus Aufgabenteil 6.1, um für einen beliebigen Graphen den Umfang zu berechnen. Welche Laufzeit erhalten Sie?

- Breitensuche für eine Zusammenhangskomponente liegt in $\mathcal{O}(|V| + |E|) = \mathcal{O}(|E|)$.
- Wiederhole für jeden Knoten und gib den kleinsten Kreis aus: $\mathcal{O}(|V||E|)$.

Umfang für planare Graphen

Algorithm PLANARGIRTH

$C \leftarrow \infty$

for jede Zusammenhangskomponente H von G **do**

$(V_1, V_2, S) \leftarrow \textsc{PlanarSeparator}(H)$

$C \leftarrow \min\{C, \min\{\textsc{ShortCircle}(v) \mid v \in S\}\}$

$C \leftarrow \min\{C, \textsc{PlanarGirth}(V_1)\}$

$C \leftarrow \min\{C, \textsc{PlanarGirth}(V_2)\}$

return C

Umfang für planare Graphen

Algorithm PLANARGIRTH

$C \leftarrow \infty$

for jede Zusammenhangskomponente H von G **do** $\mathcal{O}(n_G^{1,5} \log(n_G))$

$(V_1, V_2, S) \leftarrow \textsc{PlanarSeparator}(H)$ $\mathcal{O}(n_H)$ $\mathcal{O}(n_H^{1,5})$

$C \leftarrow \min\{C, \min\{\textsc{ShortCircle}(v) \mid v \in S\}\}$ $\mathcal{O}(\sqrt{n_H} \cdot n_H)$

$C \leftarrow \min\{C, \textsc{PlanarGirth}(V_1)\}$ $\mathcal{O}(T(|V_1|))$ $\mathcal{O}(T(n_H))$

$C \leftarrow \min\{C, \textsc{PlanarGirth}(V_2)\}$ $\mathcal{O}(T(|V_2|))$

return C

- $\mathcal{O}(\log(n))$ Rekursionslevel
- Jeder Knoten nur einmal pro Level $\Rightarrow$ pro Level $\mathcal{O}(n^{1,5})$ Arbeit
- Insgesamt: $\mathcal{O}(\log(n) \cdot n^{1,5})$

MST

MST in erwartet $\mathcal{O}(n)$

- Gegeben: G ungerichteter, planarer, zusammenhängend Graph.

Ohne Beweis zu verwenden:

- Sei v ein Knoten und e eine Kante minimalen Gewichts inzident zu v. Dann gibt es einen MST von G, der e enthält.
- Sei e eine Kante, die in einem MST von G vorkommt. Sei T ein MST auf dem Graphen, den man durch die Kontraktion von e erhält. Dann ist $T \cup e$ ein MST auf G.

MST

Algorithm MST

$T \leftarrow \emptyset$

while G nicht leer **do**

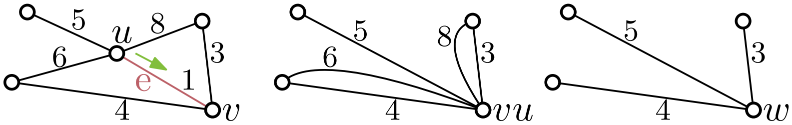
  Wähle $v \in V$ mit $\deg(v) \leq 5$

  $e \leftarrow \operatorname{argmin}_{e' \in \mathcal{N}(v)} w(e')$

  $T \leftarrow T \cup e$

  Kontrahiere e

  Entferne Multikanten und behalte nur die mit geringstem Gewicht



Erwartete Laufzeit

Kontraktion von $\{u, v\}$ mit $\deg(v) \leq 5$

- Pro Knoten eine Hashmap für seine Nachbarschaft
- Für $x \in \mathcal{N}(v)$ erkenne ob $x \in \mathcal{N}(u)$; in erwartet $\mathcal{O}(1)$
- Falls ja, behalte die Kante mit kleinerem Gewicht in Hashmap für u

Max Independent Set

Algorithm MWIS

$I_{\text{opt}} \leftarrow \emptyset$
$(S, V_1, V_2) \leftarrow \text{PST}(G)$
for $I_S \in \mathcal{P}(S)$ **do**
  $I_1 \leftarrow \text{MWIS}(G[V_1 - N(I_S)])$
  $I_2 \leftarrow \text{MWIS}(G[V_2 - N(I_S)])$
  $I \leftarrow I_S \cup I_1 \cup I_2$
  if $|I| > |I_{\text{opt}}$ **then**
    $I_{\text{opt}} \leftarrow I$
return I_{opt}

Max Independent Set

Algorithm MWIS

$I_{\text{opt}} \leftarrow \emptyset$
$(S, V_1, V_2) \leftarrow \text{PST}(G)$
for $I_S \in \mathcal{P}(S)$ **do**
  $I_1 \leftarrow \text{MWIS}(G[V_1 - N(I_S)])$
  $I_2 \leftarrow \text{MWIS}(G[V_2 - N(I_S)])$
  $I \leftarrow I_S \cup I_1 \cup I_2$
  if $|I| > |I_{\text{opt}}$ **then**
    $I_{\text{opt}} \leftarrow I$
return I_{opt}

$$t(n) \leq 2^{\sqrt{n}} \cdot t(\alpha^1 n) \leq 2^{\sqrt{n}} \cdot 2^{\sqrt{\alpha n}} \cdot t(\alpha^2 n) \leq \cdots \leq 2^{\sqrt{n} \cdot \sum_{i=0}^{\log(n)} \alpha^i}$$