Algorithmen und Datenstrukturen 2

ALGO2 · WiSe-2021/22 · tcs.uni-frankfurt.de/algo2 · 2022-02-07 · 7a1c9e6

Übungen zu Woche 14: Algorithmen für NP-schwere Probleme

Das Übungsblatt enthält alle empfohlenen Lernaktivitäten für die aktuelle Woche.

- **Heimarbeit bis Montag 17:00.**
  - Schau die Videos an und lies die Buchkapitel.
  - Bearbeite die 🌱-Aufgabe in Moodle. (Feste Abgabefrist!)
  - Lese den Aufgabentext aller Übungsaufgaben.
- **Heimarbeit.** Bearbeite die Übungsaufgaben soweit möglich. Probier zumindest alle mal!
- **Dienstag/Donnerstag.**
  - **8:00–8:15.** Besprechung im Zoom.
  - **8:15–9:15.** Bearbeite jetzt die Übungen, die du noch nicht lösen konntest. Sprich mit anderen Studis! Frag das Vorlesungsteam um Hilfe!
  - **9:15–9:45.** Lösungsspaziergang zu den Aufgaben für heute.
- **Heimarbeit bis Freitag, den 11.02., 17:00.** Gib deine Lösungen zu der ⭐-Aufgabe von diesem Übungsblatt in Moodle ab. (Feste Abgabefrist!)

Dienstag

Aufgabe 14.1 (Gieriges Vertex-Cover).

a) Finde den kleinsten Graphen (minimale Anzahl an Kanten), für den GreedyVertexCover *nicht* die kleinste Knotenüberdeckung ausgibt.

b) Beschreibe für alle ganzzahligen Werte von n einen Graphen mit höchstens $\text{poly}(n)$ Knoten, für den GreedyVertexCover eine Knotenüberdeckung der Größe $\text{OPT} \cdot \Omega(\log n)$ ausgibt.

Hinweis: Uns ist eine Konstruktion mit $\mathcal{O}(n \log n)$ Knoten bekannt.

Aufgabe 14.2 (Dummes Vertex-Cover).

a) Finde den kleinsten Graphen (minimale Anzahl an Kanten), für den DumbVertexCover *nicht* die kleinste Knotenüberdeckung ausgibt.

b) Beschreibe eine unendlich große Familie von Graphen, für welche DumbVertexCover eine Knotenüberdeckung der Größe $2 \cdot \text{OPT}$ ausgibt. (Hierbei ist OPT die Größe der kleinsten Knotenüberdeckung.)

Donnerstag

Aufgabe 14.3 (Tiefengesuchtes Vertex-Cover). Betrachte folgenden heuristischen Algorithmus, der in einem zusammenhängenden Graphen G eine Knotenüberdeckung berechnet:

1. Berechne einen Tiefensuchbaum T in G, dessen Wurzel ein beliebiger Knoten ist.
2. Gebe die Menge aller Knoten aus, die keine Blätter in T sind. (Also die Wurzel und alle inneren Knoten von T.)

a) Beweise, dass diese Heuristik eine Knotenüberdeckung in G findet.

b) Beweise, dass diese Heuristik eine 2-Approximation zur minimalen Knotenüberdeckung von G ist.

c) Beschreibe eine unendlich große Familie von Graphen, für welche diese Heuristik eine Knotenüberdeckung der Größe $2 \cdot \mathrm{OPT}$ ausgibt. (Hierbei ist OPT die Größe der kleinsten Knotenüberdeckung.)

Aufgabe 14.4 (Beschränkte Suchbäume für Hitting Set). Sei $U = \{1, \dots, n\}$ ein Universum mit n Elementen. Seien $B_1, B_2, \dots, B_m$ verschiedene Teilmengen von U. Eine Menge $H \subseteq A$ heißt *hitting set* für die Kollektion $B_1, B_2, \dots, B_m$, wenn H mindestens ein Element von jeder Menge B_i enthält. Mit anderen Worten: Für alle $i \in \{1, \dots, m\}$ gilt $H \cap B_i \neq \emptyset$.

Betrachte das HittingSet-Problem:

> HittingSet
> *Eingabe:* Mengen $B_1, \dots, B_m \subseteq U$ und Zahl $k \in \mathbb{N}$.
> *Frage:* Hat die Kollektion $B_1, \dots, B_m$ ein *hitting set* der Größe höchstens k?

Sei b die maximale Größe der Eingabemengen, also $b = \max_i |B_i|$. Überlege dir einen *bounded search tree* Algorithmus, der das HittingSet-Problem in einer Laufzeit von $f(k, b) \cdot \mathrm{poly}(n, m)$ löst, wobei $\mathrm{poly}(n, m)$ ein Polynom in n und m ist und $f(k, b)$ eine beliebige Funktion ist, die nur von b und k abhängt. Welche Funktion f und welches Polynom poly hat die Laufzeit deines Algorithmus?

Hinweis: Das Knotenüberdeckungsprolem VertexCover *ist der Spezialfall* $|B_1| = \cdots = |B_m| = 2$.

Sternaufgabe

Aufgabe 14.5 (⭐: Beschränkte Suchbäume für Clique). Wir haben in Woche 7 bereits das Entscheidungsproblem Clique kennengelernt:

> Clique
> *Eingabe:* Ein Graph G mit n Knoten und eine Zahl $k \in \mathbb{N}$.
> *Frage:* Enthält G eine Clique mit k Knoten?

a) Sei Δ der Maximalgrad von G. Überlege dir einen **rekursiven** *bounded search tree* Algorithmus, der das Clique-Problem in einer Laufzeit von $f(k, \Delta) \cdot \text{poly}(n)$ löst, wobei $\text{poly}(n)$ ein Polynom in n ist und $f(k, \Delta)$ eine beliebige Funktion ist, die nur von k und Δ abhängt. Beschreibe deinen Algorithmus **in Pseudocode**.

b) Beweise, dass dein Algorithmus korrekt ist.

c) Gib die Funktion f konkret an und beweise, dass dein Algorithmus die gegebene Laufzeitschranke einhält.

d) Wir wissen aus Woche 7, dass Clique NP-schwer ist. Impliziert dein Algorithmus aus Aufgabenteil a) nun P = NP? Erkläre deine Antwort in 2–3 Sätzen.