Algorithmen und Datenstrukturen 2

ALGO2 · WiSe-2023/24 · tcs.uni-frankfurt.de/algo2/ · 2024-02-05 · *7ae310c*

Übungen zu Woche 11: Algorithmen für NP-schwere Probleme

Dienstag: Approximationsalgorithmen

Moodle 🐣 **11.1 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 $O(n \log n)$ Knoten bekannt.

Moodle 🐣 **11.2 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.)

🏟 **11.3 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) 👍 Warum findet diese Heuristik eine Knotenüberdeckung in G?

b) 🔑 Warum berechnet sie eine 2-Approximation zur minimalen Knotenüberdeckung von G?

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

Donnerstag: Beschränkte Suchbäume

Moodle **11.4 Beschränkte Suchbäume für Hitting Set.** Sei $U = \{1, \ldots, n\}$ ein Universum mit n Elementen. Seien $B_1, B_2, \ldots, B_m$ verschiedene Teilmengen von U. Eine Menge $H \subseteq U$ heißt *hitting set* für die Kollektion $B_1, B_2, \ldots, B_m$, wenn H mindestens ein Element von jeder Menge B_i enthält. Mit anderen Worten: Für alle $i \in \{1, \ldots, m\}$ gilt $H \cap B_i \neq \emptyset$.

Betrachte das HittingSet-Problem:

> HittingSet
> *Eingabe:* Mengen $B_1, \ldots, B_m \subseteq U$ und Zahl $k \in \mathbb{N}$.
> *Frage:* Hat die Kollektion $B_1, \ldots, B_m$ 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* $|B_1| = \cdots = |B_m| = 2$.

11.5 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 (d.h. einen vollständigen Teilgraphen) 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 $f(k, \Delta) \cdot \mathrm{poly}(n)$ löst, wobei $\mathrm{poly}(n)$ 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.