Approximationsalgorithmen

Wintersemester 2018/2019

Dr. Annamaria Kovacs
Dipl-Math. Mahyar Behdju

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Institut für Informatik
AG Algorithm Engineering

Übung 1

Ausgabe: 17.10.2018
Abgabe: 24.10.2018

Aufgabe 1.1. (1 + 1 + 1 + 1 + 2 + 2 Punkte)

Was können wir vom Approximationsfaktor der folgenden Algorithmen behaupten? Gib möglichst gute obere oder untere Schranken an, wo möglich.

a) Ein Algorithmus, der für jede Instanz I eines Maximierungsproblems jeweils mit einem Zielwert von mindestens $\frac{2}{3}\mathrm{OPT}_I$ eine Lösung findet.

b) Ein Algorithmus, der für jede Instanz I eines Maximierungsproblems jeweils mit einem Zielwert von mindestens $(1-\varepsilon)\cdot \mathrm{OPT}_I$ eine Lösung findet.

c) Ein Algorithmus, der für manche Instanzen I eines Maximierungsproblems jeweils mit einem Zielwert von genau $\frac{2}{3}\mathrm{OPT}_I$ eine Lösung findet.

d) Ein Algorithmus für ein Minimierungsproblem, der manche Instanzen genau optimiert.

e) Ein Algorithmus für Min-Scheduling auf einer bestimmten Klasse von Instanzen, der für jede Instanz I aus dieser Klasse einen Schedule mit Makespan höchstens $\frac{4}{3}\alpha_I$ ausgibt, wobei $\alpha_I = \max\{p_{\max}, \frac{1}{m}\sum_{i=1}^{n} p_i\}$.

f) Ein Algorithmus für Max-Clique, der

- ein Dreieck ausgibt, wenn der Graph (mindestens) ein Dreieck enthält,
- eine Kante ausgibt, wenn der Graph kein Dreieck, aber mindestens eine Kante enthält,
- sonst einen Knoten ausgibt.

Aufgabe 1.2. (2 + 3 + 1 Punkte)

Ein ungerichteter Graph heißt *regulär*, wenn alle Knoten den gleichen Grad haben; wir nennen ihn r-regulär, wenn dieser Knotengrad r ist.

a) Bestimme die Anzahl der Kanten in einem r-regulären Graphen mit n Knoten.

b) Mindestens wie viele Knoten braucht eine Knotenüberdeckung eines r-regulären Graphen für $r \geqslant 1$?

c) Angenommen, irgendein Algorithmus für eine Klasse von *regulären* Graphen findet eine Knotenüberdeckung bestehend aus maximal $\frac{3n}{4}$ Knoten für jeden Graphen der Klasse mit n Knoten. Wie gut approximiert dieser Algorithmus das Problem Vertex Cover über diese Graphenklasse?

Begründe deine Antworten.

Bitte wenden!

Aufgabe 1.3. (3 + 4 + 3 Punkte)

Wir betrachten den Greedy LIST-Scheduling Algorithmus für das min-SCHEDULING Problem für $m = 50$ identische Maschinen, und die folgende Jobmenge: Für jede Zahl $i \in \{1, 2, \ldots, 100\}$ gibt es genau einen Job mit Laufzeit $p_i = i$.

a) Bestimme den genauen Approximationsfaktor von LIST **für diese Eingabe**, wenn die Jobs in aufsteigender Reihenfolge der Laufzeiten bearbeitet werden. Begründe kurz deine Antwort.

b) Wird der Approximationsfaktor besser, wenn zuerst alle Jobs mit Laufzeit mindestens 51 (aufsteigend) und danach die restlichen Jobs (aufsteigend) bearbeitet werden?

c) Wird der Approximationsfaktor besser (im Vergleich zu Teil a)), wenn zuerst alle Jobs mit Laufzeit mindestens 50 (aufsteigend) und danach die restlichen Jobs (aufsteigend) bearbeitet werden?

Hinweis: In den Teilen b) und c) wird eine kurze (aber exakte) Begründung ohne die Berechnung des ganzen Schedule gefragt.

Aufgabe 1.4. *Wiederholung* (2 Punkte)

Erstelle einen Huffman-Code für den Text T= *abracadabra*.

E-Mail: `{panni,mahyar}@ae.cs.uni-frankfurt.de`