Institut für Theoretische Informatik
Lehrstuhl Prof. Dr. D. Wagner

Übungsblatt 7 – Lösungsvorschläge

Vorlesung Algorithmentechnik im WS 09/10

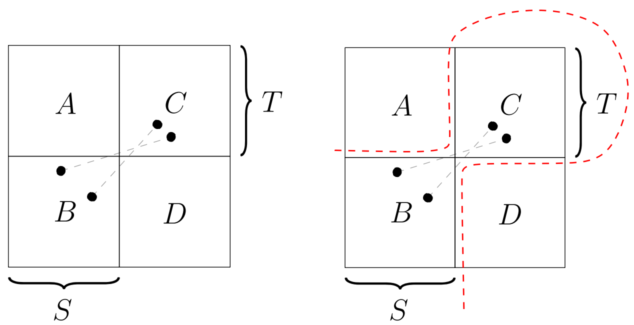
Problem 1: Minimale Schnittbasis – Approximationsalgos relativer Gütegarantie [vgl. Kap. 7.1]

Der *Kantenraum* $\mathcal{E}$ eines ungerichteten, zusammenhängenden Graphen $G = (V, E)$ sei der Vektorraum aller Teilmengen der Kantenmenge E von G über dem Körper $GF(2)$ mit symmetrischer Differenz als Vektoraddition. Desweiteren sei ein Schnitt im Graphen G definiert durch die Menge $D \subseteq E$ der Kanten, welche diesen Schnitt kreuzen. Die Menge $\mathcal{C}^*$ aller Schnitte von G (inklusive dem leeren Schnitt) ist ein Untervekrorraum von $\mathcal{E}$ (ohne Beweis). Die Kosten einer Basis $B = \{b_1, \ldots, b_d\}$ von $\mathcal{C}^*$ seien definiert als $c(B) := \sum_{i=1}^{d} c(b_i)$ mit $c(b_i)$ Anzahl der Kanten, die Schnitt b_i kreuzen.

(a) Formulieren Sie die Partitionendarstellung des Schnitts $s_3 := s_1 \oplus s_2$ (mit $s_1, s_2 \in C^*$, $s_1 := (S, V \setminus S), s_2 := (T, V \setminus T))$ in Abhängigkeit der Schnittseiten S und T.

Lösung.

Die symmetrische Differenz ergibt für $s_3 := s_1 \oplus s_2 = (s_1 \cup s_2) \setminus (s_1 \cap s_2)$ Nach Definition umfasst die Kantenmenge s_1 alle Kanten, die genau ein Ende in S haben, die Kantenmenge s_2 beinhaltet jene Kanten mit genau einem Ende in T. Das heißt, $s_1 \cap s_2$ ist die Menge der Kanten, die genau ein Ende in S und ein Ende in T haben. Für jede Kante $e = \{u, v\} \in s_1 \cap s_2$ gilt also oBdA: $u \in B := S \setminus T$, $v \in C := T \setminus S$ oder $u \in A := S \cap T$, $v \in D := V \setminus (S \cap T)$. All diese Kanten sind nicht mehr in s_3 enthalten, B, C und A, D liegen damit jeweils auf einer gemeinsamen Schnittseite von s_3. Die übrigen Kanten aus $s_1 \cup s_2$ bleiben erhalten und somit bilden A und D die gegenüberliegende Schnittseite von $B \cup C$. Insgesamt gilt: $s_3 = (B \cup C, A \cup D) = ((S \setminus T) \cup (T \setminus S), (S \cap T) \cup (V \setminus T \cap V \setminus S))$.



(b) Zeigen Sie: Für je zwei Knoten $u, v \in V$ und jede Basis B von $\mathcal{C}^*$ gilt, dass mindestens ein Schnitt aus B die Konten u und v trennt.

Lösung.

Betrachte den Schnitt $(\{u\}, V \setminus \{u\})$. Dieser Schnitt trennt u und v und ist in $\mathcal{C}^*$, muss somit also als Linearkombination bzgl. jeder Basis B von $\mathcal{C}^*$ darstellbar sein. Annahme: In $B = \{b_1, \ldots, b_d\}$ gibt es **keinen** Schnitt b_i, der u und v trennt. Dann kann $(\{u\}, V \setminus \{u\})$ bzgl. B nicht linear kombiniert werden, da die symmetrische Differenz zweier Schnitte s_1 und s_2, die beide u und v nicht teilen, ebenfalls u und v nicht teilt (was zu zeigen ist). Seien s_1, s_2, s_3

definiert wie in (a), wobei s_1, s_2 die Knoten u, v nicht trennen, das heißt, $\{u, v\}$ ist Teilmenge eines der Quadranten A, B, C oder D. Mit (a) folgt dann sofort, dass s_3 ebenfalls u, v nicht trennt.

(c) Zeigen Sie: Untenstehender Algorithmus ist ein polynomieller, relativer 2-Approximationsalgorithmus für das Optimierungsproblem MIN-SCHNITT-BASIS, welches eine minimal gewichtete Basis von $\mathcal{C}^*$ sucht. (Hinweis: Setzen Sie voraus, dass die Ausgabe B' tatsächlich eine Basis von $\mathcal{C}^*$ ist).

Algorithmus 1 : APPROX-MIN-SCHNITT-BASIS

Eingabe : Graph $G = (V, E)$
Ausgabe : Basis B' des Schnittraumes $\mathcal{C}^*$ von G

Wähle einen Knoten $v \in V$
$B' \leftarrow \emptyset$
forall $v' \in V \setminus \{v\}$ **do**
    $b_{v'} \leftarrow \{e \in E | e \text{ inzident zu } v'\}$
    $B' \leftarrow B' \cup \{b_{v'}\}$
Return B'

Lösung.

Die Basis B', die der oben angegebene Algorithmus berechnet, enthält pro Knoten v' den Schnitt $b_{v'} = (\{v'\}, V \setminus \{v'\})$, der v' von allen anderen Knoten trennt. Die Dimension des Schnittraumes ist also $|V| - 1$. Jede Kante $e = \{u, w\}$ kreuzt in B' genau die beiden Schnitte b_u und b_w, trägt also zum Gesmatgewicht der Basis B' genau den Wert 2 bei.

Sei B eine minimal gewichtete Basis von $\mathcal{C}^*$. So gilt $c(B) \leq c(B')$. Außerdem kreuzt jede Kante $e = \{u, w\}$ mindestens einen Schnitt in B, da nach (b) mindestens ein Schnitt in B die Knoten u, w trennt. Jede Kante trägt also zum Gesamtgewicht von B mindestens den Wert 1 bei. Damit gilt: $\mathcal{A}(\mathcal{I}) = c(B') \leq 2\, c(B) = 2\, \mathrm{OPT}(\mathcal{I})$. Die Laufzeit ist offensichtlich in $O(2|E|)$.

Problem 2: APPROX-SUBSET-SUM – FPAS [vgl. Kapitel 7.2 im Skript]

Das Optimierungs-SUBSET-SUM-Problem sucht zu einer Menge $S := \{x_1, \ldots, x_n\} \subset \mathbb{N}$ und einem Wert $t \in \mathbb{N}$ eine Teilmenge $M \subseteq S$, so dass die Summe z aller Elemente in M maximal, aber nicht größer als t, ist. Der folgende Algorithmus gibt abhängig von ϵ einen Wert $z' \leq t$ einer Teilmengensumme aus. Zeigen Sie: Dieser Algorithmus ist ein FPAS für den Wert z einer optimalen Lösung von Subset-Sum (Bemerkung: Ohne Zeile 5 liefert der Algorithmus einen optimalen Wert z).

Algorithmus 2 : Approx-Subset-Sum(S, t, ϵ)

Ausgabe : Teilmengensumme z'

1 $n \leftarrow |S|$
2 $L_0 \leftarrow \langle 0 \rangle$
3 **for** $i = 1, \ldots, n$ **do**
4 $\quad L_i \leftarrow$ Merge-Lists($L_{i-1}, L_{i-1} + x_i$) // addiert x_i zu jedem Element // Merge vereinigt Listen unter // Erhaltung der aufsteigenden Sortierung
5 $\quad L_i \leftarrow$ Trim($L_i, \epsilon/2n$)
6 $\quad$ Entferne alle Elemente größer t aus L_i
7 Return größtes (letztes) Element z' in L_n

Algorithmus 3 : Trim($L := \langle y_1 \ldots, y_m \rangle, \delta$)

Ausgabe : verkürzte Liste L'

1 $L' \leftarrow \langle y_1 \rangle$
2 $last \leftarrow y_1$
3 **for** $i = 2, \ldots, m$ **do**
4 $\quad$ **if** $y_i > last \cdot (1 + \delta)$ **then** // $y_i \geq last$, da L sortiert
5 $\quad\quad$ Füge y_i an L' an
6 $\quad\quad last \leftarrow y_i$
7 Return L'

Lösung.

Ohne die Kürzung in Zeile 5 berechnet Approx-Subset-Sum in Liste L_i iterativ die Summe jeder Teilmenge aus $\mathcal{P}(\{x_1, \ldots, x_i\})$, die nicht grösser als t ist und soriert diese Werte aufsteigend (ohne Beweis). Sei $P_i := \{\sum_{m \in M} m \leq t | M \in \mathcal{P}(\{x_1, \ldots, x_i\})\}$. Die gesuchte optimale Teilmengensumme z ist in L_n als grösstes Element enthalten. Nach Konstruktion der L_i gilt für jedes Element $\gamma \in P_i$:

$$\exists \hat{\gamma} \in P_{i-1}, \text{ so dass } \quad \gamma = \hat{\gamma} + x \quad \text{ mit } x \in \{0, x_i\} \text{ gilt.} \tag{1}$$

Mit der Kürzung in Zeile 5 kann es passieren, dass Vorgängerwerte, aus denen sich ohne Kürzung später (gemäß(1)) das optimale Element z entwickeln würde, gelöscht werden. Die Frage ist nun, wie weit man sich dadurch maximal vom optimalen Ergebnis entfernen kann. Betrachte also Trim genauer.

Für jedes Element γ, das aus L_i gelöscht wird, kann das bis zur Löschung zuletzt in L'_i eingefügte Element als Repräsentant $r_i(\gamma)$ betrachtet werden, d.h. $r_i(\gamma) \in L'_i$ repräsentiert γ in der gekürzten Liste. Falls γ nicht gelöscht wird sei $r_i(\gamma) := \gamma$. Für alle $\gamma \in L_i$ (vor der Kürzung) gilt dann:

$$\gamma \leq r_i(\gamma) \cdot (1 + \delta) \quad \text{und damit} \quad \frac{\gamma}{1 + \delta} \leq r_i(\gamma) \leq \gamma \quad \text{und damit} \quad \frac{\gamma}{r_i(\gamma)} \leq (1 + \delta) \tag{2}$$

Eigentlich wollen wir eine Aussage treffen über der Verhältniss von $z =: \text{OPT}(\mathcal{I})$ und $z' =: \mathcal{A}(\mathcal{I})$, wobei $z \in P_n$ und $z' \in L'_n$, denn wir möchten ja ein Approximationsschema beweisen, d.h. $\frac{z}{z'} \leq (1 + \epsilon)$. In (2) gilt zwar $\gamma \in P_i$, allerdings ist γ nur eingeschränkt aus $L_i \subseteq P_i$ wählbar. Für $i = 1$ (d.h. $L_1 = P_1$) und $r_1(\gamma) =: s$ gilt jedoch:

$$\forall \gamma \in P_1 \ \exists s \in L'_1 \quad \text{, so dass} \quad \frac{\gamma}{1 + \delta} \leq s \leq \gamma \text{ gilt.}$$

Für $\gamma \in P_2$ entsteht wieder das Problem, dass $L_2 = P_2$ aufgrund der eventuellen Kürzung von L_1 **nicht** mehr gilt. Aber mit (1) gilt: $\exists \hat{\gamma} \in P_1$, so dass $\gamma = \hat{\gamma} + x$ mit $x \in \{0, x_i\}$. Damit folgt

$$\begin{aligned} \gamma = \hat{\gamma} + x &\geq s + x \\ &\geq \frac{\hat{\gamma}}{1+\delta} + x \\ &\geq \frac{\hat{\gamma}+x}{1+\delta} = \frac{\gamma}{1+\delta} \text{ mit } s + x \in L_2 \text{ (vor Kürzung).} \end{aligned}$$

Mit (2) gilt: $(s+x)/(1+\delta) \leq r_2(s+x) \leq s + x$ und somit

$$\begin{aligned} \gamma = \hat{\gamma} + x &\geq s + x \\ &\geq r_2(s+x) \\ &\geq \frac{s+x}{1+\delta} \\ &\geq \frac{\gamma}{(1+\delta)^2} \text{ mit } r_2(s+x) \in L_2' \text{ (nach Kürzung).} \end{aligned}$$

Induktiv (Induktionsschritt $i \to i+1$ analog zu $1 \to 2$) ergibt sich somit:

$$\forall \gamma \in P_i \; \exists s \in L_i', \text{ so dass } \frac{\gamma}{(1+\delta)^i} \leq s \leq \gamma \text{ gilt.}$$

Mit $i = n$, $\gamma = z$, $z' \geq s \in L_n'$ und $\delta = \epsilon/2n$ folgt:

$$\frac{z}{z'} \leq \left(1 + \frac{\epsilon}{2n}\right)^n$$

Um nun $(1+\epsilon/2n)^n$ gegen $(1+\epsilon)$ abzuschätzen benutzen wir die Reihen-Entwicklung $e^x = \sum_{i=0}^{\infty} x^i/i!$ der Exponentialfunktion. Denn es ist

$$\begin{aligned} \left(1 + \frac{\epsilon}{2n}\right)^n &= \sum_{i=0}^{n} \binom{n}{i} \left(\frac{\epsilon}{2n}\right)^i \\ &= \sum_{i=0}^{n} \left(\frac{n!}{i!(n-i)!}\right) \left(\frac{\epsilon}{2n}\right)^i \\ &= \sum_{i=0}^{n} \left(\frac{n!}{n^i(n-i)!}\right) \left(\frac{(\epsilon/2)^i}{i!}\right) \\ &\underset{(*)}{\leq} \sum_{i=0}^{n} \left(\frac{(\epsilon/2)^i}{i!}\right) \\ &\leq \sum_{i=0}^{\infty} \left(\frac{(\epsilon/2)^i}{i!}\right) = e^{\epsilon/2} \end{aligned}$$

(*) mit $\left(\frac{n!}{n^i(n-i)!}\right) \leq 1$

Insgesamt ergibt sich damit

$$\begin{aligned} \frac{z}{z'} \leq \left(1 + \frac{\epsilon}{2n}\right)^n &\leq e^{\epsilon/2} \\ &\underset{(**)}{\leq} 1 + \frac{\epsilon}{2} + \left(\frac{\epsilon}{2}\right)^2 \\ &\leq 1 + \epsilon \end{aligned}$$

(**) mit $e^x \leq 1 + x + x^2$ für $|x| \leq 1$ (bekannt aus der Analysis).

Nun soll aus dem bewiesenen Approximationsschema ein FPAS werden, d.h. wir müssen etwas über die Laufzeit aussagen. Die Laufzeit von Approx-Subset-Sum und Trim ist polynomiell in der Länge der verarbeiteten Listen (und damit auch in der Eingabelänge $\geq$ Listenlänge). Jede neue Liste in Zeile 4 in Approx-Subset-Sum ist maximal zweimal so lang wie die vorige Liste. Untersuche also die Entwicklung der gekürzten Listenlängen: Laut Zeile 4 in Trim unterscheiden sich zwei aufeinanderfolgende Elemente y_j und y_{j+1} in einer gekürzten Liste L_i' mindestens um einen Faktor $1 + \delta$. Nach der Löschung aller Elemente größer t in Zeile 6 (Approx-Subset-Sum) enthält dann die gekürzte Liste L_i (außer 0 und eventuell 1) noch maximal $\lfloor \log_{1+\delta} t \rfloor$ Elemente. Die Listenlänge nach der Kürzung ist also allgemein beschränkt durch

$$\begin{aligned}
(\log_{1+\delta} t) + 2 \quad &= \quad \frac{\ln t}{\ln(1 + \epsilon/2n)} + 2 \\
&\underset{(***)}{\leq} \quad \frac{2n(1 + \epsilon/2n) \ln t}{\epsilon} + 2 \\
&< \quad \frac{3n \ln t}{\epsilon} + 2 \text{ (für } \epsilon < 1)
\end{aligned}$$

$(***)$ mit $\ln(1 + x) \geq x/(1 + x)$ für $-1 < x$ (bekannt aus der Analysis).
Diese obere Schranke ist polynomiell in der Eingabelänge (Mindestlänge der Eingabe: $n + \lg t$) und $1/\epsilon$.

Problem 3: Gleichverteiltes JA/NEIN – Randomisierte Algorithmen [vgl. Kapitel 8 im Skript]

Gegeben sei ein Algorithmus Biased-Random der mit Wahrscheinlichkeit p den Wert 1 ausgibt und mit $(1-p)$ den Wert 0. Geben Sie einen Algorithmus an, der Biased-Random als Methode benutzt und jeweils mit Wahrscheinlichkeit 1/2 den Wert 0 oder 1 ausgibt. Analysieren Sie außerdem die erwartete Laufzeit Ihres Algorithmus in Abhängigkeit von p.

Lösung.

Die nötige Symmetrie der Wahrscheinlichkeit erhält man durch Multiplikation: $p(1 - p) = (1 - p)p$. Diese Wahrscheinlichkeiten implizieren eine UND-Verknüpfung zweier Elementarereignisse. Bei zweimaliger unabhängiger Ausführung von Biased-Random bezeichne x den ersten Rückgabewert, y den zweiten. Dann gilt:

$$P(\{x = 0, y = 1\}) = (1 - p)p \text{ und } P(\{x = 1, y = 0\}) = p(1 - p).$$

Ein Algorithmus, der jeweils mit Wahrscheinlichkeit 1/2 den Wert 0 oder 1 ausgibt, sieht also folgendermaßen aus:

Algorithmus 4 : Unbiased-Random

while $x = y$ **do**
  $x \leftarrow$ Biased-Random
  $y \leftarrow$ Biased-Random
return x

Die erwartete Laufzeit von Unbiased-Random in Abhängigkeit von p ist bestimmt durch die Anzahl der zu erwartenden Schleifendurchläufe. Ein einzelner Schleifendurchlauf kann als Bernoulli-Experiment aufgefasst werden, bei dem mit Wahrscheinlichkeit $2p(1-p)$ ein Erfolg ($x \neq y$) eintritt. Die gesamte Schleife ist damit eine Aneinanderreihung unabhängiger Bernoulli-Experimente und damit durch die Geometrische Verteilung gekennzeichnet. Das heißt, der Erwartungswert der Zufallsvariable X für die Anzahl der Durchläufe bis zum ersten Erfolg (inklusive) ist $E(X) = 1/(2p(1-p))$. Damit ist die erwartete Laufzeit von Unbiased-Random in $\Theta(1/(2p(1 - p)))$.