

Analyse von Algorithmen und Komplexität -

Komplexitätsteil: 1. Klausur

Matrikelnummer: 1025242

Name: Marek Lenczewski

Versuch: 3

Aufgabe	1	2	3	4	5	Σ
Punkte	9	10	10	4.5	10	43.5
			7.5			41

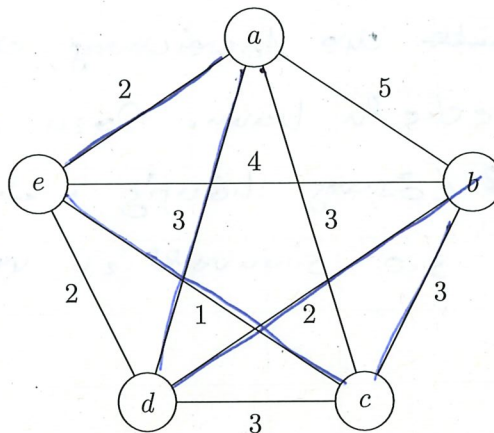
Note: 1.0

1.0

Aufgabe 1 Komplexität - Anwendung

0 9 0 / 10 Punkte

Gegeben sei folgender Graph $G = (V, E)$, bei dem die Kantengewichte jeweils an der nach außen gerichteten Seite der Kante stehen:



- Wenden Sie den Algorithmus ΔTSP_1 auf den Graphen G beginnend bei Knoten a an. Geben Sie dabei die Ergebnisse jedes Zwischenschrittes des Algorithmus an. (6P)
- Geben Sie den Wert einer optimalen Lösung für metrisches TSP für den obigen Graphen an. Ihre Antwort sollte nur aus einer Zahl bestehen. (1P)
- Der Algorithmus ΔTSP_1 berechnet eine TSP-Tour für metrische Instanzen. Geben Sie die Definition einer TSP-Tour an. (1P)
- Begründen Sie, warum man sich (auch) für approximative Algorithmen für metrisches TSP interessiert anstatt immer einen Algorithmus zur Berechnung einer optimalen Lösung zu nutzen. (2P)

Lösung:

a) ΔTSP_1 ✓

Kanten verordnen ✓

Eulerkreis ✓

Abkürzen ✓

b) 11 ✓

c) Rundreise über alle Knoten, wobei jeder Knoten nur einmal besucht wird. (Symmetrie + 1-Ungleichung muss gelten.) ✓

6/6

1/1

1/1

INF Informatik

d) Weil eine approximative Lösung die "gut genug" ist, häufig ausreicht, besonders wenn man ~~sa~~ durch die Güte die Abweichung durch, die Güte ~~ab~~ deckeln kann. Dazu ist die optimale ~~&~~ Lösung häufig nicht schnell genug, um sie sinnvoll zu verwenden.

Warum
→ NP-vollst.

1/2



Aufgabe 2 Komplexität - Vorlesungsbeweis

10 Punkte

Zur Erinnerung betrachten wir die Definitionen der Probleme SubsetSum und Partition.

Problem: SubsetSum

Eingabe: Ganze Zahlen $c_1, \dots, c_n \in \mathbb{Z}$ und $K \in \mathbb{Z}$

Entscheide: Gibt es $I \subseteq \{1, \dots, n\}$ mit $\sum_{i \in I} c_i = K$?

Problem: Partition

Eingabe: Ganze Zahlen $c_1, \dots, c_n \in \mathbb{Z}$

Entscheide: Gibt es $I \subseteq \{1, \dots, n\}$ mit $\sum_{i \in I} c_i = \frac{1}{2} \sum_{i=1}^n c_i$?

Aus der Vorlesung kennen Sie den Beweis für den folgenden Satz.

Satz (Satz 6.33 aus der Vorlesung). Partition ist NP-vollständig, sogar dann, wenn $c_1, \dots, c_n \geq 0$.

- Zeigen Sie, dass Partition in NP liegt. (2P)
- Um zu zeigen, dass Partition NP-schwer ist, führen Sie den folgenden Beweis zuende: (8P)

Beweis. Aus der Vorlesung wissen wir, dass SubsetSum sogar dann NP-schwer ist, wenn alle Zahlen nicht-negativ sind. Insofern sei eine SubsetSum-Instanz $I = (c_1, \dots, c_n, K)$ mit $c_1, \dots, c_n \in \mathbb{Z}_{\geq 0}$ und $K \in \mathbb{Z}_{\geq 0}$ gegeben.

- (Bonusaufgabe) Zeigen Sie, dass SubsetSum sogar dann NP-schwer bleibt, wenn jede Zahl in der Eingabe höchstens zweimal vorkommt. (6P)

Lösung:

a) Eingabe: $(c_1, \dots, c_n)$

Zertifikat: I

Algorithmus:

- Berechne $N = \sum_{i=1}^n c_i$ und $K = \sum_{i \in I} c_i$
- Prüfe ob $K = \frac{1}{2} N$, lehne ab wenn nein, sonst akzeptiere

Existiert eine Partition, dann auch ein Zertifikat, dass akzeptiert und sonst bei der Prüfung ablehnt.

Summieren und prüfen läuft in $O(n)$, polynomial.

Partition $\in$ NP.



3) Reduktion: Subset Sum $\leq$ Partition

Konstruktion: Sei $(c_1, \dots, c_n, k)$ eine Subset Sum Instanz mit $c_1, \dots, c_n \in \mathbb{Z}_{\geq 0}$ und $k \in \mathbb{Z}_{\geq 0}$.

$$\text{Setze } N = \sum_{i=1}^n c_i + 1.$$

$$c_{n+1} = N - k \quad c_{n+2} = k + 1$$

Dann ist $(c_1, \dots, c_n, c_{n+1}, c_{n+2})$ eine Partition

$$\text{Instanz. Dann ist } \sum_{i=1}^{n+2} c_i = \sum_{i=1}^n c_i + (N-k) + (k+1) = 2N$$

$\Rightarrow$ Sei S eine Subset Sum für k . Dann ist

$$S' := S \cup \{c_{n+1}\} \text{ eine Partition, da } \sum_{i \in S'} c_i = k + (N-k) = N \text{ und } N = \frac{1}{2} \sum_{i=1}^{n+2} c_i.$$

$\Leftarrow$ Sei S' eine Partition. Wegen $c_{n+1} + c_{n+2} = (N-k) + (k+1) = N+1$ müssen sie sich auf die

Partitionshälften aufteilen. Sei also $c_{n+1} \in S'$.

Sei also c_{n+1} in S' also $S' = S \cup \{c_{n+1}\}$,

$$\text{gilt für } S' \setminus \{c_{n+1}\} = \sum_{i \in S} c_i - c_{n+1} = N - (N-k)$$

$$= k. \text{ Also ist } S \text{ eine Subset Sum für } k.$$

Laufzeit: Summieren und Zahlen hinzufügen $O(n)$,

polynomiell. Also ist

Partition NP-Schwer.

Schön 😊



Aufgabe 3 Komplexität: NP-Vollständigkeit

10 Punkte
7.5

Wir betrachten folgendes Problem:

Problem: Connected-Clique

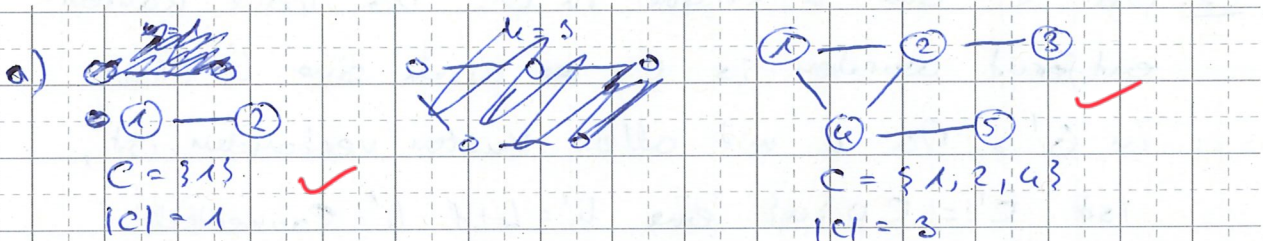
Eingabe: Ein ungerichteter Graph $G = (V, E)$ ohne isolierte Knoten, d.h. für jeden Knoten $u \in V$ existiert ein $v \in V$ mit $\{u, v\} \in E$, sowie eine Zahl $k \in \mathbb{Z}_{\geq 1}$.

Entscheide: Gibt es eine Teilmenge $C \subseteq V$ von Knoten für die die beiden folgenden Eigenschaften gelten:

- C ist eine Clique mit $|C| \geq k$, und
- für alle $v \in V$ gibt es ein $u \in C$ mit $\{u, v\} \in E$.

- Geben Sie jeweils eine Ja-Instanz von Connected-Clique für $k = 1$ und $k = 3$ an. Es genügen Zeichnungen. (2P)
- Zeigen Sie, dass Connected-Clique NP-vollständig ist. (8P)

Lösung:



b) Eingabe: $(G = (V, E), k)$

Zertifikat: $C \subseteq V$

Algorithmus:

- Prüfe $|C| \geq k$, lehne ab wenn ja ✓
- Prüfe für alle $v \in V$ ob $\exists u \in C$ mit $\{u, v\} \in E$, lehne ab wenn nein ✓
- Prüfe für alle $v, w \in C$ ob $\{v, w\} \in E$, lehne ab wenn nein ✓
- Sonst akzeptiere ✓

Existiert eine Connected-Clique, dann auch ein Zertifikat, das akzeptierendes Zertifikat dafür. Anderfalls schlägt eine Prüfung fehl und es wird abgelehnt.

Die Prüfungen laufen in $O(|V|)$ und $O(|V|^2)$,

Gesamt $O(|V|^2)$, polynomiell. Connected-Clique ✓

INF
INformatik

Reduktion: $\text{Clique} \leq \text{Connected-Clique}$

Konstruktion: Sei $(G=(V,E), k)$ eine k -Clique Instanz.

- $V' = V \cup \{u\}$ ✓
- $E' = E \cup \{\{v,u\} \mid v \in V\}$ ✓
- $k' = k+1$ ✓

Dann ist $(G'=(V',E'), k')$ eine k' -Connected-Clique Instanz.

$\Rightarrow$ Sei C eine k -Clique in G . Da keine Kanten entfernt wurden ist C auch eine k -Clique in G' . Da u mit allen Knoten verbunden ist, ist $C' := C \cup \{u\}$ eine $k' = k+1$ k' -Connected-Clique in G' . ✓

$\Leftarrow$ Sei C' eine k' -Connected-Clique in G' .

Entferne einen beliebigen Knoten x in C' . f -2

Dann ist $C := C' \setminus \{x\}$ eine $k = k' - 1$ k -Clique in G . ✓

Laufzeit: $\mathcal{O}(n)$, $\mathcal{O}(n^2)$, $\mathcal{O}(n)$ und $\mathcal{O}(n^2)$, für jeden Knoten wird eine Kante hinzugefügt, Gesamt $\mathcal{O}(n^2)$, polynomiell. Also ist

Da ~~Clique~~ NP-Vollständig Connected-Clique NP-Schwer. ✓

Da Clique NP-Vollständig ist, Connected-Clique $\in$ NP und NP-Schwer, ist Connected-Clique NP-Vollständig. ✓

Aufgabe 4 Komplexität - Approximative Algorithmen

4 · 5 / 10 Punkte

Wir erinnern uns an das Scheduling-Problem $P||C_{\max}$ aus der Vorlesung:

Problem: Scheduling-Problem $P||C_{\max}$

Eingabe: n Jobs/Aufträge mit Ausführungszeiten $p_1, \dots, p_n \in \mathbb{Z}_{\geq 0}$ und m Maschinen

Gesucht: Eine Partition von $\{1, \dots, n\}$ in Teilmengen $B_1, \dots, B_m$ mit minimaler maximaler Last

$$C_{\max} = \max_{i \in \{1, \dots, m\}} \sum_{j \in B_i} p_j$$

Betrachten Sie den folgenden Algorithmus für $P||C_{\max}$:

Algorithmus PRIMITIVE ($I = (n, p_1, \dots, p_n, m)$)

```

1  for  $i = 1$  to  $m$  do
2       $B_i = \emptyset$ 
3  od
4   $P = 0$ 
5  for  $j = 1$  to  $n$  do
6       $P = P + p_j$ 
7  od
8   $i = 1$ 
9   $L = 0$ 
10 for  $j = 1$  to  $n$  do
11      $B_i = B_i \cup \{j\}$ 
12      $L = L + p_j$ 
13     if  $L \geq \frac{1}{m}P$  then
14          $i = i + 1$ 
15          $L = 0$ 
16     fi
17 od
18 return  $(B_1, \dots, B_m)$ 

```

Für jede Instanz I bezeichnen wir mit $\mathbf{P}(I)$ die maximale Last einer Maschine in der von **PRIMITIVE** für I berechneten Lösung $(B_1, \dots, B_m)$, d.h. $\mathbf{P}(I) = \max_{i \in \{1, \dots, m\}} \sum_{j \in B_i} p_j$.

- a) Beschreiben Sie in höchstens drei Sätzen, was **PRIMITIVE** tut. **Antworten mit vier oder mehr Sätzen werden mit 0 Punkten bewertet.** (1P)
- b) Argumentieren Sie dafür, dass **PRIMITIVE** wohldefiniert ist. Das bedeutet hier ganz konkret, dass (während einer Berechnung) in Zeile 11 immer $i \leq m$ gilt. (3P)
- c) Zeigen Sie, dass **PRIMITIVE** ein approximativer Algorithmus mit Güte 2 für $P||C_{\max}$ ist, d.h. $P(I) \leq 2 \cdot \text{OPT}(I)$ für alle Instanzen I von $P||C_{\max}$. Sie dürfen hier insbesondere aus der Vorlesung bekannte Schranken für $\text{OPT}(I)$ nutzen. (6P)
- d) (Bonusaufgabe) Verbessert sich die Güte von **PRIMITIVE**, wenn die Jobs in der Eingabe absteigend nach ihrer Ausführungszeit sortiert sind? Zeigen Sie eine bessere Güte oder geben Sie eine Beispielinstanz an, die zeigt, dass sich die Güte nicht verbessert. (4P)

Lösung:

[illegible]

a) n Jobs auf m Buckets verteilen, wobei nur der letzte Job in jedem Bucket über der Grenze $\frac{1}{m} \sum_{i=1}^n p_i$ hinaus gehen kann. ~~aber~~ ok.

1/n

b) Für die ersten $m-1$ Buckets gilt $\sum_{i \in S_j} p_i \geq \frac{1}{m} P$ für $j \in [m-1]$, also die ersten $m-1$ Buckets sind mindestens $\frac{1}{m} P$ groß. Dadurch ist m Bucket höchstens $\frac{1}{m} P$ groß. Somit stoppt die for-Schleife (2.10) ^(frühestens) mit dem letzten Job wenn $i \leq m$.

2/5/3

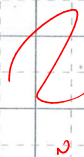
c) 2.2. $P(I) \leq 2 \text{OPT}(I)$

~~Schrittweise~~ ~~letzte~~ ~~Job~~ ~~zum~~ ~~Bm~~

$\text{OPT}(I) \geq \frac{1}{m} P$: Jeder Job muss irgendwo hinlaufen ✓

$\text{OPT}(I) \geq p_{\max}$: Jeder Job muss verarbeitet werden ✓

$$P(I) \leq$$



$$2 \text{OPT}(I)$$

1/6



Aufgabe 5 Komplexität - ETH

1 0 0 / 10 Punkte

a) Das Problem 1-aus-3-SAT sei wie folgt definiert.

Problem: 1-aus-3-SAT

Eingabe: Ein boolescher Ausdruck in KNF über Variablen $\alpha_1, \dots, \alpha_n$ und Klauseln $K_1, \dots, K_m$, wobei $|K_i| \leq 3$ für alle $i \in \{1, \dots, m\}$.

Entscheide: Existiert eine Belegung φ , sodass für jede Klausel K_i exakt ein Literal $y \in K_i$ existiert mit $\varphi(y) = 1$.

Betrachten Sie folgende Reduktion von 3-SAT auf 1-aus-3-SAT. Für eine 3-SAT Instanz über n Variablen $x_1, \dots, x_n$ und m Klauseln $C_1, \dots, C_m$ erstelle eine 1-aus-3-SAT Instanz wie folgt.

- Die Menge der Variablen sei $\{x_i \mid i \in \{1, \dots, n\}\} \cup \{\hat{x}_{k,\ell} \mid k \in \{1, \dots, 6\}, \ell \in \{1, \dots, m\}\}$.
- Definiere für jede Klausel C_j mit Literalen y_{j_1}, y_{j_2} und y_{j_3} aus der 3-SAT Instanz folgende Klauseln für die 1-aus-3-SAT Instanz

- $y_{j_1} \vee \hat{x}_{1,j} \vee \hat{x}_{4,j},$
- $y_{j_2} \vee \hat{x}_{2,j} \vee \hat{x}_{4,j},$
- $y_{j_3} \vee \hat{x}_{3,j},$
- $\hat{x}_{1,j} \vee \hat{x}_{2,j} \vee \hat{x}_{5,j}$ und
- $\hat{x}_{3,j} \vee \hat{x}_{4,j} \vee \hat{x}_{6,j}.$

- Geben Sie die Anzahl der Variablen und die Anzahl der Klauseln für die resultierende 1-aus-3-SAT Instanz in Abhängigkeit von der originalen 3-SAT Instanz an. (1P)

⚠ Eine (möglichst genaue) Angabe in O -Notation genügt.

- Zeigen Sie Lower Bounds für 1-aus-3-SAT bezüglich Anzahl der Variablen und Anzahl der Klauseln unter Annahme der ETH und der oben beschriebenen Reduktion. (4P)

b) Das Problem FeedbackArcSet sei wie folgt definiert.

Problem: Feedback Arc Set

Eingabe: Ein gerichteter Graph $\hat{G} = (\hat{V}, \hat{E})$ und eine Zahl $\hat{k}$.

Entscheide: Existiert eine Menge von Kanten $S \subseteq \hat{E}$ mit $|S| \leq \hat{k}$, sodass der Graph ohne die Kanten in S , formell der Graph $G' = (V, E \setminus S)$, kreisfrei ist?

Betrachten Sie folgende Reduktion von VertexCover auf FeedbackArcSet. Für eine Eingabe $(G = (V, E), k)$ erstelle die Instanz $(\hat{G} = (\hat{V}, \hat{E}), \hat{k})$ wie folgt.

- Definiere $\hat{V} := \{\hat{v}_v^j \mid v \in V, j \in \{1, 2\}\}$.
- Für jede Kante $\{u, v\} \in E$ definiere Kanten $(\hat{v}_u^2, \hat{v}_v^1)$ und $(\hat{v}_v^2, \hat{v}_u^1)$ in $\hat{E}$. Definiere weiterhin für jeden Knoten $v \in V$ eine Kante $(\hat{v}_v^1, \hat{v}_v^2)$ in $\hat{E}$.
- Definiere $\hat{k} := k$.

- Geben Sie die Anzahl der Knoten und die Anzahl der Kanten für die resultierende FeedbackArcSet Instanz in Abhängigkeit von der originalen VertexCover Instanz an. (1P)

⚠ Eine (möglichst genaue) Angabe in O -Notation genügt.

- Zeigen Sie Lower Bounds für FeedbackArcSet bezüglich Anzahl der Knoten und Anzahl der Kanten unter Annahme der ETH und der oben beschriebenen Reduktion. (4P)

⚠ Sie dürfen alle aus diesem Modul bekannten Lower Bounds für Vertex Cover nutzen, beispielsweise den ETH Lower Bound $2^{o(|V|)} \cdot \text{poly}(|I|)$ aus Hausaufgabe 13.

Lösung:



5a1) $\hat{n} = n + \cancel{O(n)} Gm = 3n + Gm = O(n)$ ✓

$m\hat{n} = O(n)$ ✓

111

2b) Bzgl. $\hat{n}$: Angenommen 1-ans-3-SAT wäre in $2^{O(\hat{n})} \cdot IO(n)$ lösbar. Wegen $\hat{n} = O(n)$ wäre 3-SAT in $2^{O(n)} \cdot IO(n)$ lösbar. Widerspruch zur ETH durch Spezifikations Lemma. ✓

Bzgl. $\hat{m}$: Angenommen 1-ans-3-SAT wäre in $2^{O(\hat{m})} \cdot IO(n)$ lösbar. Wegen $\hat{m} = O(n)$ wäre 3-SAT in $2^{O(n)} \cdot IO(n)$ lösbar. Widerspruch zur ETH durch Spezifikations Lemma. ✓

4/4

5b1) $|\hat{V}| = O(|V|)$

$|\hat{E}| = O(|V|^2)$

✓ 111

5b2) Bzgl. $|\hat{V}|$: Angenommen Feedbacktre Set wäre in $2^{O(|\hat{V}|)} \cdot IO(n)$ lösbar. Wegen $|\hat{V}| = O(|V|)$ wäre Vertex Cover in $2^{O(|V|)} \cdot IO(n)$ lösbar. Widerspruch zur ETH. ✓

Bzgl. $|\hat{E}|$: Angenommen Feedbacktre Set wäre in $2^{O(|\hat{E}|)} \cdot IO(n)$ lösbar. Wegen $|\hat{E}| = O(|V|^2)$ und $|V| = O(|V|)$ wäre Vertex Cover in $2^{O(|V|)} \cdot IO(n)$ lösbar. Widerspruch zur ETH. ✓

4/4

