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

Übungsblatt 6

Vorlesung Theoretische Grundlagen der Informatik im WS 16/17

Ausgabe 22. Dezember 2016
Abgabe 17. Januar 2017, 11:00 Uhr (im Kasten im UG von Gebäude 50.34)

Bitte nutzen Sie den WebInScribe Deckblattgenerator
und heften Sie das Deckblatt an Ihr Übungsblatt.
`https://webinscribe.ira.uka.de/deckblatt/index.php?course=10588`.

Aufgabe 1

(2 + 2 = 4 Punkte)

Das Bin Packing-Problem ist wie folgt definiert:

Gegeben: endliche Menge $M = \{a_1, \ldots, a_n\}$ mit Gewichtungsfunktion $s : M \to (0, 1]$.
Gesucht: Minimale Anzahl an Bins m, so dass für jeden Bin B_i mit $i = 1, \ldots, m$ gilt:

$$\sum_{a \in B_i} s(a) \leq 1$$

In der Übung wurde ein Approximationsalgorithmus zusammen mit einer oberen Schranke von 2 für dessen relative Güte vorgestellt.

(a) Zeigen Sie, dass diese obere Schranke gewissermaßen optimal ist. Geben Sie dazu für jedes $\epsilon > 0$ eine Folge von Elementen an, so dass der Approximationsalgorithmus bei Abarbeitung dieser Folge mindestens $2(1 - \epsilon)$ Bins benötigt.

(b) Die Strategie, um Elemente in die Bins einzufügen, wird nun verändert. Statt nur den letzten Bin zu betrachten, wird jetzt ein Element in den ersten Bin eingefügt, in dem noch ausreichend Platz ist. Zeigen Sie, dass für diesen neuen Approximationsalgorithmus $\mathcal{A}$ gilt $\mathcal{R}_{\mathcal{A}} > \frac{11}{7}$.

Lösung:

(a) Setze $n = 4\lceil \frac{1}{\epsilon} \rceil$ und wähle als Folge der Länge n abwechselnd $\frac{1}{2}$ und $\frac{2}{n}$. Der Approximationsalgorithmus aus der Übung benötigt dann $\frac{n}{2}$ Bins, die optimale Lösung benötigt jedoch nur $\frac{n}{4} + 1$ Bins. Für die relative Güte gilt dann:

$$\mathcal{R} = \frac{\frac{n}{2}}{\frac{n}{4} + 1} = 2\left(\frac{\frac{n}{4} + 1 - 1}{\frac{n}{4} + 1}\right) = 2\left(1 - \frac{1}{\lceil \frac{1}{\epsilon} \rceil + 1}\right) > 2(1 - \epsilon)$$

(b) Wähle zu einem beliebigen $k \in \mathbb{N}$ eine Folge von $18k$ Elementen, wobei die ersten $6k$ Elemente die Größe 0,15 haben, die nächsten $6k$ Elemente 0,34 und die letzten $6k$ Elemente 0,51. Mit der gewählten Strategie werden die ersten $6k$ Elemente in k Bins gepackt, wobei jeder dieser Bins Füllgrad 0,9 hat. Die nächsten $6k$ Elemente werden in $3k$ Bins mit Füllgrad 0,68 gepackt, und die letzten $6k$ Elemente werden in $6k$ Bins gepackt. Insgesamt werden so $10k$ Bins genutzt. Eine optimale Lösung würde jedoch nur $6k$ Bins benötigen. Als relative Güte ergibt sich somit:

$$\mathcal{R} > \frac{10k}{6k} = \frac{5}{3} > \frac{11}{7}$$

Aufgabe 2

(1 + 1 = 2 Punkte)

(a) Sei Π ein $\mathcal{NP}$-vollständiges Problem zu dem ein pseudo-polynomialer Algorithmus existiert. Warum impliziert dies nicht die Existenz eines pseudo-polynomialen Algorithmus für jedes $\mathcal{NP}$-vollständige Problem?

Lösung:
Bei der polynomialen Transformation einer Eingabeinstanz I in eine Instanz I' beschränken wir die Länge $|I'|$ polynomial in der Länge von I. Dies impliziert allerdings nicht, dass auch die größte Zahl $\max(I')$ aus I' polynomial durch I' beschränkt ist. Dies führt unmittelbar dazu das die unär Kodierung der Instanz I nicht polynomial beschränkt ist in der unär Kodierung von I.

(b) Zeigen Sie, dass ein stark $\mathcal{NP}$-vollständiges Problem genau dann von einem pseudo-polynomialen Algorithmus entschieden wird, wenn $\mathcal{P} = \mathcal{NP}$ gilt.

Lösung:
Angenommen es existiert ein pseudo-polynomialer Algorithmus $\mathcal{A}$ für ein stark $\mathcal{NP}$-vollständiges Problem. Dann sind die Anzahl der Bits in der Unärkodierung durch ein Polynom Beschränkt, da $\max(I)$ polynomial beschränkt ist. Mit Hilfe von $\mathcal{A}$ kann somit jedes Problem in $\mathcal{NP}$ in polynomial Zeit entschieden werden.

Angenommen, es gilt $\mathcal{P} = \mathcal{NP}$. Die stark $\mathcal{NP}$-vollständigen Problem eine Teilmenge der $\mathcal{NP}$-vollständigen Probleme. Es existiert also ein Algorithmus $\mathcal{A}$ der in polynomiale Laufzeit in n benötigt. Es bleibt zu zeigen, dass $\mathcal{A}$ auch pseudo-polynomial ist. Sei die Zahl n die Anzahl der Bits in der binär Kodierung (oder ein Kodierungschema das polynomial Transformierbar in die binär Kodierung ist). Die Anzahl der Bits N einer Unär Kodierung ist somit $N \in \Theta(2^n)$. Angenommen, der Algorithmus ist nicht polynomial in N, dann folgt sofort ein Widerspruch zur polynomialen Laufzeit in n. Jeder Polynomialzeit ist also auch ein Pseudo-Polynomialer Algorithmus.

Aufgabe 3

(1 + 2 + 3 = 6 Punkte)

Betrachten Sie den folgenden Algorithmus SIMPLE-MATCHER, der für einen bipartiten Graphen $G = (A \cup B, E)$ ein Matching berechnet.

Ein Matching $M \subset E$ ist eine Menge von Kanten, so dass keine zwei Kanten in M inzident sind. Ein Matching ist *inklusion-maximal*, wenn kein Matching M' existiert, dass M als echte Teilmenge

enthält. Ein Matching ist *kardinalitäts-maximal*, wenn kein Matching mit größerer Kardinalität existiert.

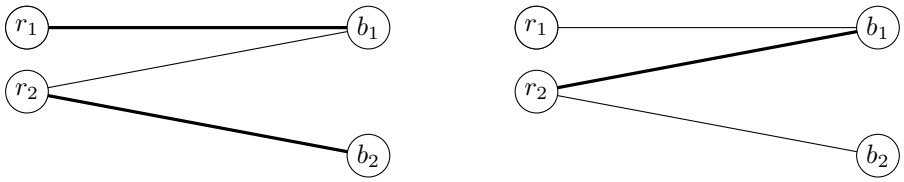
Algorithmus 1 : SIMPLE-MATCHER

Eingabe : Bipartiter Graph $G = (A \cup B, E)$
Ausgabe : Matching M

```
1 M ← ∅;
2 Für u ∈ B
3     Wenn u einen Nachbarn v hat, der zu keiner Kante in M inzident ist,
4         füge Kante {u, v} zu M hinzu.;
5 return M
```

(a) Zeigen Sie mithilfe eines Beispielgraphen $G = (A \cup B, E)$, dass SIMPLE-MATCHER ein Matching M liefern kann, das nur halb so groß ist, wie ein kardinalitäts-maximale Matching $M^\star$ in G. Begründen Sie Ihre Antwort, indem Sie ein kardinalitäts-maximales Matching $M^\star$ angeben, SIMPLE-MATCHER auf G ausführen und in jedem Schritt M und den bearbeiteten Knoten angeben.

Lösung:



Optimal SIMPLE-MATCHER

1. Schritt: Knoten b_1 wird gewählt und die Kante $\{b_1, r_2\}$ zu M hinzugefügt.

2. Schritt: Knoten b_2 wird gewählt und keine Kante hinzugefügt, da r_2 bereits zu b_1 zugeordnet ist.

Damit besteht das Matching M aus der Kante $\{b_1, r_2\}$. Das optimale Matching dagegen enthält die Kanten $\{b_1, r_1\}$ und $\{b_2, r_2\}$.

(b) Sei M ein Matching, das von SIMPLE-MATCHER berechnet wurde. Zeigen Sie, dass M ein inklusions-maximales Matching ist.

Lösung:
Angenommen es gibt eine Kante $\{u, v\} \in E \setminus M$ mit $v \in A$ und $u \in B$, sodass $M \cup \{e\}$ ein Matching ist. Dann besitzen die Knoten v und u in M keine inzidenten Kanten. Dann wurde u beim Aufdecken kein Knoten von A zugewiesen, obwohl der Knoten v noch frei war. Dies ist ein Widerspruch zur Definition des Algorithmus.

(c) Zeigen Sie, dass SIMPLE-MATCHER ein Approximationsalgorithmus mit relativer Gütegarantie 2 für das Problem ein maximales Matching in einem bipartiten Graphen zu berechnen ist.

Lösung:
Sei M ein beliebiges inklusions-maximales Matching in G und sei $M^\star$ ein beliebiges kardinalitäts-maximales Matching in G. Bezeichne nun $V(M) \subseteq V$ die Knoten, die in M eine inzidente Kante besitzen.

Es gilt $|V(M)| = 2|M|$. Außerdem besitzt für jede Kante $\{u, v\} \in M^\star$ mindestens u oder v eine inzidente Kante in M. Ansonsten wäre $M \cup \{\{u, v\}\}$ ein Matching. Damit gilt $|V(M)| \geq |M|^\star$ und $|M| = \frac{|V(M)|}{2} \geq \frac{|M^\star|}{2}$.

Aufgabe 4

(2 + 2 + 3 = 7 Punkte)

Algorithmus 2 : MAXSUM-SCHEMA

Eingabe : $L \subset \mathbb{N}_0$, Zahl $k \in \mathbb{N}$, $\delta \in \mathbb{R}_{\geq 0}$

```
1  S = {0} ;
2  Für x ∈ L
3  |  T ← {x + y | y ∈ S} ;
4  |  X ← S ∪ T ;
5  |  s ←kleinstes Element von X;
6  |  S ← {s} ;
7  |  Für y ∈ X in aufsteigender Reihenfolge
8  |  |  Wenn s · (1 + δ) < y ≤ k
9  |  |  |  S ← S ∪ {y};
10 |  |  |  s ← y;
11 return max_{y⋆∈S} y⋆
```

(a) Bezeichne mit Σ_X die Summe $\Sigma_{x \in X} x$ für eine Menge x. Zeigen Sie, dass für $\delta = 0$ der Algorithmus MAXSUM-SCHEMA die Funktion $\max_{S' \in \{S | S \subset L \wedge \Sigma_S \leq k\}} \Sigma_{S'}$ berechnet. Analysieren Sie die Laufzeit.

Lösung:
Sei $S = \{x_1, x_2, \ldots, x_n\}$, $S_i = \{x_1, x_2, \ldots, x_i\}$. Sei $P_i = \{\Sigma_Q \mid Q \subset S_i \text{ und } \Sigma_Q \leq k\})$.

Wir zeigen per Induktion, dass $Y_i = P_i$ (Y_i wie in Aufgabe b) definiert.

Sei $i = 1$. Ist $x_1 > k$, dann gilt $P_1 = \{0\} = Y_1$. Andernfalls, $P_1 = \{0, x_1\} = Y_1$. Damit ist der Induktionsanfang erfüllt.

Wir nehmen an, dass die Aussage für $i-1$ erfüllt ist, d.h. $P_{i-1} = Y_{i-1}$. Vor der i-ten Iteration gilt also $S = Y_{i-1}$.

Sei $x \in P_i$ enthalten, dann existiert eine Teilmenge $Q \subset S_i$, ist x_i nicht in Q enthalten, dann ist die Aussage nach Induktionvoraussetzung erfüllt. Sonst ist Q die Vereinigung einer Menge $Q' \subset S_{i-1}$ und $\{x_i\}$. Nach Induktionsvoraussetzung gilt $\Sigma_{Q'} \in Y_i$. Da x in P_i enthalten ist, gilt insbesondere $\Sigma_Q \leq k$ und damit auch $x \in Y_i$.

Nach Induktionvoraussetzung ist S eine Teilmenge von P_i, damit ist auch T eine Teilmenge von P_i, somit ist auch $Y_i \subset P_i$.

Laufzeit Analyse: Aufgrund der Vereinigung in Zeile 4, kann sich die Länge der Liste in jedem Schritt verdoppeln, aber nicht mehr als $O(2^n)$ Werte enthalten.

(b) Bezeichne mit Y_i die Menge S nach Iteration i mit $\delta = 0$ und Z_i die entsprechende Menge für ein beliebiges $\delta > 0$. Nehmen Sie für beide Menge $Y_0 = Z_0 = \{0\}$ an. Zeigen Sie, für $y \in Y_i$ existiert ein $z \in Z_i$ mit folgender Eigenschaft:

$$\frac{y}{(1+\delta)^i} \leq z$$

Lösung:
Wir beweisen die Aussage per Induktion. Der Induktionsanfang ist für $i = 0$ per Definition erfüllt.

Wir nehmen an, dass die Aussage für $i-1$ erfüllt ist, d.h. zu jedem $y' \in P_{i-1}$ existiert ein z' mit der geforderten Eigenschaft.

Sei $y \in P_i$. Wir nehmen an, dass x_i an der Summenbildung von y erfüllt ist, sonst greift sofort die Induktionsvoraussetzung. Es existiert ein $y' \in P_{i-1}$ mit $y = y' + x_i$. Nach Induktionsvoraussetzung gibt es zu y' ein z' mit der geforderten Eigenschaft. Es gilt also die folgende Ungleichung:

$$y = y' + x_i \leq (1+\delta)^{i-1} z' + x \leq (1+\delta)^{i-1}(z' + x).$$

Der Wert $z' + x$ muss nicht notwendiger Weise in Y_i enthalten sein, nach Zeile 8 existiert allerdings ein $z \in Y_i$ mit $z' + x \leq (1+\delta)x$. Einsetzten in die Ungleichung zeigt wie gefordert die Behauptung.

(c) Zeigen Sie, dass MaxSum-Schema für ein geeignetes δ ein vollständiges polynomiales Approximationschema ist.

Lösung:
Wir zeigen die Aussage für $\delta = \epsilon/(2n), \epsilon > 0, n = |L|$. Wir zeigen zu erst, dass der Algorithmus eine $(1+\epsilon)$ Approximation berechnet und anschließend, dass der Algorithmus nur polynomiale Laufzeit in $1/\epsilon$ und der Eingabe benötigt.

Approximation

Es gilt zu zeigen, dass die relative Approximationsgüte $\mathcal{R}_{\mathcal{A}}(I) = \mathcal{OPT}(I)/\mathcal{A}(I)$ durch $(1+\epsilon)$ beschränkt ist. Sei $y^\star = \mathcal{OPT}(I)$ und $z^\star = \mathcal{A}(I)$. Nach Aussage b) gibt es zu $y^\star$ ein z' mit $y^\star/z' \leq (1+\delta)^n$, dann gilt die Aussage aber insbesondere auch für das Maximum $z^\star$. Somit bleibt zu zeigen, dass $(1+\epsilon/(2n))^n$ mit $1+\epsilon$ abgeschätzt werden kann. Mit Mitteln aus der Analysis können die folgenden Umformungen bestätigt werden.

$$(1+\epsilon/(2n))^n \leq e^{\epsilon/2} \leq 1+\epsilon$$

Laufzeit

Es bleibt zu zeigen, dass der Algorithmus polynomiale Zeit in der Länge der Liste L benötigt. Hier zu muss insbesondere gezeigt werden, dass die Länge der Liste S polynomial in der Länge der Liste L beschränkt ist.

Zwei Element $x, y \in S$ sind mindestens einen Faktor $(1+\epsilon/(2n))$ voneinander entfernt. Somit enthält S maximal $\log_{1+\epsilon/(2n)} k$ Elemente.

$$\log_{1+\epsilon/(2n)} k \leq \frac{\ln k}{\ln(1+\epsilon/(2n))} \leq \frac{2n(1+\epsilon/(2n))\ln k}{\epsilon} + 2 \leq \frac{3n \ln k}{\epsilon} + 2$$

Die Schranke ist polynomial in n, k und $1/\epsilon$. Alle Operationen des Algorithmus können ebenfalls in polynomieller Zeit implementiert werden, womit gezeigt ist, dass der Algorithmus für unser δ tatsächlich ein FPAS ist.

Aufgabe 5 (2 Punkte)

Formulieren Sie das Problem Exact Cover als Integer Linear Program. Begründen Sie Ihre Modelierung.

Lösung:

Sei $I = (X, S)$ für eine Menge $X = \{x_1, \ldots, x_n\}$ und eine Menge von Teilmengen $S = \{S_1, \ldots, S_m\}$ von X eine Instanz von EXACT COVER. Wir definieren

$$a_{ij} := \begin{cases} 1 & \text{, falls } x_i \in S_j \\ 0 & \text{, falls } x_i \notin S_j \end{cases}$$

für $1 \leq i \leq n$ und $1 \leq j \leq m$. Außerdem seien $A = (a_{ij})$ und $b = (1, \ldots, 1)$. Es existiert genau dann eine Menge $S' \subseteq S$, so dass jedes Element aus X in genau einer Menge aus S' liegt, wenn es einen Vektor $z = (z_1, \ldots, z_m)$ gibt, so dass

$$Az = b$$

gilt. Ein Eintrag des Vektors z kann entweder den Wert 0 oder 1 annehmen und kodiert, welche der Mengen aus S in S' sind. In Matrix-Komponenten-Schreibweise ist das Integer Program dann gegeben als

$$\begin{pmatrix} A \\ -A \end{pmatrix} z \leq \begin{pmatrix} b \\ -b \end{pmatrix} = \begin{pmatrix} \mathbf{1} \\ -\mathbf{1} \end{pmatrix} .$$

Formal müssen wir noch $c_i = 0$ $(1 \leq i \leq m)$ und $B = 0$ spezifizieren. $\sum_{i=1}^{m} c_i x_i = B$ ist dann trivialerweise erfüllt.

Aufgabe 6 (4 Punkte)

Der ebenso geniale wie auch skrupellose Wissenschaftler und Superbösewicht Doktor Meta ist in Weihnachtsstimmung. Entsprechend groß sind seine Pläne für Elsas Geschenk. Nachdem ihm aber selbst die Allianz der Diktatoren die nötige finanzielle Unterstützung verweigert, muss er nun selber Geld verdienen. Ein einfacher Bankraub ist einem Wissenschaftler und Superbösewicht selbstverständlich nicht würdig. Doktor Meta hat eine bessere Idee: er wird den Weihnachtsmann ausrauben! Mit Geschenken für über sieben Milliarden Menschen ist für Metas Zukunft ausgesorgt.

Das Problem an Doktor Metas Plan ist, dass der Weihnachtsmann, wenn er erst einmal auf seiner Route unterwegs ist, schwer zu schnappen ist. Doktor Meta muss deshalb verhindern, dass der Weihnachtsmann überhaupt erst eine Route findet. Sitzt der Weihnachtsmann dann am Nordpol fest, kann Doktor Meta ihn dort, fernab jeglicher Zeugen, ausrauben.

Meta weiß, dass das Traveling Santa-Problem

Gegeben: Graph $G = (V, E)$ vollständig und gewichtet mit Gewichtungsfunktion $c : E \to \mathbb{Q}$. **Gesucht:** Optimale Tour in G bezüglich c.

im Allgemeinen $\mathcal{NP}$-schwer ist und deshalb praktisch nicht gelöst werden kann. Er weiß aber auch, dass das Traveling Santa-Problem *mit* Dreiecksungleichung durch einen polynomialen Algorithmus mit relativer Gütegarantie von 2 approximiert werden kann – eine solche Tour wäre für den Weihnachtsmann immer noch gut genug.

Doktor Metas Blick fällt auf seinen Patentprojektor, mit dem er den euklidischen Raum in den Metaraum projizieren kann, in dem die Dreiecksungleichung nicht gilt. Helfen Sie Doktor Meta, indem Sie zeigen, dass der Weihnachtsmann im Metaraum eine optimale Route nicht einmal mit konstanter relativer Güte approximieren kann.

Zeigen Sie dazu, dass für das Traveling Santa-Problem *ohne* Dreiecksungleichung ein Approximationsalgorithmus mit konstanter relativer Gütegarantie genau dann existiert, wenn $\mathcal{P} = \mathcal{NP}$ gilt. Sie dürfen dabei die $\mathcal{NP}$-Vollständigkeit der folgenden Probleme nutzen:

Induzierter Pfad:

Gegeben: Ungerichteter Graph $G = (V, E)$ und Parameter $k \in \mathbb{N}$. **Gesucht:** Menge $V' \subseteq V$, $\lvert V' \rvert = k$, so dass der durch V' induzierte Subgraph von G ein Pfad ist.

Hamiltonpfad:

Gegeben: Ungerichteter Graph $G = (V, E)$. **Gesucht:** Geschlossener Pfad in G, der jeden Knoten genau einmal enthält.

Längster Pfad:

Gegeben: Ungerichteter Graph $G = (V, E)$. **Gesucht:** Längster Pfad in G.

Hinweis: Überlegen Sie sich, wo genau die Dreiecksungleichung im Beweis aus der Vorlesung genutzt wurde. Welches Problem entsteht, wenn die Dreiecksungleichung nicht mehr gilt?

Lösung:
Angenommen, es existiert ein Approximationsalgorithmus $\mathcal{A}$ für das Traveling Santa-Problem mit konstanter relativer Gütegarantie $\mathcal{R}_{\mathcal{A}}^{\infty}$. Dann kann $\mathcal{A}$ genutzt werden, um das Hamiltonpfad-Problem zu lösen. Gehe dazu wie folgt vor. Sei $G = (V, E)$ ein ungerichteter Graph mit $n = |V|$. Sei $G' = (V, E') = K_n$ der vollständige, ungerichtete Graph auf der Knotenmenge V. Setze

$\alpha \coloneqq \mathcal{R}_{\mathcal{A}}^{\infty} \cdot n + 1$ und definiere $c : E' \to \mathbb{Q}$ als Gewichtungsfunktion mit $c : e \mapsto 1$ falls $e \in E$, und $c : e \mapsto \alpha$ sonst. Dann enthält G genau dann einen Hamiltonpfad, wenn G' eine Tour mit einem Gesamtgewicht von höchstens $\mathcal{R}_{\mathcal{A}}^{\infty} \cdot n$ enthält.

Existiert nämlich ein Hamiltonpfad in G, dann existiert auch eine Tour in G' mit Gesamtgewicht n. Mit der konstanten relativen Güte gilt dann $\mathcal{A}(G') \leq \mathcal{R}_{\mathcal{A}}^{\infty} \cdot n$.

Gilt umgekehrt $\mathcal{A}(G') \leq \mathcal{R}_{\mathcal{A}}^{\infty} \cdot n$, existiert aufgrund der relativen Gütegarantie eine Tour in G mit Gesamtgewicht höchstens $\mathcal{R}_{\mathcal{A}}^{\infty} \cdot n$. Diese Tour kann keine Kante $e \notin E$ enthalten, denn sonst wäre das Gesamtgewicht mindestens $\alpha = \mathcal{R}_{\mathcal{A}}^{\infty} \cdot n + 1$. Sie enthält also ausschließlich Kanten aus E und ist damit ein Hamiltonpfad in G.

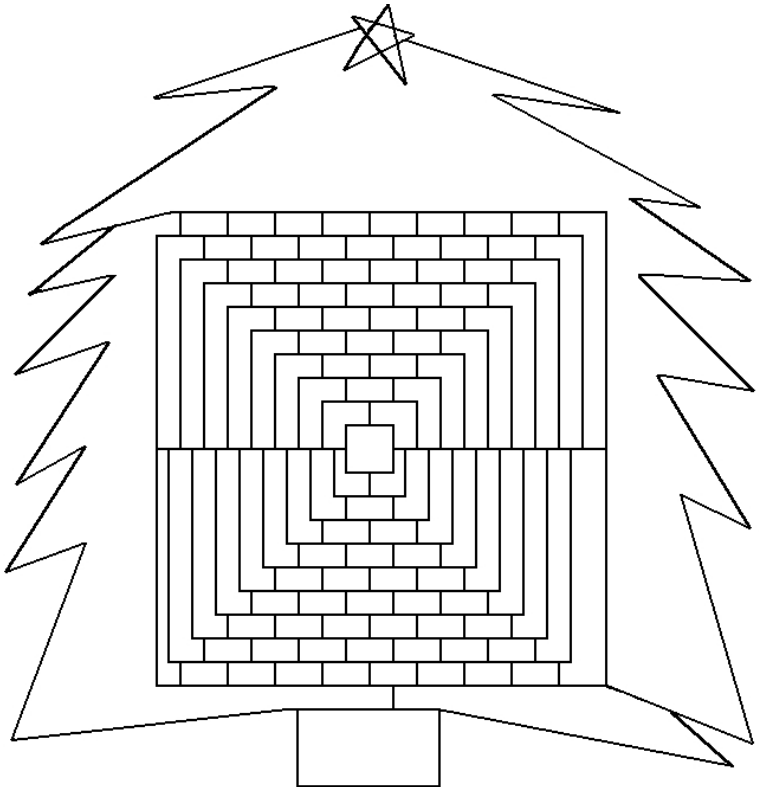
Da das Hamiltonpfad-Problem $\mathcal{NP}$-vollständig ist, ist damit gezeigt, dass aus der Existenz eines Approximationsalgorithmus mit konstanter relativer Gütegarantie die Aussage $\mathcal{P} = \mathcal{NP}$ folgt. Gilt umgekehrt $\mathcal{P} = \mathcal{NP}$, dann existiert ein polynomialer Algorithmus $\mathcal{A}$, der das Traveling Santa-Problem löst. Dieser Algorithmus ist insbesondere ein Approximationsalgorithmus mit relativer Gütegarantie 1.

Aufgabe 7 (2 Punkte)

Ein *planarer Graph* ist ein Graph, der so in der Ebene gezeichnet werden kann, dass sich keine zwei Kanten kreuzen. Die *Facetten* eines planaren Graphen bezüglich einer gegebenen Einbettung sind die 'maximalen, durch Kanten abgeschlossenen Flächen'. Insbesondere wird das 'den Graphen umgebende Gebiet' als *Äußere Facette* bezeichnet. Zwei Facetten sind *adjazent*, falls sie durch eine gemeinsame Kante begrenzt werden.

Die nachfolgende Zeichnung ist als planarer Graph zu betrachten, wobei die Knoten implizit als Schnitt- bzw. Berührungspunkte der Kanten gegeben seien.

Färben Sie die Facetten des Graphen mit vier Farben so, dass keine zwei adjazenten Facetten dieselbe Farbe haben.



FROHE WEIHNACHTEN, einen BRAUSENDEN JAHRESWECHSEL
und ein ERFOLGREICHES JAHR 2017