2. Klausur zur Vorlesung
Algorithmen II
Wintersemester 2012/2013

Lösung!

Beachten Sie:

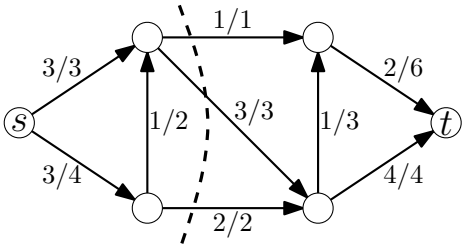
- Bringen Sie den Aufkleber mit Ihrem Namen und Matrikelnummer auf diesem Deckblatt an und beschriften Sie jedes Aufgabenblatt mit Ihrem Namen und Matrikelnummer.
- Schreiben Sie die Lösungen auf die Aufgabenblätter und Rückseiten. Zusätzliches Papier erhalten Sie bei Bedarf von der Aufsicht.
- Zum Bestehen der Klausur sind **20** der möglichen **60** Punkte hinreichend.
- Es sind keine Hilfsmittel zugelassen.

Aufgabe	Mögliche Punkte					Erreichte Punkte				
	a	b	c	d	Σ	a	b	c	d	Σ
1	1	1	3	2	7					
2	3,5	2,5	-	-	6			-	-	
3	1,5	3	1,5	-	6				-	
4	1,5	2	1,5	-	5				-	
5	1,5	5	3,5	-	10				-	
6	1,5	2	2,5	-	6				-	
7	2	1	2	-	5				-	
8	1,5	1	2,5	3	8					
9	7x1				7					
Σ					60					

Problem 1: Flüsse und lineare Programmierung $1 + 1 + 3 + 2 = 7$ Punkte

(a) Betrachten Sie das unten stehende Flussnetzwerk, in dem jede Kante mit ihrer Kapazität beschriftet ist. Geben Sie einen maximalen *st*-Fluss an, indem Sie die Flusswerte an die Kanten schreiben.

Lösung:



(b) Zeichnen Sie in das Netzwerk aus Aufgabenteil (a) einen minimalen *st*-Schnitt ein. Begründen Sie, warum Ihr Schnitt minimal ist.

Lösung: Der Schnitt hat Gewicht 6 und ist damit so groß wie der Fluss aus Aufgabenteil (a). Aus der Dualität zwischen maximalem Fluss und minimalem Schnitt folgt, dass es keinen kleineren Schnitt (und auch keinen größeren Fluss) geben kann.

(c) Sei (D, s, t, c) ein Flussnetzwerk bestehend aus einem gerichteten Graph $D = (V, E)$, der Quelle s, der Senke t, sowie der Kapazitätsfunktion $c\colon E \longrightarrow \mathbb{N}$. Formulieren Sie die Berechnung eines maximalen *st*-Flusses als lineares Programm (LP).

Hinweis: Benutzen Sie für jede Kante $(u, v) \in E$ eine Variable $x_{u,v}$, die angibt, wie viel Fluss auf der Kante (u, v) fließt.

Lösung:

Maximiere $$\sum_{(s,v)\in E} x_{s,v} - \sum_{(v,s)\in E} x_{v,s}$$

Mit Nebenbedingung $$\forall (u,v) \in E\colon \quad 0 \le x_{u,v} \le c(u,v) \qquad \text{(Kapazitätsbed.)}$$

und $$\forall u \in V \setminus \{s,t\}\colon \quad \sum_{(u,v)\in E} x_{u,v} - \sum_{(v,u)\in E} x_{v,u} = 0 \qquad \text{(Flusserhaltungsbed.)}$$

(d) Sei zusätzlich zu Aufgabenteil (c) eine Kostenfunktion $w\colon E \longrightarrow \mathbb{R}$ gegeben, mit der Bedeutung, dass ein Fluss $f(e)$ auf der Kante e die Kosten $f(e) \cdot w(e)$ verursacht. Geben Sie ein LP an, das einen st-Fluss mit festem Wert b und minimalen Kosten berechnet.

Lösung:

Minimiere $\displaystyle\sum_{(u,v)\in E} x_{u,v} \cdot w(u,v)$

Mit Nebenbedingung $\forall (u,v) \in E\colon \quad 0 \leq x_{u,v} \leq c(u,v)$ (Kapazitätsbed.)

und $\forall u \in V \setminus \{s,t\}\colon \quad \displaystyle\sum_{(u,v)\in E} x_{u,v} - \sum_{(v,u)\in E} x_{v,u} = 0$ (Flusserhaltungsbed.)

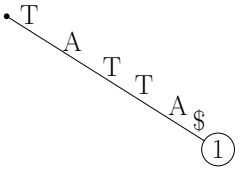
sowie $\displaystyle\sum_{(s,v)\in E} x_{s,v} - \sum_{(v,s)\in E} x_{v,s} = b$ (Flussgröße b)

Problem 2: Suffixbäume 3,5 + 2,5 = 6 Punkte

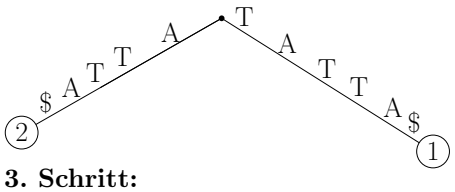
(a) Erstellen Sie den Suffixbaum der Zeichenkette $T =$ `TATTA`. Verwenden Sie das iterative Verfahren aus der Vorlesung, bei dem die Suffixe $S_1, S_2, \ldots$ nacheinander in den Baum einfügt werden und zeichnen Sie für **jeden Schritt einen eigenen Baum**. Stellen Sie sicher, dass jedes Suffix von einem Pfad zwischen einem Blatt und der Wurzel repräsentiert wird.

Lösung:

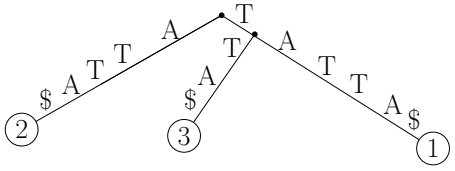
1. Schritt:



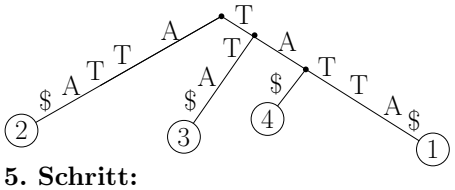
2. Schritt:



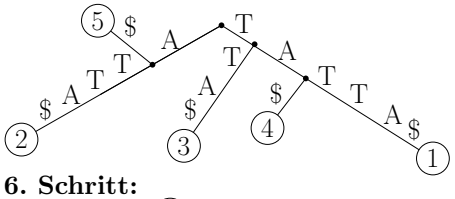
3. Schritt:



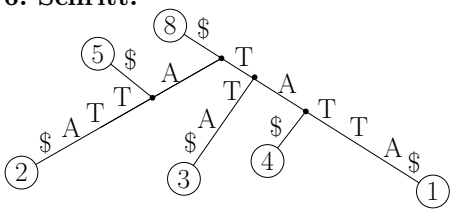
4. Schritt:



5. Schritt:



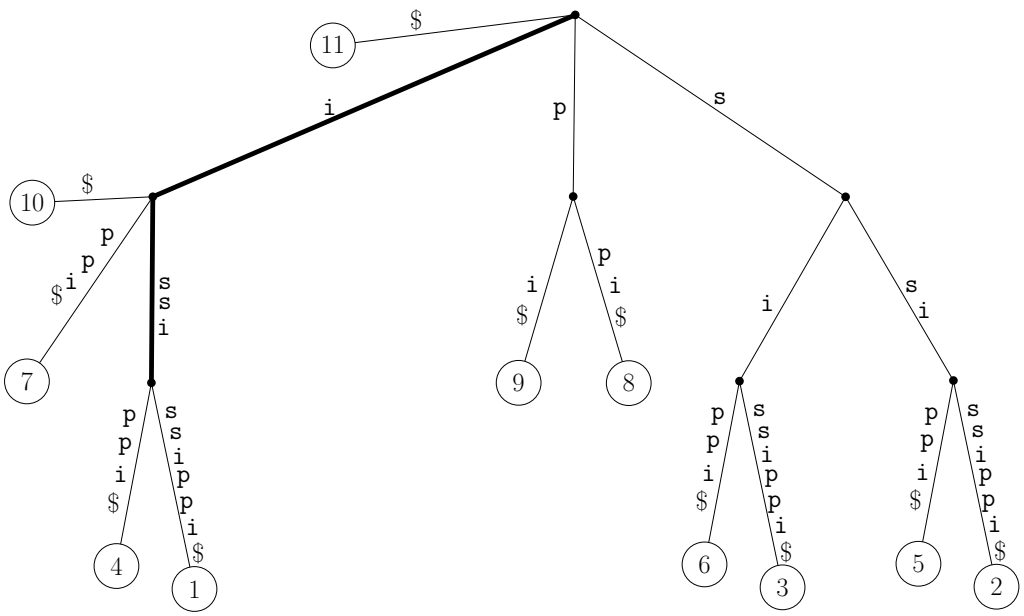
6. Schritt:



(b) Der *längste wiederholte Teilstring* in einer Zeichenkette T, ist der längste Teilstring von T, der mindestens zwei Mal in T vorkommt.

(i) Geben Sie den längsten wiederholten Teilstring von $T =$ `ississippi` an und markieren Sie in folgendem Suffixbaum den Pfad, der diesen Teilstring repräsentiert.

Lösung: Teilstring: `issi`

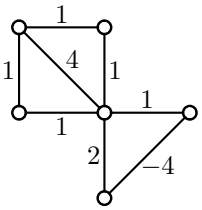


(ii) Skizzieren Sie einen Algorithmus, der die Länge des längsten wiederholten Teilstrings in einer Zeichenkette T in Zeit $O(|T|)$ bestimmt. Sie dürfen annehmen, dass der Suffixbaum $S = (V, E)$ von T gegeben ist und dass die String-Tiefe $d(v)$ eines Knotens $v \in V$ in $O(1)$ Zeit bestimmt werden kann.

Lösung: Wende zum Beispiel eine Breitensuche auf dem Baum an und merke global die größte String-Tiefe d_{max} der bereits besuchten inneren Knoten. Falls $d_{max} = 0$ nach der Breitensuche, dann gibt es keine Wiederholung von Teilstrings in T. Ansonsten ist d_{max} der gesuchte Wert.

Problem 3: Kreisraum & Kreisbasen 1,5 + 3 + 1,5 = 6 Punkte

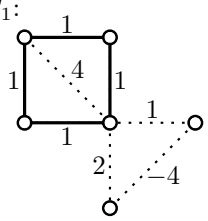
Gegeben sei der unten abgebildete Graph $G = (V, E)$. Das Kantengewicht $w(e)$ jeder Kante e ist an der Kante notiert.



(a) Zeichnen Sie in die folgenden Abbildungen alle (auch nicht einfache) Kreise ein und geben Sie das Gewicht der Kreise an.

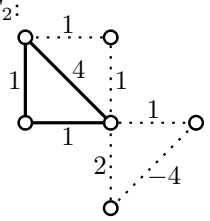
Lösung:

C_1:



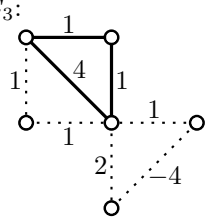
$w(C_1) = 4$

C_2:



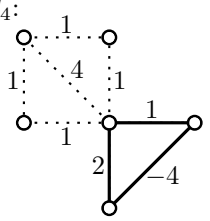
$w(C_2) = 6$

C_3:



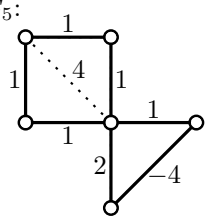
$w(C_3) = 6$

C_4:



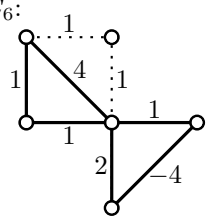
$w(C_4) = -1$

C_5:



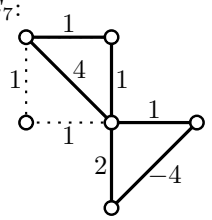
$w(C_5) = 3$

C_6:



$w(C_6) = 5$

C_7:



$w(C_7) = 5$

(b) Geben Sie eine minimale Kreisbasis B von G an. Verwenden Sie die Bezeichnungen der Kreise aus Aufgabenteil (a). Zeigen Sie, dass B eine minimale Kreisbasis ist (zu zeigen sind Minimalität **und** Kreisbasis-Eigenschaft).

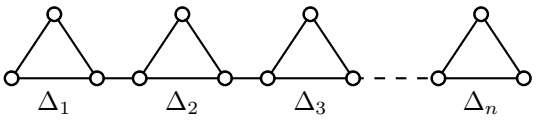
Lösung: $B = \{C_4, C_5, C_6\}$.

Basis-Eigenschaft: Offensichtlich ist B linear unabhängig. Außerdem gilt $n = 6$ und $m = 8$. $\Rightarrow$ Jeder Spannbaum enthält 5 Kanten. $\Rightarrow$ Eine Fundamentalbasis (und damit jede Basis) hat die Größe $8 - 5 = 3$. $\Rightarrow$ B muss Basis sein.

Minimalität: Da der Kreisraum ein Matroid ist, liefert die Greedy-Methode eine minimale Kreisbasis. Laut Aufgabenteil (a) sind C_4 und C_5 die beiden kleinsten Kreise. Der nächst größere Kreis ist C_1, allerdings gilt $C_1 = C_4 + C_5$. $\Rightarrow$ C_4 und C_5 bilden zusammen mit C_6 oder C_7 eine minimale Kreisbasis.

(c) Geben Sie eine Familie von Graphen G_n (für $n = 1, 2, 3, \dots$) an, sodass G_n $O(n)$ Knoten und $\Omega(2^n)$ Kreise enthält. Begründen Sie Ihre Antwort.

Lösung: Der Graph G_n besteht aus n Dreiecken $\Delta_1, \dots, \Delta_n$. Die Dreiecke können zusätzlich durch einen Pfad verbunden sein (falls der Graph zusammenhängend sein soll).



G_n hat $3n \in O(n)$ Knoten. Außerdem bildet jede Teilmenge der Dreiecke einen Kreis. $\Rightarrow$ Es gibt $\Omega(2^n)$ Kreise.

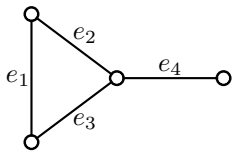
Problem 4: Matroide 1,5 + 2 + 1,5 = 5 Punkte

(a) Sei $G = (V, E)$ ein zusammenhängender Graph mit $|V| \geq 2$. Eine Teilmenge E_u induziert einen unzusammenhängenden Teilgraph, wenn $G_u = (V, E_u)$ unzusammenhängend ist. Zeigen Sie, dass $\mathcal{U}_u = \{E_u \mid E_u \subseteq E \text{ induziert unzusammenhängenden Teilgraph}\}$ ein Unabhängigkeitssystem ist.

Lösung: Beide Bedingungen für Unabhängigkeitssysteme sind erfüllt:

- Es gilt $\emptyset \in \mathcal{U}_u$, denn $(V, \emptyset)$ ist offensichtlich ein unzusammenhängender Graph.
- Falls der Graph $G_u = (V, E_u)$ unzusammenhängend ist, dann wird er durch Löschen von Kanten nicht zusammenhängend. $\Rightarrow E'_u \in \mathcal{U}_u$ für alle $E'_u \subseteq E_u$.

(b) Geben Sie für den folgenden Graphen alle Mengen in $\mathcal{U}_u$ explizit an.



Lösung:

$\mathcal{U}_u = \{\emptyset, \{e_1\}, \{e_2\}, \{e_3\}, \{e_4\}, \{e_1, e_2\}, \{e_1, e_3\}, \{e_1, e_4\}, \{e_2, e_3\}, \{e_2, e_4\}, \{e_3, e_4\}, \{e_1, e_2, e_3\}\}$

(c) Zeigen Sie, dass $\mathcal{U}_u$ im allgemeinen **kein** Matroid ist.

Lösung: Der Graph aus Aufgabenteil (a) ist ein Gegenbeispiel dafür, dass $\mathcal{U}_u$ im allgemeinen ein Matroid ist, denn: $\{e_1, e_4\} \in \mathcal{U}_u$ und $\{e_1, e_2, e_3\} \in \mathcal{U}_u$, aber $\{e_1, e_2, e_4\} \notin \mathcal{U}_u$ und $\{e_1, e_3, e_4\} \notin \mathcal{U}_u$.

Problem 5: Parametrisierte Algorithmen 1,5 + 5 + 3,5 = 10 Punkte

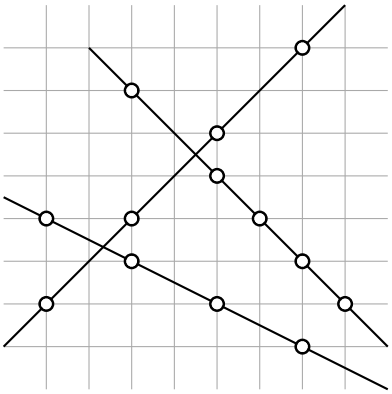
Das Problem PUNKTÜBERDECKUNG ist wie folgt definiert.

Gegeben: Eine Menge von Punkten $P = \{p_1, \ldots, p_n\}$ in der Ebene, sowie ein Parameter k.

Gesucht: Eine Menge von maximal k Geraden $G = \{g_1, \ldots, g_k\}$, sodass es für jeden Punkt $p_i \in P$ eine Gerade $g_j \in G$ mit $p_i \in g_j$ gibt.

(a) Gegeben sei eine Instanz von PUNKTÜBERDECKUNG bestehend aus der unten abgebildeten Punktmenge, sowie dem Parameter $k = 3$. Zeichnen Sie 3 Geraden ein, die diese Instanz von PUNKTÜBERDECKUNG lösen.

Lösung:



(b) Geben Sie einen FPT-Algorithmus für PUNKTÜBERDECKUNG an. Begründen Sie, warum ihr Algorithmus korrekt ist.

Hinweis: Benutzen Sie die Technik der Kernbildung. Zeigen Sie dazu zunächst, dass eine Gerade, die mehr als k Punkte aus P enthält, in jeder Lösung enthalten sein muss.

Lösung: Zeige zunächst den Hinweis. Sei g eine Gerade, die ℓ Punkte aus P enthält. Dann ist entweder g in der Lösung G enthalten, oder durch jeden der ℓ Punkte geht eine andere Gerade aus G. $\Rightarrow$ falls $\ell > k$, so muss g in der Lösung enthalten sein, da sonst $|G| > k$ gilt.

Damit erhält man folgenden Algorithmus, bestehend aus drei Schritten.

1. Solange es eine Gerade g gibt, die mehr als k Punkte aus P enthält, füge g zu G hinzu und lösche alle enthaltenen Punkte aus P.

   Sei P' die resultierende Punktmenge und sei $k' = k - |G|$. Es gibt offensichtlich für P mit Parameter k eine Lösung genau dann, wenn es für die Punktmenge P' mit dem Parameter k' eine Lösung gibt.

2. Falls $|P'| > k^2$, dann gebe *falsch* aus.

   Begründung: Jede Gerade kann nur k der übrigen Punkte enthalten. Daher kann man mit $k' \leq k$ Geraden nicht mehr als k^2 Punkte überdecken.

3. Betrachte die Menge aller Geraden $G^\star$, die mindestens zwei Punkte aus P' enthalten. Überprüfe für jede Teilmenge von $G^\star$ der Größe k', ob sie alle Punkte in P' überdeckt. Falls es eine solche Teilmenge gibt, gib sie zusammen mit den in Schritt 1. gefundenen Geraden aus.

   Man prüft also mittels Brute-Force-Methode, ob es Eine Lösung für P' mit Parameter k gibt.

(c) Zeigen Sie mit Hilfe einer Laufzeitanalyse, dass Ihr Algorithmus aus Aufgabenteil (b) ein FPT-Algorithmus ist.

Lösung: Es gibt $O(n^2)$ Geraden, die durch mindestens zwei Punkte in P gehen, für jede dieser Geraden muss man überprüfen, ob sie mehr als k Punkte enthalten. $\Rightarrow$ Laufzeit für Schritt 1.: $O(n^3)$.

Schritt 2. kann in konstanter Zeit ausgeführt werden.

In Schritt 3. gilt: $|P| \in O(k^2)$. $\Rightarrow$ Es gilt $|G^\star| \in O(k^4)$.

Die Anzahl der Teilmengen von $G^\star$ der Größe k ist also in $O((k^4)^k) = O(k^{4k})$. Pro Teilmenge benötigt man $O(|P'| \cdot k) = O(k^3)$ Zeit zum Überprüfen, ob jeder Knoten abgedeckt wird. Für Schritt 3. erhält man also eine Laufzeit von $O(k^{4k} \cdot k^3)$.

Insgesamt hat der Algorithmus die Laufzeit $O(k^{4k} \cdot k^3 + n^3)$ und ist damit ein FPT-Algorithmus.

Problem 6: Approximationsalgorithmus 1,5 + 2 + 2,5 = 6 Punkte

Ein gerichteter Graph heißt *azyklisch*, falls er keinen gerichteten Kreis enthält.

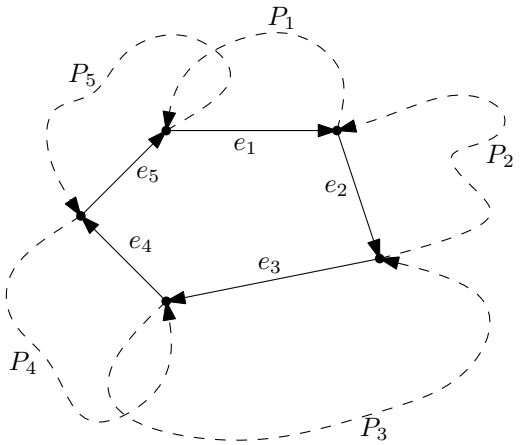
Sei $G = (V, E)$ ein gerichteter Graph und sei $G_1 = (V, E_1 \subseteq E)$ ein inklusionsmaximaler azyklischer Teilgraph von G. Des Weiteren sei $G_2 = (V, E_2 = E \setminus E_1)$ das Komplement zu G_1.

(a) Zeigen Sie: Für jede Kante $(u, v) \in E_2$ gibt es in G_1 einen gerichteten Pfad von v nach u.

Lösung: Annahme: Es gibt eine Kante $(u, v) \in E_2$, sodass es keinen gerichteten Pfad von v nach u in G_1 gibt. Dann kann die Kante (u, v) zu E_1 hinzu genommen werden, ohne das ein gerichteter Kreis entsteht. Dies ist ein Widerspruch dazu, dass G_1 inklusionsmaximal, azyklischer Graph ist.

(b) Zeigen Sie: G_2 ist azyklisch.

Lösung: Annahme: G_2 enthält einen Kreis $P = (e_1, e_2, \ldots, e_k)$. Nach Teilaufgabe (a) kann für jede dieser Kanten $e_i \in P$ ein gerichteter Pfad P_i in G_1 gefunden werden, der vom Zielknoten von e_i zum Startknoten von e_i führt:



Falls diese Pfade disjunkt sind, so bilden sie eine Kreis in G_1, was ein Widerspruch dazu ist, dass G_1 azyklisch ist. Andernfalls hat jeder Knoten in dem von den Pfaden induzierten Subgraph den gleichen Eingangs wie Ausgangsgrad. Damit enthält dieser Subgraph (und auch G_1) einen gerichteten Kreis.

(c) Betrachten Sie das Problem MAXIMUM ACYCLIC GRAPH:

Gegeben: Gerichteter Graph $G = (V, E)$.

Gesucht: Kardinalitätsmaximaler azyklischer Teilgraph von G.

Skizzieren Sie einen Approximationsalgorithmus für MAXIMUM ACYCLIC GRAPH mit relativer Gütegarantie 2. Beweisen Sie diese Gütegarantie.

Lösung:

Algorithmus 1 : MAGAPPROXIMATION

Eingabe : Gerichteter Graph $G = (V, E)$.
Ausgabe : Azyklischer Teilgraph von G.

```
1 Berechne inklusionsmaximalen azyklischen Teilgraph G_1 = (V, E_1) von G.
2 Berechne Komplementgraph G_2 = (V, E \ E_1) zu G_1.
3 Wenn |E_1| ≥ |E_2|
4 |   return G_1
5 sonst
6 |   return G_2
```

Nach Berechnungsvorschrift ist G_1 ein inklusionsmaximaler, azyklischer Teilgraph von G. Wegen Teilaufgabe b) folgt außerdem, dass der Komplementgraph G_2 von G_1 auch azyklisch ist. Da wegen $|E_1| + |E_2| = |E|$ und $E_1 \cap E_2 \neq \emptyset$ entweder $|E_1| \geq \frac{1}{2}|E|$ oder $|E_2| \geq \frac{1}{2}|E|$ gelten muss und die optimale Lösung maximal $|E|$ Kanten enthalten kann, folgt die Behauptung.

Problem 7: Randomisierte Algorithmen 2 + 1 + 2 = 5 Punkte

Ein gerichteter Graph $G = (V, E)$ ist *d-regulär*, wenn jeder Knoten genau d ausgehende und genau d eingehende Kanten hat.

Algorithmus 2 : RANDOMISIERTER ALGORITHMUS

Eingabe : d-regulärer gerichteter Graph $G = (V, E)$, Wahrscheinlichkeit $p \in [0, 1]$
Ausgabe : Teilmenge $S \subseteq E$

$S \leftarrow \emptyset$
Für *jedes* $(u, v) \in E$
  $x_{(u,v)} \leftarrow$ blau
  $x_{(u,v)} \leftarrow$ rot mit Wahrscheinlichkeit p
Für *jedes* $(u, v) \in E$
  Wenn $x_{(u,v)} =$ *rot* **oder** $x_{(u,w)} =$ *blau für alle von* u *ausgehenden Kanten* $(u, w) \in E$
    $S \leftarrow S \cup \{(u, v)\}$

(a) Sei S das Ergebnis von Algorithmus 2. Zeigen Sie: In dem von S induzierten Graph $G' = (V, S)$ hat jeder Knoten mindestens eine ausgehende Kante.

Lösung: Angenommen $u \in V$ hat keine ausgehende Kante in G'. Dann müssen alle ausgehenden Kanten $(u, v) \in E$ blau gefärbt sein, da sonst eine dieser Kanten in S enthalten wäre. Sind alle von u ausgehenden Kanten blau, so werden alle zu S hinzugefügt und u hat ausgehende Kanten in G'.

(b) Geben Sie die Wahrscheinlichkeit $P(e \in S)$ dafür an, dass eine gegebene Kante $e \in E$ in S liegt.

Lösung:

$$P(e \in S) = p + (1 - p)^d$$

(c) Geben Sie den Erwartungswert von $|S|$ in Abhängigkeit von p, d und $m = |E|$ an.

Hinweis: Betrachten Sie die Zufallsvariablen $Y_e = \begin{cases} 1, & \text{wenn } e \in S \\ 0, & \text{sonst.} \end{cases}$

Lösung:

$$\begin{aligned} \mathbb{E}(|S|) &= \mathbb{E}\left(\sum_{e \in E} Y_e\right) \\ &= \sum_{e \in E} \mathbb{E}(Y_e) \\ &= \sum_{e \in E} \left(p + (1 - p)^d\right) \\ &= m \cdot p + m \cdot (1 - p)^d \end{aligned}$$

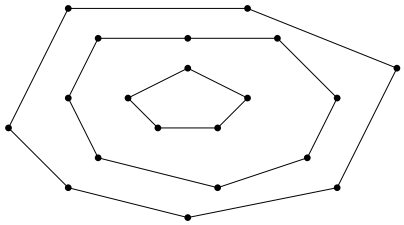
Problem 8: Konvexe Hülle 1,5 + 1 + 2,5 + 3 = 8 Punkte

Gegeben sei eine endliche, nicht-leere Punktmenge $P \subset \mathbb{R}^2$. Sei $H(S)$ die konvexe Hülle einer Teilmenge $S \subseteq P$. Die *Schichten* von P sind rekursiv definiert durch:

Basisfall: $S_1 = H(P)$

Allgemeiner Fall: $S_i = H(P \setminus (S_1 \cup \ldots \cup S_{i-1}))$ für $i > 1$

Betrachten Sie nun das Problem ONION PEELING: Gesucht sind die Schichten $S_1, \ldots, S_k$ von P, sodass $S_k \neq \emptyset$ und $S_{k+1} = \emptyset$ gilt. Folgende Abbildung zeigt eine Punktmenge mit $k = 3$ nicht-leeren Schichten.



(a) Zeigen Sie, dass für zwei nicht-leere Schichten S_{i-1} und S_i mit $i > 1$ gilt: Die Punkte aus S_i liegen innerhalb des konvexen Polygons S_{i-1}.

Lösung: Sei $P_1 = P$ und $P_i = P \setminus (S_1 \cup \ldots \cup S_{i-1})$ für $i > 1$. Da nach Definition P_i im Vergleich zu P_{i-1} nur Punkte verliert, aber keine neuen hinzukommen, gilt $P_i \subseteq P_{i-1}$. Des Weiteren, da S_{i-1} das kleinste konvexe Polygon bildet, das alle Punkte P_{i-1} einschließt, folgt dass P_i vollständig in S_{i-1} enthalten ist. Wegen $S_i \subseteq P_i$ gilt somit die Behauptung.

(b) Zeigen Sie, dass der Algorithmus SIMPLEONIONPEELING für $n = |P|$ die Laufzeit $O(n^2 \log n)$ hat.

Algorithmus 3 : SIMPLEONIONPEELING

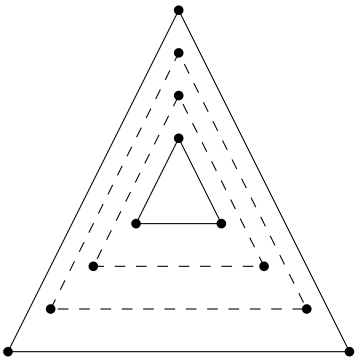
Eingabe : Endliche Menge $P \subset \mathbb{R}^2$ bestehend aus n Punkten.
Ausgabe : Die Schichten $S_1, \ldots, S_k$ von P, sodass gilt $S_k \neq \emptyset$ und $S_{k+1} = \emptyset$.

```
1 i ← 1
2 P_1 ← P
3 solange P_i ≠ ∅ tue
4 |   S_i ← Ergebnis von Graham Scan angewendet auf P_i
5 |   P_{i+1} ← P_i \ S_i //Kann in O(n) Zeit berechnet werden.
6 |   i ← i + 1
7 return S_1, ..., S_{i-1}
```

Lösung: Da in jedem Durchlauf Punkte aus der aktuellen Punktemenge entfernt werden, kann es maximal n Durchläufe der äußeren Schleife geben. In der Schleife selber stellt Graham Scan mit $O(n \log n)$ Zeit den aufwendigsten Teil dar. Damit ergibt sich die Laufzeit $O(n^2 \log n)$.

(c) Geben Sie eine Familie von Punktmengen $P_1, P_2, \ldots$ mit $|P_n| \in O(n)$ an, sodass die Laufzeit von SIMPLEONIONPEELING für diese Instanzen in $\Omega(n^2 \log n)$ liegt. Begründen Sie Ihre Antwort.

Lösung: Betrachte folgende Punktmenge P, die $\frac{n}{3}$ Schichten induziert, die jeweils Dreiecke bilden. Offensichtlich gilt nach dem i-ten Schleifendurchlauf $|P_{i+1}| = |P_i| - 3$, d.h. die betrachtete



Punktmenge wird pro Durchlauf nur um konstant viele Elemente kleiner, was somit in $\Theta(n)$ Schleifendurchläufen resultiert. Da Graham Scan $\Theta(|P_i| \log |P_i|)$ Zeit benötigt, folgt damit die Laufzeit $\Theta(n^2 \log n)$.

(d) Skizzieren Sie einen Algorithmus, der das Problem ONION PEELING in $O(n^2)$ Laufzeit löst und beweisen Sie diese Laufzeit.

Lösung: Ersetze in SIMPLEONIONPEELING den Aufruf von Graham Scan durch Gift Wrapping. Für eine Punktmenge der Größe n mit konvexer Hülle der Größe h benötigt Gift Wrapping $O(n \cdot h)$ Zeit. Da die Punktmenge bei keinem Aufruf mehr als n Elemente enthält, wird die i-te Schicht in $O(n \cdot |S_i|)$ Zeit berechnet. Summiert man über alle Schichten, so erhält man $O(\sum(n \cdot |S_i|)) = O(n \cdot \sum |S_i|) = O(n^2)$, da jedes Element nur in einer Schicht enthalten ist.

Problem 9: Verschiedenes $7 \times 1 = 7$ Punkte

Jeder der folgenden Aussagenblöcke umfasst mehrere Einzelaussagen. Die sich unterscheidenden Aussagenbausteine sind durch Kästchen gekennzeichnet. Kreuzen Sie genau jene Bausteine an, die in einer wahren Einzelaussage enthalten sind. Jeder Aussagenblock enthält mindestens eine wahre Einzelaussage. Unvollständig oder falsch angekreuzte Aussagenblöcke werden mit null Punkten bewertet. Sie erhalten einen Punkt für jeden Aussagenblock, für den Sie genau die richtige Menge an Aussagenbausteinen angekreuzt haben.

Ein Algorithmus mit asymptotischer Laufzeit

- ☐ $\Theta(k \log k \cdot n^2 + 5^k \cdot n^k)$
- ☒ $\Theta(n)$
- ☐ $\Theta(2^k \cdot n! + \sqrt{n})$
- ☒ $\Theta(2^{k!} \cdot n + n^2 \log n)$

ist ein FPT-Algorithmus, wobei n die Eingabegröße und k der Parameter ist.

Bei einer Menge von n Strecken in der Ebene

- ☐ können alle Schnittpunkte zwischen diesen Strecken in $O(n \log n)$ berechnet werden.
- ☒ kann jede Strecke $n - 1$ andere Strecken schneiden.
- ☐ schneidet jede Strecke mindestens eine andere.
- ☒ kann es $\Omega(n^2)$ viele Schnittpunkte zwischen diesen Strecken geben.

Ein minimaler st-Schnitt in einem zusammenhängenden, ungerichteten und ungewichteten Graphen $G = (V, E)$

- ☒ kann in $O(|E|)$ Zeit berechnet werden, wenn ein maximaler st-Fluss gegeben ist.
- ☒ kann Gewicht $|V| - 1$ haben.
- ☐ trennt t von mindestens einem Knoten $v \in V$ mit $v \neq s$.
- ☒ hat Gewicht mindestens 1.

Eine Menge von Algorithmen $\{A_\varepsilon \mid \varepsilon \in \mathbb{R}^+\}$ mit Laufzeit $\Theta(3^{\frac{1}{\varepsilon}} \cdot n^3)$ und relativer Approximationsgüte $1 + \varepsilon$ ist ein

- ☒ PAS
- ☐ FPAS
- ☐ AFPAS
- ☒ APAS

Sei G ein Graph und C ein minimales VERTEX COVER der Größe k in G. Dann enthält C

- ☐ alle Knoten mit Grad maximal k.
- ☒ keine isolierten Knoten. (Ein Knoten ist isoliert, wenn er Grad 0 hat.)
- ☐ für jede Kante maximal einen der beiden Endknoten.
- ☒ alle Knoten mit Grad größer als k.

Sei $G = (V, E)$ ein Graph. Die Menge $\mathcal{U}$ enthalte die Menge $V' \subseteq V$ genau dann, wenn für alle $\{u, v\} \in E$ gilt: $u \notin V'$ oder $v \notin V'$. Dann

- ☐ ist $\mathcal{U}$ ein Matroid.
- ☒ ist $\mathcal{U}$ ein Unabhängigkeitssystem.
- ☐ gilt $V \in \mathcal{U}$.
- ☒ ist es $\mathcal{NP}$-Schwer eine kardinalitätsmaximale Menge in $\mathcal{U}$ zu finden.

Beim (h, k)-Paging

- ☐ darf der Online-Algorithmus maximal k Fehlzugriffe haben.
- ☐ hat der optimale Offline-Algorithmus immer weniger Fehlzugriffe als der Online-Algorithmus.
- ☒ hat der Online-Algorithmus einen mindestens genauso großen Cache zur Verfügung wie der optimale Offline-Algorithmus.
- ☒ sind konservative Online-Algorithmen 2-kompetitiv, wenn $h = \frac{k}{2}$.