1. Klausur zur Vorlesung
Algorithmen II
Wintersemester 2012/2013

Lösung!

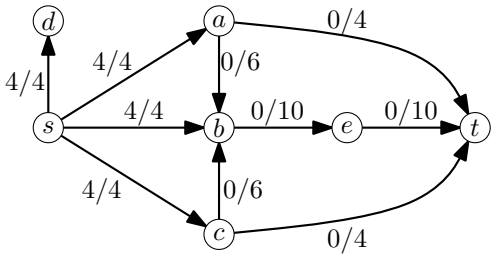
Beachten Sie:

- 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	Σ	a	b	c	Σ
1	1	5	–	6			–	
2	2,5	3	1,5	7				
3	1	1	3	5				
4	4	2	1	7				
5	1,5	3,5	3	8				
6	3	2	4	9				
7	2	1	2	5				
8	2	2	2	6				
9	7x1			7				
Σ				60				

Problem 1: Algorithmus von Goldberg & Tarjan 1 + 5 = 6 Punkte

Betrachten Sie folgendes Flussnetzwerk mit Quelle s und Senke t, das den Zustand des Goldberg-Tarjan-Algorithmus direkt nach der Initialisierung zeigt. Jede Kante e ist mit dem aktuellen Präfluss $f(e)$ sowie der Kapazität $c(e)$ im Format $f(e)/c(e)$ beschriftet.



(a) Geben Sie die zulässige Markierung $\text{dist}(v)$ sowie den Flussüberschuss $e(v)$ für jeden Knoten v nach der Initialisierung an und zählen Sie alle aktiven Knoten auf.

Lösung:

Aktive Knoten: a, b, c und d.

Überschuss: $e(a) = e(b) = e(c) = e(d) = 4$, für restliche Knoten $v \in V \setminus \{t\}$ gilt $e(v) = 0$.

Zulässige Markierung: $\text{dist}(s) = 7$, für restliche Knoten $v \in V$ gilt $\text{dist}(v) = 0$.

(b) Geben Sie eine Sequenz bestehend aus RELABEL- und PUSH-Operationen an, sodass folgendes gilt:

- Die Sequenz darf beliebig viele RELABEL-Operationen enthalten.
- Die Sequenz enthält genau zwei nicht-saturierende PUSH-Operationen.
- Die Sequenz enthält genau drei saturierende PUSH-Operationen.

Geben Sie außerdem für jede Operation an, wie sich die zulässige Markierung $\text{dist}(\cdot)$, sowie die Flussüberschüsse $e(\cdot)$ ändern.

Hinweis: Führen Sie die zwei nicht-saturierenden vor den drei saturierenden PUSH-Operationen aus.

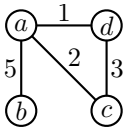
Lösung:

1. RELABEL(a): $\text{dist}(a) = 1$
2. PUSH$(a, b) = 4$: $e(a) = 0$, $e(b) = 8$, nicht saturierend
3. RELABEL(c): $\text{dist}(c) = 1$
4. PUSH$(c, b) = 4$: $e(c) = 0$, $e(b) = 12$, nicht saturierend
5. RELABEL(b): $\text{dist}(b) = 1$
6. PUSH$(b, e) = 10$: $e(b) = 2$, $e(e) = 10$, saturierend

7. RELABEL(e): $\text{dist}(e) = 1$
8. PUSH$(e, t) = 10$: $e(e) = 0$, $e(t) = 10$, saturierend
9. RELABEL(d): $\text{dist}(d) = 8$
10. PUSH$(d, s) = 4$: $e(d) = 0$, saturierend

Problem 2: Global minimaler Schnitt 2,5 + 3 + 1,5 = 7 Punkte

(a) Wenden Sie den Stoer-Wagner Algorithmus auf den nebenstehenden Graphen an. Die Kantengewichte sind an den Kanten notiert. Geben Sie für jede Phase i die Knoten s_i und t_i, den Schnitt der Phase und dessen Gewicht an. Zeichnen Sie den nach dem Verschmelzen resultierenden Graphen mit Kantengewichten. Der Startknoten in jeder Phase enthalte a. Geben Sie zum Schluss den minimalen Schnitt $S_{\min}$ an.

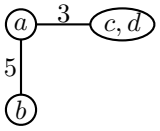


Lösung:

Phase 1:

$s_1 = c$ $t_1 = d$

Graph nach Verschmelzen:



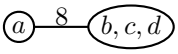
Schnitt der Phase $S_1 = \{d\}$

$c(S_1, V \setminus S_1) = 4$

Phase 2:

$s_2 = b$ $t_2 = c, d$

Graph nach Verschmelzen:



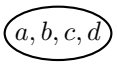
Schnitt der Phase $S_2 = \{c, d\}$

$c(S_2, V \setminus S_2) = 3$

Phase 3:

$s_3 = a$ $t_3 = b, c, d$

Graph nach Verschmelzen:



Schnitt der Phase $S_3 = \{b, c, d\}$

$c(S_3, V \setminus S_3) = 8$

Minimaler Schnitt $S_{\min} = (\{a, b\}, \{c, d\})$

(b) Beweisen Sie, dass der Schnitt der Phase mit minimalem Gewicht ein global minimaler Schnitt ist. Sie dürfen verwenden, dass der Schnitt jeder Phase ein minimaler st-Schnitt ist.

Lösung: Induktion über $n = |V|$:

I.A. ($n = 2$): Offensichtlich ist der einzige mögliche Schnitt ein global minimaler Schnitt.

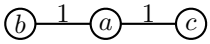
I.V.: In einem Graphen mit $n - 1$ Knoten ist der Schnitt der Phase mit minimalem Gewicht ein global minimaler Schnitt.

I.S.: Der Schnitt S_1 der Phase 1 ist ein minimaler s_1t_1-Schnitt. Falls ein global minimaler Schnitt s_1 und t_1 trennt, so ist S_1 global minimal und wird damit gefunden.

Andernfalls bleibt der global minimale Schnitt beim Verschmelzen von s_1 und t_1 erhalten. Der resultierende Graph hat $n - 1$ Knoten und damit ist nach I.V. der Schnitt der restlichen $n - 2$ Phasen mit minimalem Gewicht ein global minimaler Schnitt.

(c) Geben Sie einen Graphen an, bei dem der Stoer-Wagner Algorithmus je nach Ausführung unterschiedliche Schnitte zurückgeben kann. Begründen Sie kurz warum. An welchen Stellen im Algorithmus können Entscheidungen getroffen werden?

Lösung:



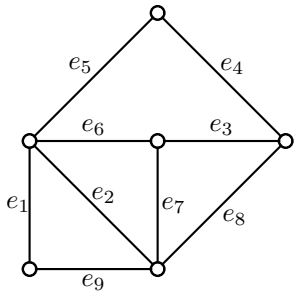
Wählt man a als Startknoten so erhält man entweder $s = b$ und $t = c$ oder umgekehrt, da b und c gleich stark mit a verbunden sind. Damit bekommt man in der ersten Phase entweder den Schnitt $(\{c\}, V \setminus \{c\})$ oder $(\{b\}, V \setminus \{b\})$. Da der Schnitt der zweiten Phasen in beiden Fällen teurer ist, wird einer dieser beiden Schnitte ausgegeben.

Stellen an denen Entscheidungen getroffen werden können:

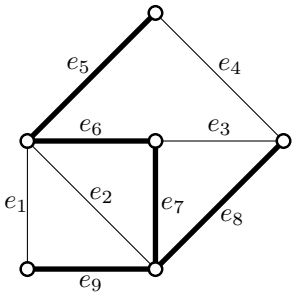
1. Zu Beginn einer Phase: Auswahl eines Startknotens.
2. In einer Phase: Sind mehrere Knoten gleich stark mit dem bisher abgeschnittenen Teil verbunden, so kann man wählen, welchen man hinzufügt.
3. Nach allen Phasen: Falls zwei Phasen einen Schnitt gleicher Größe liefern, so kann man einen auswählen und zurückgeben.

Problem 3: Kreisraum & Kreisbasen $1 + 1 + 3 = 5$ Punkte

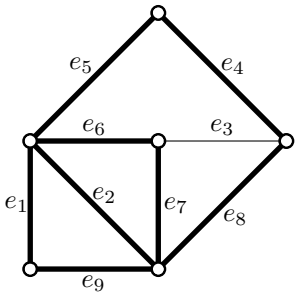
Gegeben sei der unten abgebildete Graph $G = (V, E)$, zusammen mit einem Spannbaum T und einem Kreis C (jeweils fett eingezeichnet).



Graph $G = (V, E)$



Spannbaum T



Kreis C

(a) Geben Sie die Fundamentalbasis B_T des Kreisraums von G bezüglich des Spannbaums T an.

Lösung:

$$C_1 = \{e_1, e_6, e_7, e_9\}$$
$$C_2 = \{e_2, e_6, e_7\}$$
$$C_3 = \{e_3, e_7, e_8\}$$
$$C_4 = \{e_4, e_5, e_6, e_7, e_8\}$$
$$B_T = \{C_1, C_2, C_3, C_4\}$$

(b) Stellen Sie den Kreis C als Summe von Kreisen aus B_T dar.

Lösung:

$$C = C_1 \oplus C_2 \oplus C_4$$

(c) Geben Sie eine Kreisbasis B des Kreisraums von G an, die **keine** Fundamentalbasis ist. Begründen Sie warum B eine Basis, aber keine Fundamentalbasis ist.

Lösung:

$$B = \{C_1, C_2, C_3, C\}$$

Aus Aufgabenteil (b) folgt, dass C_4 die Summe von C_1, C_2 und C ist. Damit kann jeder Vektor aus der Basis B_T als Summe von Vektoren aus B dargestellt werden, das heißt B ist ein erzeugendes System. Außerdem ist es nur so groß wie die Basis B_T und muss daher selbst eine Basis sein.

Außerdem kann B keine Fundamentalbasis sein, da sie einen Kreis enthält der nicht einfach ist.

Problem 4: Matroide $4 + 2 + 1 = 7$ Punkte

(a) Sei $G = (V, E)$ ein zusammenhängender Graph. Eine Teilmenge $E_w \subseteq E$ induziert einen Wald, wenn $G_w = (V, E_w)$ keinen Kreis enthält. Zeigen Sie, dass das Unabhängigkeitssystem $\mathcal{U}_w = \{E_w \mid E_w \subseteq E \text{ induziert einen Wald}\}$ ein Matroid ist.

Hinweis: Sie dürfen verwenden, dass ein Wald $G_w = (V, E_w)$ genau $|V| - k$ Kanten enthält, wobei k die Anzahl der Zusammenhangskomponenten in G_w ist.

Lösung: Seien E_w und E'_w zwei Wälder mit $|E_w| > |E'_w|$. Aus $|E_w| = |V| - k$ folgt, dass E_w weniger Zusammenhangskomponenten hat als E'_w.

$\Rightarrow$ Es gibt zwei Knoten u und v, die in (V, E'_w) in unterschiedlichen Zusammenhangskomponenten liegen, in (V, E_w) aber in der selben.

$\Rightarrow$ Auf dem Pfad von u nach v in (V, E_w) gibt es eine Kante $\{u', v'\}$, sodass u' und v' in unterschiedlichen Zusammenhangskomponenten in (V, E'_w) liegen.

$\Rightarrow$ Der von $E'_w \cup \{\{u', v'\}\}$ induzierte Graph enthält keinen Kreis.

$\Rightarrow E'_w \cup \{\{u', v'\}\} \in \mathcal{U}_w$.

(b) Zeigen Sie: Jeder Spannbaum in G ist eine Basis von $\mathcal{U}_w$ und umgekehrt.

Lösung: Die Basen eines Matroids sind genau die maximalen unabhängigen Teilmengen. Ein Wald mit $|V|-k$ Kanten ist genau dann maximal, wenn er aus $k = 1$ Zusammenhangskomponente besteht, also genau dann, wenn er ein Spannbaum ist.

(c) Sei $w\colon E \longrightarrow \mathbb{R}$ eine Gewichtsfunktion auf den Kanten von G. Berechnet der folgende Algorithmus einen minimalen Spannbaum in G? Begründen Sie Ihre Antwort.

Lösung: Ja, da die Greedy-Methode bei Matroiden eine optimale Basis liefert, also in diesem Fall einen minimalen Spannbaum in G.

Algorithmus 1 : MAYBE-MST

Eingabe : Ein zusammenhängender Graph G
Ausgabe : Ein Teilgraph $G' = (V, E')$

```
1 Sortiere E aufsteigend nach Gewichtsfunktion w, sei e_1, ..., e_m die Sortierung
2 E' ← ∅
3 Für i = 1 bis m
4 |   Wenn E' ∪ {e_i} ∈ U_w
5 |   |   E' ← E' ∪ {e_i}
6 gib G' = (V, E') zurück
```

Problem 5: Parametrisierte Algorithmen 1,5 + 3,5 + 3 = 8 Punkte

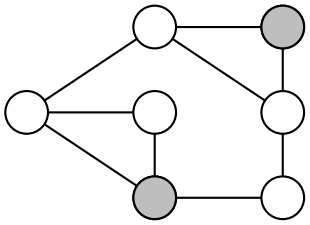
Das Problem DOMINIERENDE MENGE für Graphen mit Maximalgrad d ist wie folgt definiert.

Gegeben: Ein Graph $G = (V, E)$, sodass jeder Knoten maximal d Nachbarn hat, sowie ein Parameter $k = d + \ell$.

Gesucht: Eine Knotenmenge $V' \subseteq V$ mit $|V'| \leq \ell$, sodass für jeden Knoten $v \in V$ der Knoten v und/oder einer seiner Nachbarn in V' enthalten ist.

(a) Gegeben sei eine Instanz von DOMINIERENDE MENGE bestehend aus dem unten abgebildeten Graphen mit Maximalgrad $d = 3$, sowie dem Parameter $k = 5$. Markieren Sie $\ell = 2$ Knoten, die diese Instanz von DOMINIERENDE MENGE lösen.

Lösung:



(b) Geben Sie einen FPT-Algorithmus für DOMINIERENDE MENGE an. Begründen Sie, warum Ihr Algorithmus korrekt ist.

Hinweis: Benutzen Sie einen Entscheidungsbaum.

Lösung: Sei $V' \subseteq V$ die Menge der bisher ausgewählten Knoten (initial: $V' = \emptyset$). Ein Knoten heißt im folgenden *dominiert*, falls er selbst oder einer seiner Nachbarn in V' enthalten ist. Sind alle Knoten dominiert, so gib V' aus. Andernfalls sei $v \in V$ ein Knoten, der noch nicht dominiert wird. Dann muss entweder v oder einer seiner maximal d Nachbarn zu einer Lösung von DOMINIERENDE MENGE gehören. Man erhält also $d + 1$ Kandidaten für den nächsten Knoten. Verfahre für jeden der Kandidaten u wie folgt:

1. Wenn $|V'| < \ell$, dann fahre rekursiv fort, wobei $V' \cup \{u\}$ als Menge der bisher ausgewählte Knoten übergeben wird.
2. Wenn $|V'| = \ell$, so endet die Rekursion, da V' noch keine dominierende Menge ist, man aber keinen Knoten mehr zu V' hinzufügen kann.

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

Lösung: Die rekursiven Aufrufe bilden einen Entscheidungsbaum der Höhe ℓ, wobei jeder Knoten maximal $d+1$ Kinder hat. Damit enthält der Baum $O((d+1)^{\ell+1}) \in O(k^k)$ Knoten. Entsprechend werden $O(k^k)$ Schritte ausgeführt, wobei in jedem Schritt ein noch nicht dominierter Knoten gefunden werden muss, was in $O(n)$ Laufzeit geht. Damit ergibt sich eine Gesamtlaufzeit von $O(k^k \cdot n)$.

Problem 6: Approximationsalgorithmen 3 + 2 + 4 = 9 Punkte

Das Problem INDEPENDENT SQUARES sei wie folgt definiert:

Gegeben: Menge $Q = \{q_1, \ldots, q_n\}$ gleichgroßer, achsenparalleler Quadrate in der Ebene.

Gesucht: Möglichst große unabhängige Menge $S \subseteq Q$. Dabei heißt $S \subseteq Q$ *unabhängig*, falls für alle $q_i, q_j \in S$ mit $i \neq j$ gilt, dass q_i und q_j sich nicht schneiden.

Betrachten Sie den Algorithmus SWEEPLINE, der eine inklusionsmaximale unabhängige Teilmenge $S \subseteq Q$ berechnet.

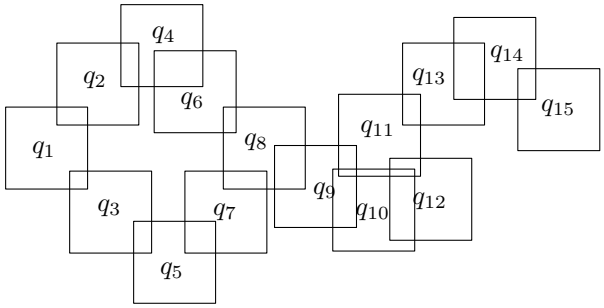
Algorithmus 2 : SWEEPLINE

Eingabe : Menge $Q = \{q_1, \ldots, q_n\}$ gleichgroßer, achsenparalleler Quadrate in der Ebene mit Mittelpunkten $c_1, \ldots, c_n$, sodass für die x-Koordinaten der Mittelpunkte gilt $x(c_1) < \ldots < x(c_n)$.

Ausgabe : Unabhängige Menge $S \subseteq Q$.

```
1 S ← ∅
2 Für i = 1, ..., n
3     Wenn q_i ∈ Q
4         S ← S ∪ {q_i}
5         Q ← Q \ ({q_i} ∪ {q_j ∈ Q | q_j und q_i schneiden sich.})
6 return S
```

(a) Geben Sie die unabhängige Menge S an, die SWEEPLINE auf folgender Instanz berechnet. Geben Sie zudem eine kardinalitätsmaximale unabhängige Menge S' derselben Instanz an.



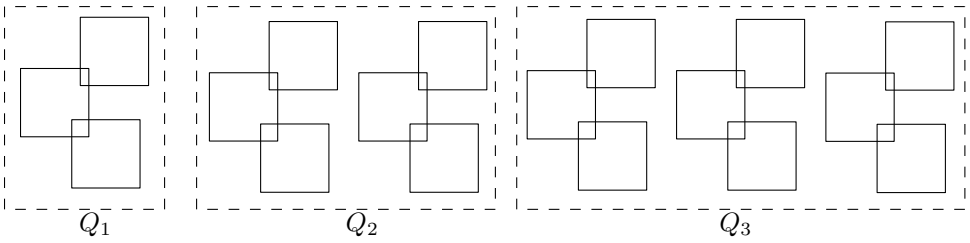
Lösung:

SWEEPLINE: q_1, q_4, q_5, q_8, q_{10}, q_{13}, q_{15}

Optimal: q_2, q_3, q_6, q_7, q_9, q_{12}, q_{13}, q_{15}

(b) Geben Sie eine Familie $Q_1, Q_2, Q_3, \ldots$ gleichgroßer, achsenparalleler Quadrate an, sodass gilt $|Q_n| \in \Theta(n)$ und $|\textsc{SweepLine}(Q_n)| = \frac{1}{2}|\operatorname{OPT}(Q_n)|$ für alle $n \in \mathbb{N}$. Dabei bezeichnet $\operatorname{OPT}(Q)$ die kardinalitätsmaximale unabhängige Menge von Q. Begründen Sie Ihre Antwort.

Lösung:



Definiere Q_1 wie in der Abbildung. Q_i ist dann die Aneinanderreihung von i Kopien von Q_1, sodass ausreichend Platz zwischen den Kopien ist.

Für Q_1 liefert SweepLine das Quadrat, das am weitesten links liegt, während die optimale Lösung die zwei rechten Quadrate wählt. Somit unterscheiden sich beide Lösungen um den Faktor 2. Da Q_i die i-fache Aneinanderreihung von Q_1 ist, sodass keine weiteren Überlappungen entstehen und SweepLine die Quadrate von links nach rechts abarbeitet, ergibt sich für Q_i der selbe Faktor.

(c) Zeigen Sie, dass für jede Instanz Q die Ungleichung $|\textsc{SweepLine}(Q)| \geq \frac{1}{2}|\operatorname{OPT}(Q)|$ gilt. Dabei bezeichnet $\operatorname{OPT}(Q)$ die kardinalitätsmaximale unabhängige Menge von Q.

Lösung: Betrachte den Fall, dass q_i im i-ten Schritt in S eingefügt wird. Sei $K = \{q_j \in Q_i \mid q_j \text{ und } q_i \text{ schneiden sich}\}$. Da alle Quadrate gleichgroß und achsenparallel sind, überlappt jedes Quadrat $q_j \in Q$ mindestens eine Ecke von q_i. Wegen $x(c_j) > x(c_i)$ für alle $q_j \in Q$ kommen nur die obere-rechte und die untere-rechte Ecke in Frage. Folglich ist $|\operatorname{OPT}(K)| \leq 2$ und somit können im schlimmsten Fall zwei Quadrate der optimalen Lösung verloren gehen, während eins zur Lösung hinzugenommen wird.

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

Ein Graph $G = (V, E)$ ist *d-regulär*, wenn jeder Knoten genau d Nachbarn hat.

Algorithmus 3 : RANDOMISIERTER ALGORITHMUS

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

```
1 S ← ∅
2 Für jedes v ∈ V
3 |   x_v ← blau
4 |   x_v ← rot mit Wahrscheinlichkeit p
5 Für jedes v ∈ V
6 |   Wenn x_v = rot oder x_u = blau für alle Nachbarn u von v
7 |   |   S ← S ∪ {v}
```

(a) Zeigen Sie: Das Ergebnis S von Algorithmus 3 ist eine dominierende Menge, das heißt, für alle $v \in V$ gilt $v \in S$ oder $u \in S$ für einen Nachbar u von v.

Lösung: Ein Knoten v wird nur dann nicht zu S hinzugefügt, wenn er selber blau und einer der Nachbarn rot ist. Damit ist sichergestellt, dass einer der Nachbarn von v in S liegt, falls v nicht selber in S liegt.

(b) Geben Sie die Wahrscheinlichkeit $P(v \in S)$ dafür an, dass ein gegebener Knoten v in S liegt.

Lösung:

$$P(v \in S) = p + (1 - p)^{d+1}$$

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

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

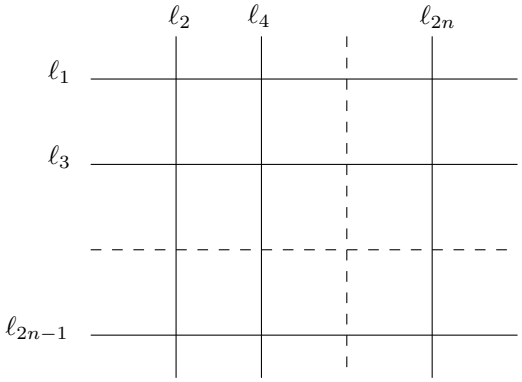
Lösung:

$$\begin{aligned} \mathbb{E}(|S|) &= \mathbb{E}\left(\sum_{v \in V} Y_v\right) \\ &= \sum_{v \in V} \mathbb{E}(Y_v) \\ &= \sum_{v \in V} \left(p + (1 - p)^{d+1}\right) \\ &= n \cdot p + n \cdot (1 - p)^{d+1} \end{aligned}$$

Problem 8: Schnittpunkte von Strecken $2 + 2 + 2 = 6$ Punkte

(a) Geben Sie eine Familie von Streckenmengen $S_1, S_2, \ldots$ mit $|S_n| \in O(n)$ an, sodass S_n mindestens $\Omega(n^2)$ sich schneidende Streckenpaare enthält. Begründen Sie Ihre Antwort.

Lösung:

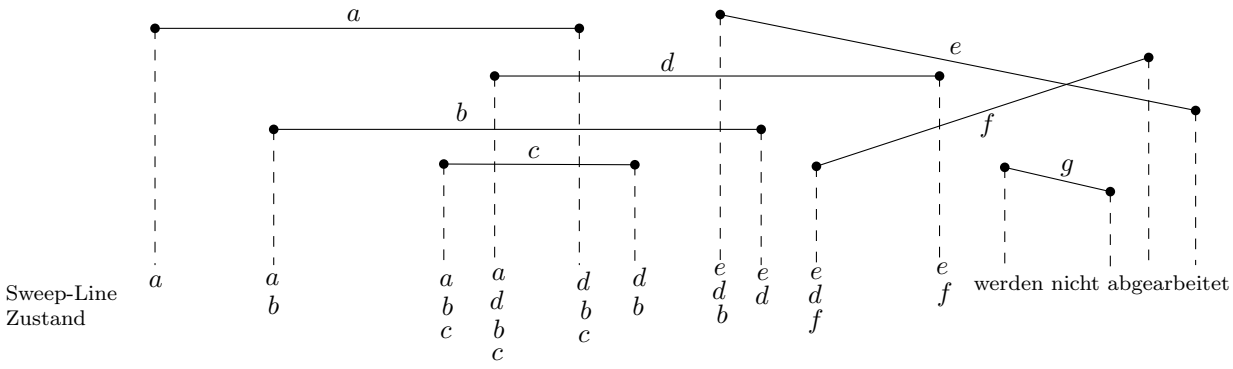


Betrachte ein Gitter bestehend aus n horizontalen und n vertikalen Strecken. Jede horizontale Strecke schneidet alle n vertikalen Strecken. Damit ergeben sich insgesamt n^2 Schnittpunkte.

(b) Um zu überprüfen ob zwei Strecken sich schneiden, soll auf die unten abgebildeten Strecken der aus der Vorlesung bekannte Sweep-Line-Algorithmus INTERSECT angewendet werden. Die potentiellen Haltepunkte (Event-Point Schedule) sind durch gestrichelte Linien markiert. Die ersten beiden Haltepunkte sind bereits mit dem Sweep-Line Zustand beschriftet.

Vervollständigen Sie die Abbildung, indem Sie jeden Haltepunkt mit dem zugehörigen Sweep-Line Zustand beschriften. *Achtung:* potentielle Haltepunkte an denen der Algorithmus nicht anhält sollen nicht beschriftet werden.

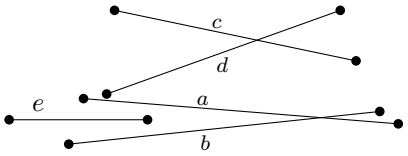
Lösung:



(c) Sei S eine Menge von Strecken mit paarweise disjunkten Endpunkten und seien $a, b, c, d \in S$ vier unterschiedliche Strecken, wobei es in S genau zwei Schnittpunkte zwischen Streckenpaaren gibt, nämlich zwischen a und b sowie zwischen c und d.

Zeigen oder widerlegen Sie: Wenn a und b vor c und d in den Sweep-Line Zustand aufgenommen werden, dann findet der Algorithmus INTERSECT aus der Vorlesung immer den Schnittpunkt zwischen a und b.

Lösung:



Zwar werden a und b früher in den Sweep-Line-Status aufgenommen als c und d, aber die Strecke e erzwingt, dass die Nachbarschaft von a und b später entdeckt wird, als die von c und d. Damit wird der Schnittpunkt von c und d von INTERSECT entdeckt.

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

- ☒ $\Theta(n^2 + n \log n \cdot 3^{k!})$
- ☐ $\Theta(n \log n + 3^{3k} \cdot n!)$
- ☐ $\Theta(2^k \cdot n^k + \sqrt{k} \cdot 3n)$
- ☒ $\Theta(\log n)$

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

Die konvexe Hülle

- ☒ von n Strecken kann in $O(n \log n)$ Zeit berechnet werden.
- ☐ von n Punkten hat immer die Größe mindestens $\Omega(n)$.
- ☒ eines Dreiecks ist das Dreieck selbst.
- ☐ eines einfachen Polygons ist das einfache Polygon selbst.

Ein maximaler st-Fluss in einem gerichteten Graphen $G = (V, E)$ mit Kapazitäten 1

- ☒ kann in polynomieller Zeit berechnet werden, wenn eine minimale Kreisbasis von G gegeben ist.
- ☒ kann den Wert $|V| - 1$ haben.
- ☒ hat ganzzahligen Wert.
- ☒ kann den Wert 0 haben.

Eine Menge von Algorithmen $\{A_\varepsilon \mid \varepsilon \in \mathbb{R}^+\}$ mit Laufzeit $O(\left(\frac{1}{\varepsilon}\right)^5 \cdot 2^\varepsilon \cdot n^3)$ und relativer Approximationsgüte $1 + \varepsilon$ ist ein

- ☒ PAS
- ☒ FPAS
- ☒ AFPAS
- ☒ APAS

Sei G ein Graph und I ein maximales INDEPENDENT SET der Größe k in G. Dann enthält I

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

Sei G ein ungewichteter Graph, T ein Spannbaum in G, sowie B_T die Fundamentalbasis des Kreisraums von G bezüglich T. Dann

- ☒ enthält B_T nur einfache Kreise.
- ☐ ist B_T eine minimale Kreisbasis.
- ☐ ist jede Kante von G in maximal zwei Kreisen von B_T enthalten.
- ☐ ist jede Kante von G in mindestens einem Kreis von B_T enthalten.

Beim (h, k)-Paging

- ☐ hat der Online-Algorithmus einen kleineren Cache zur Verfügung als der optimale Offline-Algorithmus.
- ☒ hat der optimale Offline-Algorithmus mindestens $h - k$ Fehlzugriffe.
- ☒ sind konservative Online-Algorithmen k-kompetitiv.
- ☐ hat der optimale Offline-Algorithmus mindestens so viele Fehlzugriffe wie der Online-Algorithmus.