Nachname:

Vorname:

Matrikelnummer:

Lösungsvorschlag

Karlsruher Institut für Technologie
Institut für Theoretische Informatik

Prof. Dr. P. Sanders 11.03.2025

Klausur Algorithmen II

Aufgabe 1.	Kleinaufgaben	14 Punkte
Aufgabe 2.	Shortest Paths: Dijkstra	10 Punkte
Aufgabe 3.	Reduktionsregeln: MST	12 Punkte
Aufgabe 4.	Flussalgorithmen: Hypergraphen	10 Punkte
Aufgabe 5.	Online-Algorithmen: Bahncard	6 Punkte
Aufgabe 6.	Geometrische Algorithmen	8 Punkte

Bitte beachten Sie:

- Als Hilfsmittel ist nur **ein** DIN-A4 Blatt mit Ihren **handschriftlichen** Notizen zugelassen.
- Schreiben Sie Ihre Antworten nur in blau oder schwarz, mit dokumentenechtem Stift.
- **Schreiben** Sie auf **alle** Blätter der Klausur und Zusatzblätter Ihre **Matrikelnummer**.
- Die Klausur enthält **17 Seiten**.
- Zum Bestehen der Klausur sind 30 Punkte hinreichend.

Lösungsvorschlag

EK

ZK

Aufgabe 1. Kleinaufgaben [14 Punkte]

a. Gegeben sei

- ein Entscheidungsproblem Π mit Eingabegröße n
- ein Monte Carlo Algorithmus A, der Π in Zeit $\mathcal{O}(n)$ mit einseitiger und unabhängiger Fehlerwahrscheinlichkeit $1 - \frac{1}{n}$ löst.

Zeigen Sie: Es gibt einen Monte Carlo Algorithmus, der Π in Zeit $\mathcal{O}(n^2)$ löst und eine einseitige Fehlerwahrscheinlichkeit von $\Theta(1)$ aufweist. [2 Punkte]

Lösung

Wir führen n mal A aus. Da der Fehler einseitig ist, reduziert sich die kombinierte Fehlerwahrscheinlichkeit. Die Fehlerwahrscheinlichkeit ist $(1-1/n)^n \approx 1/e$.

b. Es sei ein Approximationsalgorithmus A für ein Minimierungsproblem Π gegeben. A habe eine Laufzeit von $f(n,\varepsilon)$, um einen Approximationsfaktor von $1+\varepsilon$ für die Eingabegröße n zu erreichen. Geben Sie an, welche der folgenden Laufzeiten A zu einem *Fully Polynomial Time Approximation Scheme* (FPTAS) für Π machen. [2 Punkte]

	$f(n,\varepsilon)$	ja	nein
1.	$\frac{1}{\varepsilon^3}n^2$	☐	☐
2.	$n^{1+\frac{1}{\varepsilon}}$	☐	☐

Lösung

- Ja, denn $\frac{1}{\varepsilon^3}n^2$ ist polynomiell in $\frac{1}{\varepsilon}$ und in n.
- Nein, denn $n^{\frac{1}{\varepsilon}}$ ist nicht polynomiell in $\frac{1}{\varepsilon}$.

c. Es sind zwei Arrays gegeben, eines der Größe n und eines der Größe m. Beide enthalten jeweils keine Duplikate und sind zu groß für den internen Speicher. Wie in der Vorlesung sei M die Größe des internen Speichers und B die Blockgröße ($M \gg B$).

Geben Sie zwei externe Algorithmen A und B an, die alle Elemente ausgeben, die in beiden Arrays enthalten sind (Schnittmenge). Dabei soll die asymptotische Anzahl IOs im Worst-Case

- $\mathcal{O}\left((m+n)\log_B(\frac{n}{M})\right)$ für Algorithmus A betragen und
- $\mathcal{O}\left(\frac{n+m}{B}\log_{M/B}(\frac{n+m}{M})\right)$ für Algorithmus B betragen.

[4 Punkte]

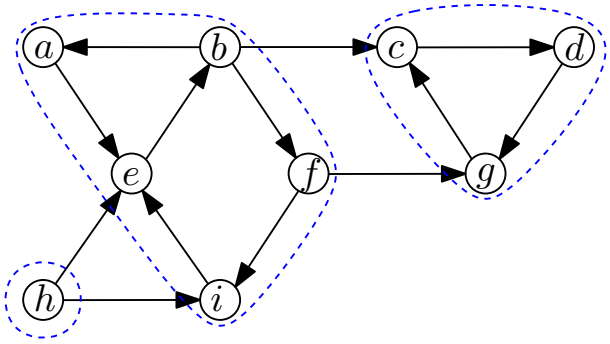
Lösung

A: Externen Suchbaum über Array der Größe n erstellen. Dann jedes Element aus dem Array der Größe m im Suchbaum suchen.
B: Die Vereinigung der Arrays sortieren und dann mit einem Scan aufeinanderfolgende Elemente auf Gleichheit prüfen.

d. Markieren Sie im unten abgebildeten Graphen alle starken Zusammenhangskomponenten (SCCs). Kann man durch Hinzufügen einer einzigen Kante erreichen, dass der Graph aus einer einzigen SCC besteht? Falls ja, geben Sie eine solche Kante an. Falls nein, begründen Sie.

Zwei Kopien. Wenn Sie beide beschriften, machen Sie deutlich, welche Kopie korrigiert werden soll. Andernfalls wird diese Teilaufgabe mit 0 Punkten bewertet. [2 Punkte]

Lösung



Hinzugefügte Kante bzw. Begründung: Kante z.B. von g nach h

e. Gegeben seien zwei parallele Algorithmen A und B, die dasselbe Problem lösen. Sei n die Eingabegröße und p die Anzahl verfügbarer PEs. Der beste sequentielle Algorithmus habe Laufzeit $T_{\text{seq}}(n) = n \log n$. Die Laufzeiten der parallelen Algorithmen seien:

$$T_A(n,p) = \frac{n^2}{p} \qquad \text{bzw.} \qquad T_B(n,p) = \frac{n \log n}{\sqrt{p}}$$

Geben Sie absoluten Speedup und Effizienz der beiden Algorithmen an. Wie viele PEs müssen in Abhängigkeit von n mindestens verfügbar sein, damit Algorithmus A effizienter ist als Algorithmus B? [4 Punkte]

Lösung

Algorithmus	A	B
Speedup	$S_A = \frac{p \log n}{n}$	$S_B = \sqrt{p}$
Effizienz	$E_A = \frac{\log n}{n}$	$E_B = \frac{1}{\sqrt{p}}$

Algorithmus A ist effizienter als Algorithmus B, falls

$$E_A(n,p) > E_B(n,p) \iff \frac{\log n}{n} > \frac{1}{\sqrt{p}} \iff \sqrt{p} > \frac{n}{\log n} \iff p > \frac{n^2}{\log^2 n}$$

Lösungsvorschlag

EK

ZK

Aufgabe 2. Shortest Paths: Dijkstra [10 Punkte]

a. Geben sie einen Graphen an, der negative Kantengewichte aber keinen Kreis enthält, und auf dem Dijkstras Algorithmus eine inkorrekte Lösung ausgibt. [2 Punkte]

Lösung

Start und Ziel direkt mit Gewicht 1 verbinden. Start über einen dritten Knoten mit Ziel verbinden wobei die Kosten der ersten Kante 2 und der zweiten -2 sind.

Anmerkung zur Dijkstra-Implementierung:
Der in der Vorlesung verwendete Pseudocode für Dijkstra kann tatsächlich mit negativen Kantengewichten umgehen; wenn ein neuer kürzester Weg gefunden wird, werden Knoten ein weiteres Mal in die PQ eingefügt (negative Kantengewichte können also zu quadratischer Laufzeit führen, aber nicht zu einem inkorrekten Ergebnis). Da die Aufgabe mit dieser Dijkstra-Implementierung nicht lösbar wäre, wird für die Aufgabe stattdessen angenommen, dass Dijkstra jeden Knoten nur einmal scannt.

b. Es sei ein Graph $G = (V, E, c)$ gegeben, wobei alle Kantengewichte im Intervall $[0, C]$ liegen. Was muss zusätzlich gelten, damit Radix-Heaps in Dijkstras Algorithmus benutzt werden können? Was ist in diesem Fall die asymptotische Worst-Case Laufzeit? [2 Punkte]

Lösung

Die Kantengewichte müssen ganzzahlig sein. Die Laufzeit ist dann $\mathcal{O}(m + n \log C)$.

c. Geben Sie einen Graphen an, auf dem bidirektionaler Dijkstra mehr Knoten scannen muss als normaler Dijkstra. Gehen Sie davon aus, dass Vor- und Rückwärtssuche eine gemeinsame Priority Queue verwenden. [3 Punkte]

Lösung

Verbinde Start und Ziel über einen Knoten wobei die Kosten von beiden Kanten 2 sind. Verbinde Start und Ziel über einen weiteren Knoten wobei die Kosten erst 10 und dann 1 sind. Bei normalem Diskstra wird nur der eigentliche Pfad gescant. Bei bidirektionalem Diskstra wird zuätzlich der zweite Pfad gescant.

d. Geben Sie einen Graphen an, auf dem bei Ausführung von bidirektionalem Dijkstra der letzte gescannte Knoten nicht Teil des kürzesten Weges ist. [3 Punkte]

Lösung

Es gibt wieder zwei Pfade vom Start zum Ziel. Auf dem einen Pfad sind die Kosten 3-3-3, auf dem anderen 5-5.

Lösungsvorschlag

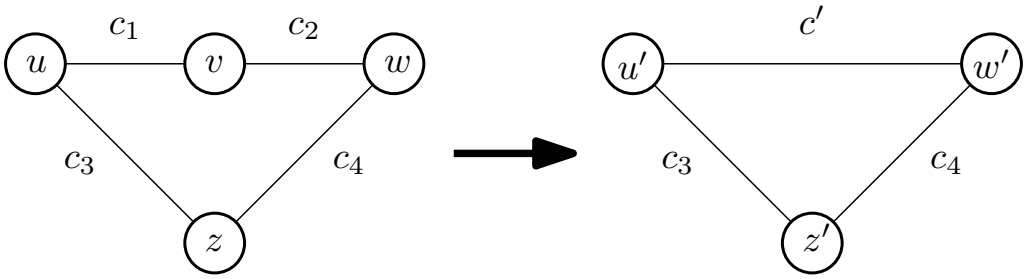
Aufgabe 3. Reduktionsregeln: MST [12 Punkte]

Erinnerung: Ein MST (Minimum Spanning Tree) ist eine Teilmenge der Kanten eines zusammenhängenden, gewichteten Graphen, die alle Knoten verbindet und deren Gesamtkantengewicht minimal ist. Er kann in $\mathcal{O}(|E|\log|V|)$ bestimmt werden.

Die **Kreiseigenschaft** besagt, dass eine Kante e, die in einem Kreis C liegt und schwerer ist als alle anderen Kanten in C, nicht Teil eines MST sein kann.

Der linke gewichtete Graph G soll in den rechten gewichteten Graph G' durch Löschen von v überführt werden, sodass

$$\{u,v\} \text{ und } \{v,w\} \text{ sind im MST von } G \iff \{u',w'\} \text{ ist im MST von } G'$$



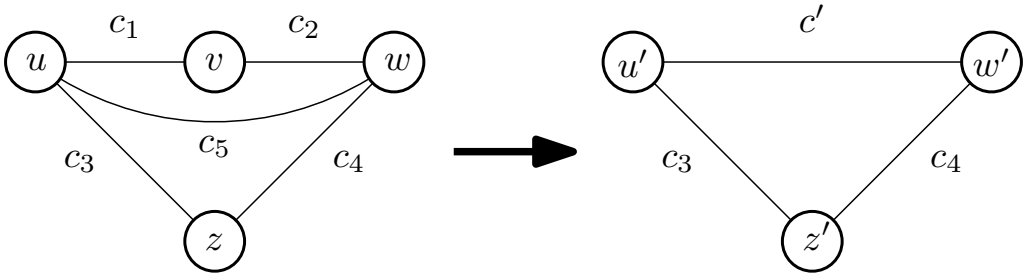
Hierfür wird $c' = \max\{c_1, c_2\}$ gewählt.

a. Falls $\{u',w'\}$ nicht im MST von G' ist und $c_1 \neq c_2$, welche der Kanten $\{u,v\}$ oder $\{v,w\}$ ist nicht im MST von G? [2 Punkte]

Lösung

Die mit dem höheren Gewicht (Kreiseigenschaft).

Nun existiere in G zuätzlich die Kante $\{u,w\}$ mit Gewicht c_5.



b. Analog zur vorigen Situation soll ein Graph G' ensprechend der Abbildung konstruiert werden. Sie dürfen im Folgenden annehmen, dass c_1, c_2 und c_5 paarweise verschieden sind. Geben Sie ein Verfahren an, um

- einen Wert für das Gewicht c' zu bestimmen
- und aus der Information, ob $\{u', w'\}$ im MST von G' ist, zu bestimmen, welche der Kanten $\{u,v\}$, $\{v,w\}$, $\{u,w\}$ im MST von G sind.

Hinweis: Verwenden Sie die Kreiseigenschaft für $\{u,v\}$, $\{v,w\}$ und $\{u,w\}$ und treffen Sie ggfs. eine Fallunterscheidung danach, welche der drei Kanten am schwersten ist. [4 Punkte]

Lösung

Lösche die schwerste der drei Kanten (ist nie im MST erhalten). Falls dies $\{u,w\}$ ist, gehe anschließend analog zur Situation in Teilaufgabe **a.** vor.

Falls die gelöschte Kante $\{u,v\}$ oder $\{v,w\}$ ist, hat v nun Grad 1. Die andere der beiden Kanten ist also immer im MST enthalten. Wähle $c' = c_5$. Es gilt

$$\{u,w\} \text{ ist im MST von } G \iff \{u',w'\} \text{ ist im MST von } G'$$

c. Gegeben sei ein gewichteter zusammenhängender Graph $G = (V,E,c)$, wobei alle Knoten mindestens Grad 2 haben. Außerdem sei $k = |E| - |V|$. Geben Sie einen Algorithms an, der in Zeit $\mathcal{O}(|E| + k \log k)$ den MST von G bestimmt. Die Laufzeit muss nicht begründet werden.

Hinweis: Es gibt maximal $2k$ Knoten mit Grad 3 oder höher, welche insgesamt zu maximal $3k$ Kanten inzident sind. [4 Punkte]

Lösung

Lösche iterativ in Zeit $\mathcal{O}(|E|)$ Knoten mit Grad 2 und ersetze sie durch eine Kante. Bei jedem Löschen wird so vorgegangen, wie in den vorherigen Teilaufgaben beschrieben. Berechne den MST auf dem verbleibenden Graphen in Zeit $\mathcal{O}(k \log k)$ (möglich da der Graph höchstens $2k$ Knoten und $3k$ Kanten hat). Mache die Löschoperationen wieder rückgängig. In jedem Schritt wird der MST für die daraus entstehenden Kanten aktualisiert, entsprechend den vorherigen Teilaufgaben.

Ergänzung:
In bestimmten Fällen können durch die Kontraktionen Knoten mit Grad 1 erstehen. Dies kann analog zu Teilaufgabe **d.** behandelt werden (Knoten löschen und Kanten dem MST hinzufügen).

Alternative Lösung:
Es kann auch unmittelbar ein Algorithmus mit der geforderten Laufzeit angegeben werden, z.B. wie folgt: Arbeite Knoten in aufsteigender Reihenfolge des Grades ab. Dies ist mit einer Bucket Queue in Zeit $\mathcal{O}(|E|)$ möglich. Gebe die leichteste benachbarte Kante aus, anschließend kontrahiere die Kante (Kantenlisten der beiden Knoten aneinanderhängen). Dies erreicht die geforderte Laufzeit, da der Aufwand pro Grad 2 Knoten konstant ist und für die verbleibenden Knoten die Grade bei der Kontraktion mit einer harmonischen Summe abgeschätzt werden können.

Beweis Anzahl Knoten und Kanten (gehört nicht zur Aufgabe):
Sei j die Anzahl Knoten mit Grad 3 oder höher. Dann gilt $2|E| = \sum_{v \in V} d(v) \geq 3j + 2(|V| - j)$. Folglich $2(|V| + k) \geq 2|V| + j$ und somit $j \leq 2k$. Der summierte Grad dieser Knoten ist $2|E| - 2(|V| - j) = 2(|E| - |V| + j) = 2(k + j) \leq 6k$.

d. Der Eingabegraph G enthalte nun auch Knoten mit Grad 1. Beschreiben Sie, wie der Algorithmus aus Teilaufgabe **c.** ergänzt werden kann, so dass er weiterhin in Zeit $\mathcal{O}(|E| + k \log k)$ den MST von G bestimmt. [2 Punkte]

Lösung

Entferne alle Knoten mit Grad 1 und deren Kanten und füge sie dem MST hinzu. Danach kann der Algorithmus aus der vorherigen Teilaufgabe verwendet werden.

Lösungsvorschlag

EK

ZK

Konzeptpapier

Lösungsvorschlag

EK

ZK

Aufgabe 4. Flussalgorithmen: Hypergraphen [10 Punkte]

Ein *Hypergraph-Flussnetzwerk* $H = (V, E, c)$ sei gegeben durch eine Knotenmenge V, eine Menge von *Hyperkanten* E und eine Kapazitätsfunktion $c\colon E \to \mathbb{N}_{>0}$. Eine Hyperkante $e \in E$ kann im Unterschied zu Graphen beliebig viele Knoten verbinden, sie ist also eine beliebige Teilmenge $e \subseteq V$. Zusätzlich sind ein Source-Knoten $s \in V$ und ein Sink-Knoten $t \in V$ gegeben.

Ein gültiger *Hypergraph-Fluss* ist eine Funktion $f\colon V \times E \times V \to \mathbb{N}_{\geq 0}$, wobei $f(u,e,v) = x$ beschreibt, dass ein Fluss mit Wert x ausgehend von u über e nach v fließt. Dabei gilt:

1. Fluss muss über Hyperkanten laufen:

$$f(u,e,v) > 0 \Rightarrow u \in e \text{ und } v \in e$$

2. Summierter Fluss darf die Hyperkanten-Kapazität nicht übersteigen:

$$\sum\{f(u,e,v) \mid u \in V, v \in V\} \leq c(e) \text{ für } e \in E$$

3. Ausgehender Fluss gleich eingehender Fluss:

$$\sum\{f(u,e,v) \mid e \in E, v \in V\} = \sum\{f(v,e,u) \mid e \in E, v \in V\} \text{ für } u \in V \setminus \{s,t\}$$

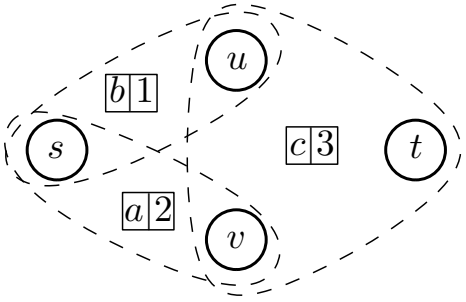
a. Gegeben sei das unten abgebildete Hypergraph-Flussnetzwerk und ein Hypergraph-Fluss f wie folgt (f ist 0 auf allen anderen Eingaben):

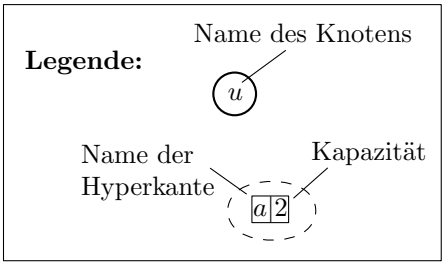
$f(s,a,v) = 1$ $f(v,c,t) = 1$ $f(s,b,u) = 1$ $f(u,c,t) = 1$

Was ist der ausgehende Fluss von s (bzw. eingehende Fluss von t)? Ist f ein maximaler Fluss? Wenn nein, geben Sie einen maximalen Fluss an. [3 Punkte]

Lösung

f hat Wert 2, was jedoch kein maximaler Fluss ist. Ein Fluss mit Wert 3 lässt sich erzeugen, indem wir $f(s,a,v) = 2$ und $f(v,c,t) = 2$ wählen (der Rest bleibt unverändert).





b. Geben Sie an, wie ein Hypergraph-Flussnetzwerk $H = (V_H, E_H, c_H)$ in ein reguläres Flussnetzwerk $G = (V_G, E_G, c_G)$ transformiert werden kann. Hierbei soll für beliebig gewählte $s \in V_H$ und $t \in V_H$ jeder gültige Fluss f_G in G (mindestens) einem gültigen Hypergraph-Fluss f_H in H entsprechen. Insbesondere soll $V_H \subseteq V_G$ gelten und es soll für $v \in V_H$ der aus- bzw. eingehende Fluss von v für f_H und f_G identisch sein.

Beschreiben Sie außerdem kurz, wie sichergestellt wird, dass die Kapazitäts-Bedingung (2.) eingehalten wird.

Hinweis: Ersetzen Sie jede Hyperkante e durch 2 Knoten und $2|e|+1$ gerichtete Kanten, wobei eine der Kanten Kapazität $c(e)$ erhält und die übrigen Kanten Kapazität ∞. [5 Punkte]

Lösung

Das gewünschte äquivalente Graph-Flussnetzwerk G lässt sich erzeugen, indem wir jede Hyperkante e durch die folgende Konstruktion ersetzen:

- Es werden zwei neue Knoten u und v eingefügt
- Für jeden Knoten $x \in e$ werden gerichtete Kanten (x,u) und (v,x) eingefügt, die jeweils Kapazität ∞ haben (jede Kapazität $\geq c(e)$ funktioniert)
- Es wird eine Kante (u,v) mit Kapazität $c(e)$ eingefügt

Ein Hypergraph-Fluss $f(x,e,y) = a$ enspricht einem Fluss auf G mit Wert a für die Kanten (x,u), (u,v) und (v,y). Hierbei werden alle über e laufenden Flüsse in (u,v) aufsummiert. Die in der Konstruktion gewählte Kapazität $c((u,v)) = c(e)$ stellt somit sicher, dass f' in einen Hypergraph-Fluss übersetzt werden kann, der die Kapazitätsbedingung efüllt.

Präzisere Begründung der Korrektheit (nicht in der Aufgabe gefordert):
Es gelten die folgenden Gleichungen für die Übersetzung der Flusswerte:

$$\sum\{f(x,e,y) \mid y \in V\} = f'((x,u)) \text{ für } x \in e$$

$$\sum\{f(x,e,y) \mid x \in V\} = f'((v,y)) \text{ für } y \in e$$

I.A. gibt es mehrere Lösungen um einen Hypergraph-Fluss f aus dem Graph-Fluss f' zu erhalten (die sich darin unterscheiden, welchem der $|e|^2$ möglichen Knotenpaare ein Flusswert genau zugeornet wird), welche jedoch alle in identischem ein- und ausgehenden Fluss für alle Knoten resultieren.

Insbesondere implizieren obige Gleichungen gemeinsam mit der Flusserhaltungs-Bedingung für G, dass folgendes gilt:

$$f'((u,v)) = \sum\{f(x,e,y) \mid x,y \in V\}$$

Dies zeigt, dass die Kapazitätsbedingungen auf H und G äquivalent sind.

Analog lässt sich aus obigen Gleichungen zeigen, dass ein- und ausgehender Fluss für alle $v \in V$ in beiden Netzwerken gleich sind:

$$\sum\{f'((x,u)) \mid u \in V'\} = \sum\{f(x,e,y) \mid e \in E, y \in V\} \text{ für } x \in V \text{ (ausgehend)}$$

$$\sum\{f'((v,y)) \mid v \in V'\} = \sum\{f(x,e,y) \mid e \in E, x \in V\} \text{ für } y \in V \text{ (eingehend)}$$

c. Die Transformation aus Teilaufgabe **b.** soll genutzt werden, um den *Highest-level Preflow-Push*-Algorithmus auf G auszuführen und somit einen maximalen Fluss für H zu berechnen. Geben Sie die asymptotische Worst-Case-Laufzeit in Abhängigkeit von $n = |V_H|$, $m = |E_H|$ und $\ell = \sum_{e \in E_H} |e|$ an. [2 Punkte]

Lösung

Basierend auf dem Hinweis für Teilaufgabe **b.** gilt $|V'| = |V| + 2|E| = n + 2m$ und $|E'| = \sum_{e \in E} 2|e| + 1 = m + 2\ell$. Highest-level Preflow-Push hat nach VL die Laufzeit $\mathcal{O}\left(|V|^2\sqrt{|E|}\right) = \mathcal{O}\left((n+m)^2\sqrt{m+\ell}\right)$.

Nicht gefordert: Da $m \in \mathcal{O}(\ell)$ vereinfacht sich dies zu $\mathcal{O}\left((n+m)^2\sqrt{\ell}\right)$.

Lösungsvorschlag

EK

ZK

Aufgabe 5. Online-Algorithmen: Bahncard [6 Punkte]

Wann sollte man sich die Bahncard50 kaufen? Eine Bahncard50 kostet 100 €, kann nur einmal gekauft werden, und reduziert alle zukünftigen Ticketpreise um 50 %. Im Folgenden sei für das Online-Szenario unbekannt, wie viele Tickets insgesamt gekauft werden, und es muss beim Kauf sofort entschieden werden, ob zuerst die Bahncard50 gekauft wird.

a. Sei K die Gesamtkosten der Tickets (ohne Bahncard-Rabatt). Geben Sie die Kosten der optimalen Offline-Strategie in Abhängigkeit von K an. [2 Punkte]

Lösung

$\min(K, 100 + K/2)$

b. Geben Sie den kompetitiven Faktor der Strategie an, sich nie eine Bahncard50 zu kaufen. [1 Punkt]

Lösung

Faktor 2
Worst Case: Gesamtkosten gehen gegen unendlich

c. Geben Sie den kompetitiven Faktor der Strategie an, sich sofort für das erste Ticket die Bahncard50 zu kaufen. Nehmen Sie an, dass alle Tickets ohne Bahncard50 mindestens 2 € kosten. [1 Punkt]

Lösung

$101/2 = 50.5$
Worst Case: keine weiteren Tickets

d. Geben Sie eine Online-Strategie mit kompetitivem Faktor von maximal $\frac{3}{2}$ an. [2 Punkte]

Lösung

Strategie: BC kaufen wenn durch nächstes Ticket die Gesamtkosten größer oder gleich 200 € werden.

Begründung (nicht gefordert): Der Worst Case ist der Moment, nachdem die BC gekauft wurde. Also $(100 + 200)/(100 + 200/2) = 300/200 = 1.5$

Lösungsvorschlag

EK

ZK

Aufgabe 6. Geometrische Algorithmen [8 Punkte]

a. Gegeben ist eine Menge L von Linien, die parallel zur x-Achse verlaufen. Dabei wird eine Linie $(x_l, x_r, y) \in L$ jeweils durch den linken Endpunkt (x_l, y) und den rechten Endpunkt (x_r, y) beschrieben, wobei $x_l < x_r$. Für einen vertikalen Strahl s ist nun gefragt, wie viele Linien in L von s geschnitten werden. Dabei verläuft ein Strahl von seinem Startpunkt $s = (x, y)$ vertikal nach oben (positive y-Richtung).

Geben Sie

1. einen Vorbereitungs-Algorithmus an, der in Zeit $\mathcal{O}(|L| \log |L|)$ Datenstrukturen erzeugt
2. und einen Abfrage-Algorithmus an, der für einen Strahl s in Zeit $\mathcal{O}(\log |L|)$ berechnet, wie viele Linien in L von s geschnitten werden.

Sie dürfen annehmen, dass alle x- und y-Koordinaten jeweils paarweise verschieden sind.

Hinweis: Nutzen Sie zwei Wavelet-Trees. [4 Punkte]

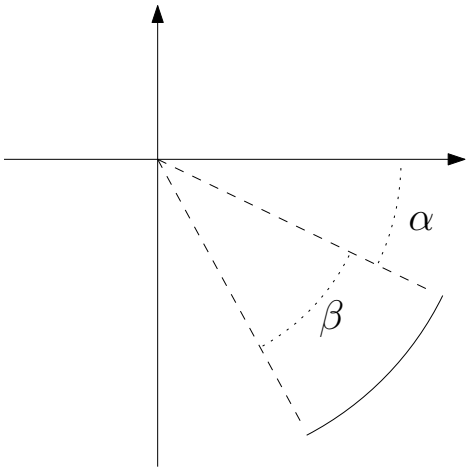
Lösung

Vorbereitung: Einen Wavelet-Tree für die linken Endpunkte und einen für die rechten Endpunkte konstruieren, Zeit $\mathcal{O}(|L| \log |L|)$.

Abfrage: Dominance Count des Strahlursprungs s (gibt Anzahl der Punkte oben rechts von s zurück) beim Wavelet-Tree der rechten Endpunkte minus Dominance Count für s beim Wavelet-Tree der linken Endpunkte. Alternativ können Counting Queries für einen beliebigen der beiden oberen Quadranten benutzt werden.

b. Gegeben ist eine Menge K an Kreisbögen auf dem Einheitskreis, mit Zentrum im Ursprung $(0,0)$. Dabei wird ein Kreisbogen $(\alpha,\beta) \in K$ jeweils durch den Startwinkel $\alpha \in \mathbb{R}$ und die Bogenlänge $\beta \in \mathbb{R}$ beschrieben. Wir betrachten beliebige Strahlen, die dem Ursprung $(0,0)$ entspringen. Geben Sie einen Algorithmus an, der in Zeit $\mathcal{O}(|K|\log|K|)$ berechnet, wie viele Kreisbögen aus K durch einen solchen Strahl maximal geschnitten werden können.

Der Startwinkel $0 \leq \alpha < 360°$ ist der Winkel im Uhrzeigersinn relativ zur x-Achse, der Endwinkel ergibt sich durch $\alpha + \beta \bmod 360°$, wobei immer $\beta < 360°$ gilt. Sie dürfen annehmen, dass alle Start- und Endwinkel paarweise verschieden sind. [4 Punkte]



Lösung

Radiale Sweepline 360° im Uhrzeigersinn. Für jeden Bogen ein Startevent bei α und ein Endevent bei $\alpha + \beta \bmod 360°$. Events sortieren und bei beliebigem Winkel starten. Counter und Max auf die Anzahl Kreisbögen, die den Startwinkel beinhalten, initialisieren. Bei Startevent Counter erhöhen und Max ggf. updaten. Bei Endevent Counter reduzieren.

Lösungsvorschlag

EK

ZK

Konzeptpapier