Zweite Runde Theorie

Aufgaben

Swiss Olympiad in Informatics

4. März 2017

Anweisungen

Die Lösungen werden nach ähnlichen Kriterien wie die theoretischen Aufgaben der ersten Runde bewertet. Die wichtigsten Kriterien sind die Korrektheit und die asymptotische Laufzeit der Lösung. Die Qualität der Beschreibung und die Argumente für die Korrektheit der Lösung werden ebenfalls bewertet.

Falls ein Algorithmus verlangt ist, solltest du dich als Richtlinie an folgende Struktur halten:

1. Beschreibe die Idee hinter deinem Algorithmus so verständlich wie möglich.
2. Erkläre, wieso der Ansatz die Aufgabe korrekt löst.
3. Analysiere die asymptotische Laufzeit und den asymptotischen Speicherverbrauch.
4. Schreibe ein Lösungsprogramm in Pseudocode auf. Du kannst einfache Teile wie Eingabe/Ausgabe weglassen und auch mathematische Ausdrücke benutzen.

Wenn ein Teil deiner Lösung für mehrere Teilaufgaben gilt, reicht es, ihn einmal aufzuschreiben und von da an auf ihn zu verweisen.

Fleptonen

Maus Stofl ist ein brillianter theoretischer Physiker und hat dich gerade über seine neueste Entdeckung informiert, die Fleptonen! Ein Fleptonpaar kann spontan aus dem Nichts an einer beliebigen Stelle im Raum erscheinen; die zwei Fleptonen separieren sich instantan. Nach der Seperation fliegen diese beiden Fleptonen einen beliebigen Pfad durch den Raum, bis sie wieder miteinander kollidieren und gleich darauf verschwinden.

Stofl hat noch eine weitere merkwürdige Eigenschaft der Fleptonen gezeigt: Ein Flepton wird nie einen Ort zweimal besuchen und wird ebenfalls nie einen Ort durchqueren, den das andere Flepton bereits besucht hat. Ausserdem fliegt ein Flepton nie an einen Ort, den es in kürzerer Zeit hätte erreichen können. (Also sind die Pfade der Fleptonen immer kürzeste Pfade.)

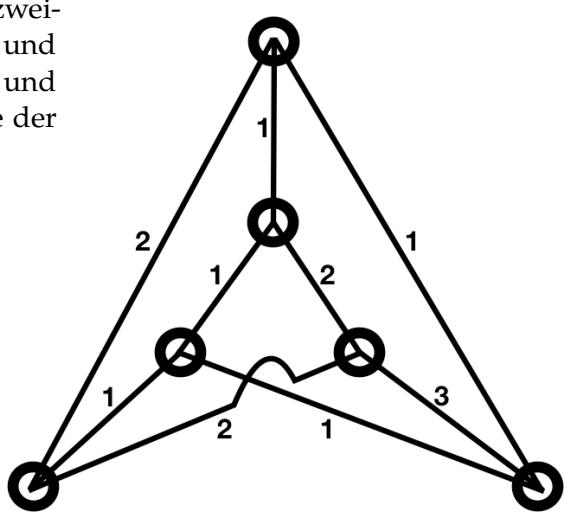
Deine Aufgabe ist es nun, Stofl zum Nobelpreis zu verhelfen, den er nur dann bekommt, wenn er die Fleptonen durch ein Experiment nachweisen kann. Maus Stofl hat bereits ein Netzwerk aus m molekularen Röhren und n Detektoren gebaut, in denen er hofft, ein Fleptonpaar zu messen. Eine molekulare Röhre ist immer zwischen zwei Detektoren und wegen einer praktischer Limitierung können zwei Detektoren höchstens einmal miteinander verbunden sein. Ein Flepton bewegt sich immer mit der Lichtgeschwindigkeit und fliegt deswegen mit konstanter Geschwindigkeit. Da die molekularen Röhren so klein sind, sind die Fleptonen gezwungen sich in eine Richtung zu bewegen (sie können nicht umkehren). Die Länge einer Röhre wird gemessen anhand der Zeit, welche ein Flepton benötigt, um diese Röhre zu durchqueren (in Nanosekunden ns).

Du sollst nun Stofl helfen, zwei Detektoren zu finden, zwischen denen Fleptonen erstellt bzw. zerstört werden können, sodass Stofl diese beiden Detektoren genauer untersuchen kann.

Formale Beschreibung Gegeben ein gewichteter ungerichteter einfacher Graph, finde zwei *kürzeste* Pfade, welche beide am gleichen Knoten starten und am gleichen Knoten enden, die selbe Länge haben und keine andere Knoten untereinander teilen. (Ein Pfad kann einen Knoten höchstens einmal besuchen.)

Teilaufgabe 1: Lösen eines Beispiels (10 Punkte)

Finde zwei solche *kürzeste* Pfade zwischen zwei Verzweigungen im folgenden System von Molekularröhren und markiere sie. Die Kreise stellen eine Verzweigung dar und die Nummern an den Röhren beschreiben die Länge der Röhren (bzw. die Geschwindigkeit der Fleptonen).



Teilaufgabe 2: Algorithmus konzipieren (50 Punkte)

Erstelle einen Algorithmus für Stofl, der für ein gegebenes Netzwerk von Molekularröhren zwei solche *kürzeste* Pfade findet, durch welche sich ein Fleptonenpaar bewegen könnte (Erstellung bis Zerstörung). Argumentiere warum dein Algorithmus *korrekt* ist und schreibe implementiere ihn in *Pseudocode*. Beschreibe die *asymptotische Laufzeit* (wie viel Zeit in Abhängigkeit von m und n dein Algorithmus braucht) und den benutzten *Speicherverbrauch* deines Algorithmus.

Teilaufgabe 3: Konfiguration für jedes nicht-benachbarte Detektorenpaar (40 Punkte)

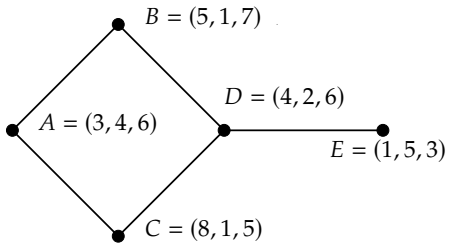
Verflixt! Stofl hat fast all sein Geld für Detektoren ausgegeben und er hat noch immer keine Fleptonen gefunden. Wegen seinem kleinen Budget kann Stofl nur noch gleich lange Röhren kaufen. Zusätzlich möchte er die Wahrscheinlichkeit erhöhen, Fleptonen zu finden. Hilf ihm eine komplett neue Konfiguration von beliebig vielen gleichlangen Röhren zwischen n Detektoren zu konstruieren, sodass jedes *nicht-benachbarte* Paar von Detektoren Fleptonen erzeugen und zerstören kann (Ein Paar von Detektoren ist nicht-benachbart, wenn es keine direkte Röhre zwischen ihnen gibt). Umso weniger Kanten du brauchst, umso mehr Punkte erhälst du. Zum Beispiel für $2 \cdot n$ erhälst du mehr Punkte als für $5 \cdot n$.

Teleport

Die Mäuse auf einem weit entfernten Planeten reisen zwischen ihren Städten mithilfe einem Netwerk von Teleportern. Ein Teleporter besteht aus 3 Sendern, welche alle in einer unterschiedlichen Frequenz senden. Keine zwei Teleporter haben alle drei Senderfrequenzen gleich.

Eine Maus kann sich nur zwischen zwei Teleportern bewegen, wenn keine der 6 Sender auf der gleichen Frequenz senden. Dies ist so, da sich die Sender gegenseitig stören und somit unbeabsichtigte Nebenwirkungen auftreten könnten (Spaghettifizierung, zusätzliche Ohren, usw.).

Betrachte folgendes Beispiel mit 5 Teleporter A, B, C, D und E. Die Verbindungen zeigen alle möglichen Paare, zwischen welchen eine direkte Teleportation möglich ist:



Man stellt fest, dass um sich zwischen zwei Teleportern zu bewegen, man sich manchmal über mehrere Zwischenteleporter bewegen muss. Möchte man sich zum Beispiel zwischen B und C bewegen, muss man über A (oder D) einen Zwischenteleport einlegen. Formal definieren wir die minimale Reisedistanz zwischen zwei Teleportern x und y als $d(x, y)$. Im vorhin genannten Fall hätten wir $d(B, C) = 2$.

Die Alienmäuse sind sehr stolz auf ihr Transportationsnetzwerk und behaupten es sei ziemlich effizient: Man müsse nicht all zu viele Teleporter verwenden um zwischen zwei Orten zu reisen.

Ein objektives Mass dafür ist die Maximaldistanz des Gesamtnetzwerks, welches die maximale kürzeste Distanz zwischen beliebigen zwei Teleportern a und b (also $\max d(a, b)$) ist. Im Beispiel oben ist die Maximaldistanz $d(A, E) = 3$.

Teilaufgabe 1: Lösen anhand eines Beispiels (10 Punkte)

Wir haben 7 Teleporter mit den folgenden Frequenzen:
$A = (6, 8, 3)$, $B = (4, 7, 0)$, $C = (5, 6, 2)$, $D = (1, 7, 3)$, $E = (2, 4, 5)$, $F = (6, 8, 1)$, $G = (4, 8, 0)$.
Liste alle Paare von Teleporter auf, zwischen denen Maus Stofl direkt reisen kann.

Teilaufgabe 2: Netzwerk mit Maximaldistanz (45 Punkte)

Stofl wundert sich, ob es ein Netzwerk gibt, bei dem die Maximaldistanz zwischen zwei Teleportern gross ist.

Aufgabe Gebe die Senderfrequenzen von mindestens 2 Teleportern und markiere x und y, sodass die Distanz zwischen x und y so gross wie möglich ist.

Bewertung Für ein Netzwerk mit einer maximalen Distanz ℓ, erhälst du $\min(5\ell - 15, 45)$ Punkte. Das heisst, wenn die Maximaldistanz 12 oder grösser ist, bekommst du die volle Punktzahl.

Beispiel Die Teleporter im oberen Beispiel sind nicht so weit voneinander entfernt. Die Maximaldistanz zwischen A und E und beträgt 3. Ein solches Netzerk erhält 0 Punkte :)

Teilaufgabe 3: Schranke für die Maximaldistanz (45 Punkte)

Stofl verzweifelt am Versuch, ein Netzwerk zu erstellen, dessen Maximaldistanz 2017 ist und kommt auf die Vermutung, dass diese Distanz gar nicht möglich ist.

Aufgabe Beweisen sie, dass die Maximaldistanz eines Netzwerkes kleiner als 2017 ist. Versuche diese obere Schranke so weit hinunter zu bekommen wie möglich.

Bewertung Jeder korrekte Beweis für eine beliebige Grösse kleiner als 2017 erhält 30 Punkte. Falls die Konstante höchstens zwei Mal grösser ist als das Optimum, erhälst du 45 Punkte. Andere Schranken werden ähnlich benotet.

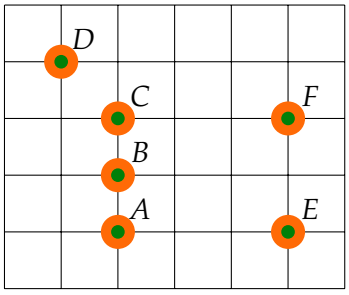
Karottengarten

Maus Stofl hat angefangen, seine Karotten selbst anzubauen. Da Maus Stofl ein Wissenschaftler ist, ist ihm das reine Anbauen sehr schnell zu langweilig geworden und er hat seine Aufmerksamkeit den Parasiten seiner Plantage zugewandt. Maus Stofl glaubt einen neuen Parasiten entdeckt zu haben und hat ihn auch gleich nach sich selbst benannt: "Stoffulus". Das interessante an Stoffulus ist, wie er sich im Karottenbeet fortpflanzt. Eine Infektion startet immer mit 2 infizierten Karotten. Eine gesunde Karotte wird genau dann infiziert, wenn die Distanz zu einer infizierten Karotte kleiner oder gleich ist, als die Distanz zwischen zwei bereits infizierten Karotten.

Maus Stofl möchte nun folgendes herausfinden: Wenn er 2 Karotten mit Stoffulus infiziert, dann wird sich der Parasit ausbreiten bis er keine weiteren Karotten mehr erreichen kann, er ist dann im "Endstadium". Welche Karotten im Endstadium infiziert sind, hängt davon ab, mit welchen 2 Karotten man startet. Maus Stofl möchte nun wissen, wie viele verschiedene solche Endstadien es in seinem Beet gibt. Es gibt aber extrem viele Möglichkeiten, um 2 Karotten zu infizieren ($\binom{N}{2} = \frac{N(N-1)}{2}$ um genau zu sein). Maus Stofl hat keine Lust so viele Experimente durchzuführen und bittet dich um Hilfe.

Formale Beschreibung Gegeben sind n Punkte $p_1, \ldots, p_n$, wobei $p_i = (x_i, y_i) \in \mathbb{R}^2$ in einer Ebene (Stofls Garten). Jedes Experiment startet mit zwei infizierten Punkten: $p_a \neq p_b$. Ein Punkt p_i wird infiziert, wenn es 3 (nicht zwingend verschiedene) infizierte Punkte p_v, p_w, p_j gibt, so dass $\text{dist}(p_i, p_j) \leq \text{dist}(p_v, p_w)$[1]. Sobald kein weiterer Punkt mehr infiziert werden kann, ist man im Endstadium. Für jedes Paar p_a und p_b gibt es ein Endstadium. Wie viele *verschiedene* solche Endstadien gibt es? Zwei Endstadien sind verschieden, wenn es (mindestens) eine Karotte gibt, welche in einem der beiden Endstadien aber nicht in beiden infiziert ist.

Beispiel In diesem Beispiel sind alle Karotten in einem Gitter angeordnet (die Kreise stehen für Karotten). Es gibt 2 verschiedene Endstadien: (a) alle (b) nur E und F.

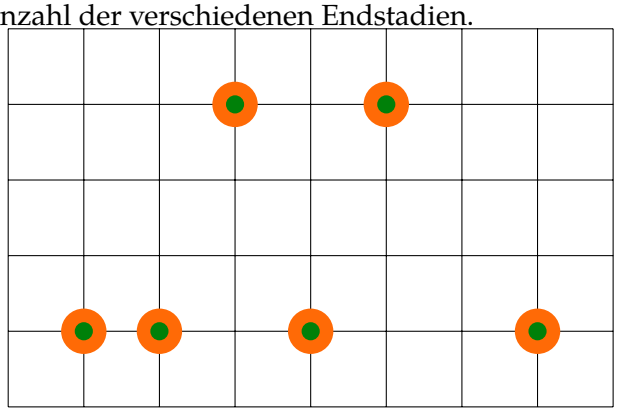


Was passiert, wenn A und B infiziert sind? Da $d(A, B) \geq d(B, C)$ wird auch C befallen. Wegen $d(A, C) \geq d(C, D)$ kommt auch noch D hinzu. Dadurch kann die Infektion nach rechts gelangen: $d(A, D) \geq d(A, E)$ und $d(A, D) \geq d(C, F)$, somit sind alle Karotten befallen.

[1] dist ist der euklidische Abstand $\text{dist}((x_1, y_1), (x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Teilaufgabe 1: Lösen anhand eines Beispiels (10 Punkte)

Für dieses Beispiel nehmen wir ebenfalls an, dass die Karotten in einem Grid angeordnet sind. Wie viele verschiedene Endstadien gibt es? Schreibe eine natürliche Zahl auf, die Anzahl der verschiedenen Endstadien.



Teilaufgabe 2: Algorithmus konzipieren (90 Punkte)

Entwickle einen Algorithmus für Stofl, der als Eingabe eine Liste $L = [(x_1, y_1), \ldots, (x_n, y_n)]$ von Punkten nimmt und eine einzige Zahl ausgibt, die Anzahl möglicher Endstadien.

Wie üblich, erkläre deinen Algorithmus mit *Pseudocode*, analysiere dessen *asymptotische Laufzeit* (wie lange in Abhängigkeit von der Anzahl Karotten n dein Algorithmus hat) und den benutzten *Speicher* und argumentiere dessen *Korrektheit*.

Cake Protection Agency

Mausland hat mehrere polizeiliche Elite-Einheiten mit unterschiedlichen Einsatzgebieten. Eine der wichtigsten ist sicherlich die CPA, die Kuchenschutzpolizei (*Cake Protection Agency*), die damit beauftragt ist, alle Küchen im Land vor Attacken zu schützen.

Die CPA hat gerade von einem bevorstehenden Angriff eines äussert gefährlichen Kuchenterroristen, Maus Gehr, erfahren. Anscheinend soll er planen von zu Hause aus zu einem köstlichen Kuchen zu laufen und diesen auf den Boden zu werfen. Die CPA Agenten können ihn nur stoppen, wenn sie entweder strikt früher beim Kuchen sind (und ihn so an einen sichereren Ort bringen können) oder Maus Gehr vorher abfangen können, um ihn festzunehmen. Um Maus Gehr vor dem Kuchen abzufangen, genügt es wenn ein CPA Agent gleichzeitig mit Gehr an einem Ort ist.

Deine Aufgabe ist es nun, den noblen Agenten der CPA zu helfen Kuchenterrorist Gehr zu stoppen.

Formale Beschreibung Mausland ist als gerichteter *gewichteter* Graphen $G = (V, E)$ mit $n + 3$ Knoten $V = \{v_{\text{cpa}}, v_{\text{gehr}}, v_{\text{cake}}, v_1, \dots v_n\}$ gegeben, der Startbasis der CPA Agenten, der Anfangsposition von Maus Gehr, der Position des Kuchens und den verbleibenden Kreuzungen in Mausland. Von jedem Punkt kann man den Kuchen erreichen (not a lie). Eine Kante $e = (u, v, t) \in E$ verbindet $u \in V$ zu $v \in V$ mit der ganzzahligen Fahrzeit $t \geq 0$. Merke, dass alle Strassen in Mausland Einbahnstrassen sind; die Kante (u, v, t) kann also nur gebraucht werden um von u nach v zu kommen, aber nicht von v nach u (es könnte sein, dass der Graph noch eine weitere Kante (v, u, t') hat, die dies erlauben würde). Die CPA Agenten können Maus Gehr stoppen, falls entweder einer früher beim Kuchen ist als Maus Gehr oder falls einer zur gleichen Zeit wie Maus Gehr den Knoten $v \neq v_{\text{cake}}$ betritt.

Teilaufgabe 1: Späteste Startzeit (15 Punkte)

Maus Gehr war nicht vorsichtig genug und sein geheimer Plan mit der Route $p = (v_{\text{gehr}}, \dots, v_{\text{cake}})$, wie er zum Kuchen kommt, sowie seine (ganzzahlige) Startzeit T wurden den Agenten zugespielt. Beschreibe einen Algorithmus, der die späteste (ganzzahlige) Zeit T' findet, so dass ein CPA Agent die Position v_{cpa} verlässt und Maus Gehr immer noch stoppen kann, in dem er entweder früher beim Kuchen ankommt oder ihn auf dem Weg abgfängt (auch gleichzeitig). Gib Argumente warum dein Algorithmus korrekt ist, schreibe Pseudocode und analysiere dessen Laufzeit und Speicherverbrauch.

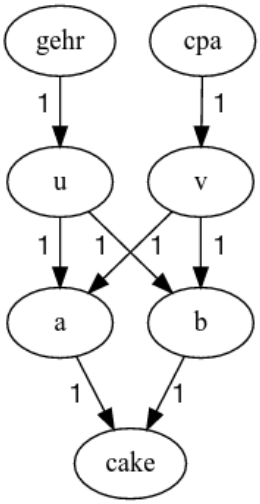
Teilaufgabe 2: Beliebig viele Agenten (35 Punkte)

Maus Gehr war nun etwas vorsichtiger und seine Route ist nicht bekannt geworden. Gehe davon aus, dass Maus Gehr seinen Heimknoten zum (ganzzahligen) Zeitpunkt T verlässt und dass die CPA beliebig viele Agenten aussenden kann. Finde den spätesten (ganzzahligen) Zeitpunk T', wann diese Agenten ausgesandt werden müssen, um

Maus Gehr noch zu stoppen. Gib Argumente warum dein Algorithmus korrekt ist und analysiere die Laufzeit.

Merke, dass Maus Gehr immer weiss wo die CPA Agenten sind und umgekehrt. Desweiteren, falls Maus Gehr und ein CPA Agent zum genau gleichen Zeitpunkt eine Entscheidung treffen müssen (eine ausgehende Kante wählen), dann muss sich Maus Gehr zuerst entscheiden und der CPA Agent weiss dann, welche Kante er genommen hat.

Im folgendem Graphenbeispiel haben alle Kanten Gewicht 1 und dann ist *ein* CPA Agent, der zur gleichen Zeit wie Maus Gehr losgeht, immer in der Lage ihn zu stoppen. Im ersten Schritt wird Maus Gehr zum Knoten u und der CPA Agent zum Knoten v gehen. Danach muss sich Maus Gehr zuerst zwischen Knoten a und b entscheiden, und der CPA Agent kann seine Wahl einfach nachmachen und ihn so dort (a oder b) festnehmen.



Teilaufgabe 3: Ein Agent (50 Punkte)

Das Setup sei das gleiche wie in Subtask B, jedoch steht nur ein CPA Agent zur Verfügung. Bestimme wiederum den spätesten (ganzzahligen) Zeitpunkt T', wann der CPA Agent die CPA Basis v_{cpa} verlassen muss um Maus Gehr noch zu stoppen. Gib Argumente warum dein Algorithmus korrekt ist, schreibe Pseudocode und analysiere die Laufzeit und Speicherverbrauch.