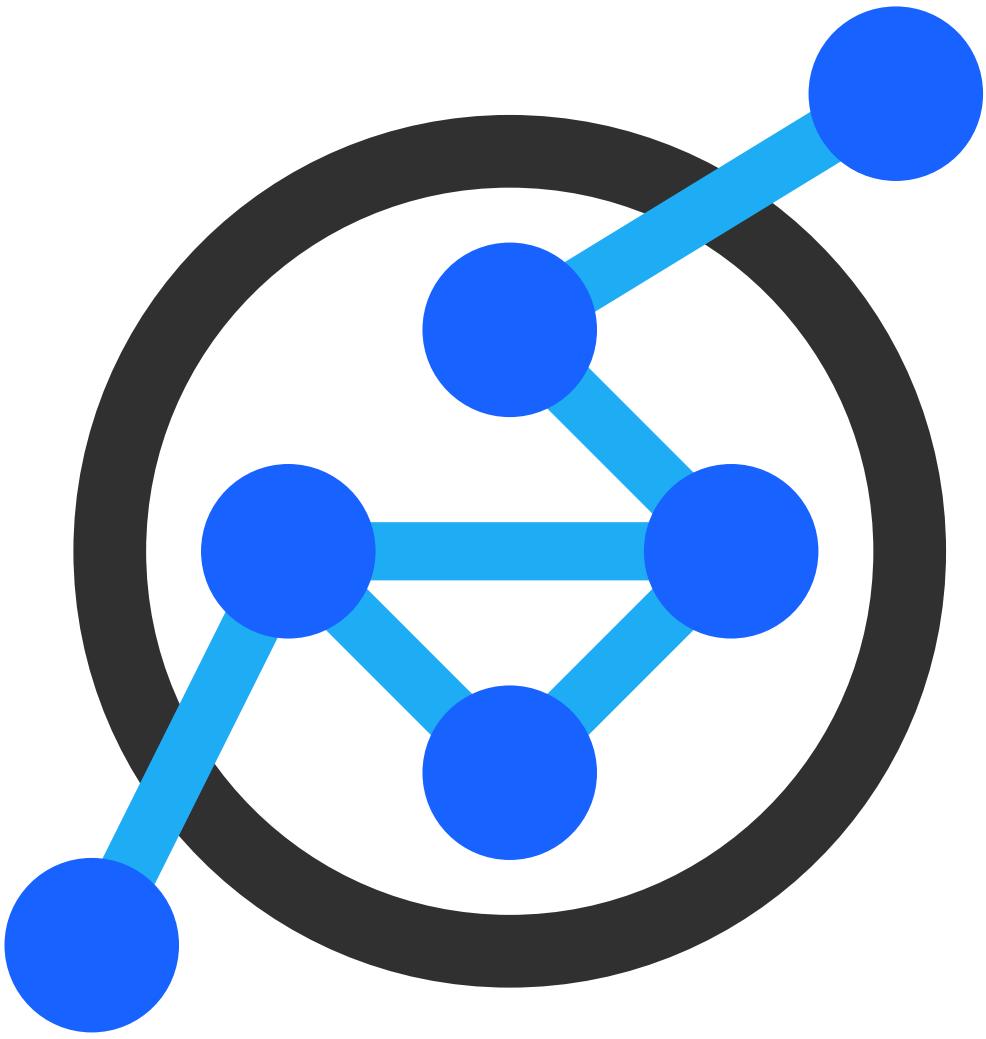
Zweite Runde Theorie

Aufgaben



Swiss Olympiad in Informatics

6./7. März 2021

Anweisungen

- Du hast 5 Stunden und 30 Minuten Zeit, um die Aufgaben zu lösen. Stelle sicher, dass du genügend Zeit zum Einsenden einplanst. Wir empfehlen dir, spätestens nach 5 Stunden eine Blätter zu scannen und hochzuladen. Wenn du danach noch Zeit hast, kannst du weiterarbeiten und eine verbesserte Version später noch hochladen.
- Der Wettbewerb ist “Open Book”; du darfst das Internet, Bücher, alte Lösungen, und so benutzen.
- Falls du Fragen während der Runde hast, frage uns via E-Mail (`info@soi.ch`). Wir geben dir keine Hinweise, wie die Aufgabe zu lösen ist, deshalb versuche, klare Ja/Nein-Fragen zu stellen.
- Deine Lösung muss vollständig von Hand geschrieben werden (mit Papier und Stiften).
- Schreibe lesbar.
- Deine Lösung sollte keine externe Quellen (bekannte Algorithmen, Blog-Artikel, Papers) referenzieren, sondern alles von Grund auf erklären. Ausgenommen davon sind das, was im SOI Wiki in den Abschnitten “First Round” und “Second Round” steht oder vorausgesetzt wird.
- Du sollst ein PDF pro Aufgabe hochladen. Stelle sicher, dass die Auflösung gut genug ist, dass wir alles lesen können. Einige Megabytes pro Aufgabe sollte ausreichen (wir setzen eine Grenze von 100 MiB).
- Wir schauen uns nur die letzte Einsendung jeder Aufgabe an. D.h. du kannst anfangs eine Version hochladen und diese später aufbessern.
- Du kannst eine Vorschau deiner Einsendung sehen, wenn du auf "Detail"klickst. Das erlaubt dir sicherzustellen, dass du die richtige Datei hochgeladen hast.
- Solltest du technische Probleme mit deinem Scanner oder dem Grader haben, kannst du uns notfalls die Aufgaben auch manuell und in geringerer Auflösung schicken. Sende dieses zu Johannes Kapfhammer über E-Mail, Discord oder Signal/WhatsApp (Kontaktangaben gelöscht für das öffentliche Task-Archiv).

Bewertung

Deine Lösung wird anhand ihrer Korrektheit und ihrer asymptotischen Laufzeit und Speichernutzung bewertet, sowie der Begründung dieser Punkte. Wir erwarten, dass du einen Beweis oder eine Beweisskizze für die Korrektheit und Laufzeit/Speichernutzung lieferst.

Du kannst jederzeit auf den Inhalt des SOI Wikis und 2H verweisen. Du musst nicht erklären, wie z.B. Dijkstra funktioniert, aber solltest begründen, wo und wieso man ihn anwenden kann. Im Fall von Dijkstra muss der Graph klar definiert sein und seine Kantengewichte dürfen nicht negativ sein.

Der Algorithmus sollte genug detailliert beschrieben sein, dass man ihn anhand der Erklärung programmieren könnte. Wichtig ist hier auch, dass alle verwendeten Datenstrukturen genau beschrieben werden. Am einfachsten ist es erfahrungsgemäss, kurz einen Pseudocode hinzuschreiben.

Wir empfehlen dir, dich an folgende Struktur zu halten:

1. Beschreibe die Idee hinter deinem Algorithmus so verständlich wie möglich.
2. Gib Pseudocode oder erkläre, wie man den Algorithmus implementieren könnte.
3. Begründe, wieso der Ansatz die Aufgabe korrekt löst.
4. Analysiere die asymptotische Laufzeit und den asymptotischen Speicherverbrauch.

Wenn ein Teil deiner Lösung für mehrere Teilaufgaben gilt, reicht es, ihn nur einmal aufzuschreiben und von da an auf ihn zu verweisen. Beachte aber, dass ein allgemeiner Algorithmus die vorherigen Teilaufgaben zwar lösen kann, aber nicht zwingend optimal ist.

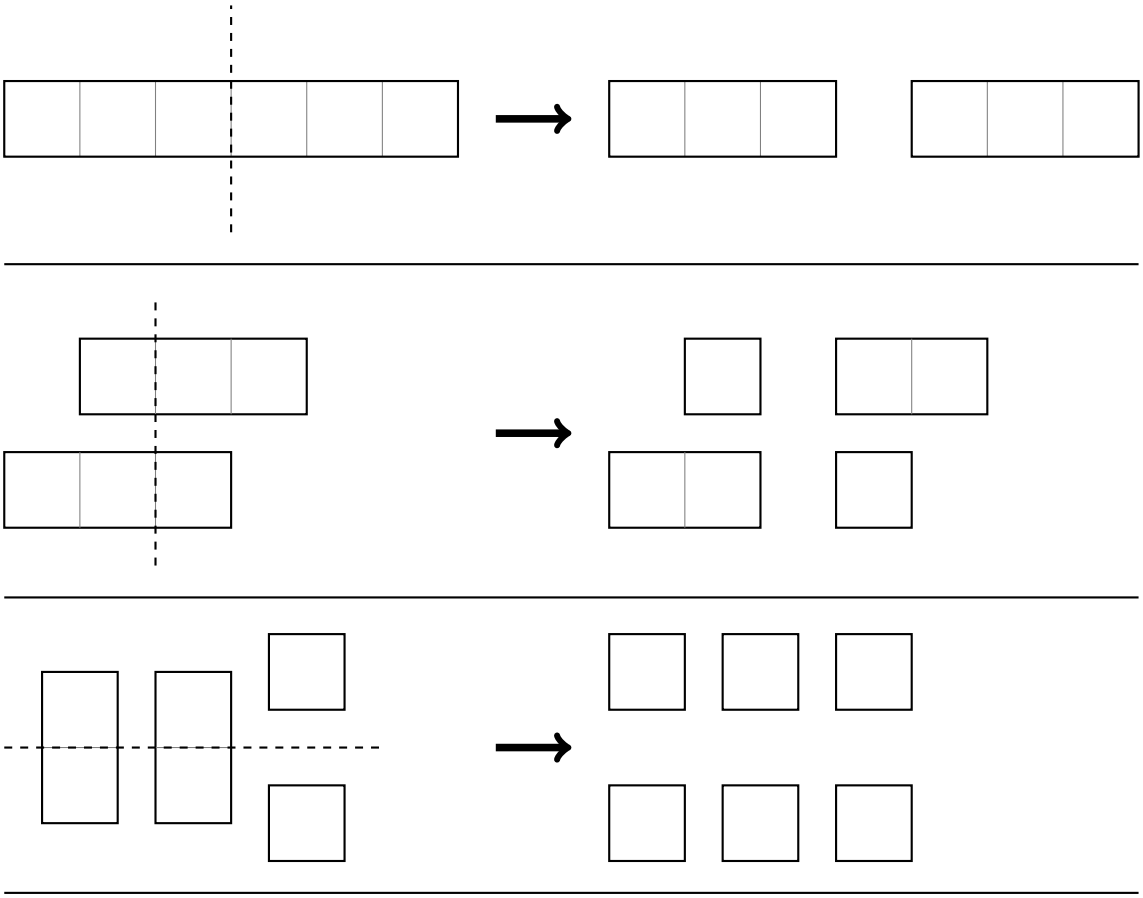
Viel Erfolg!

Laserschnitt

Maus Binna hat kürzlich einen Laserschneider erhalten. Sie möchte damit eine $n \times m$ Tafel Schokolade in Quadrate der Grösse 1×1 zerschneiden. Der Laserschneider funktioniert folgendermassen: Maus Binna kann ein oder mehrere Stücke Schokolade aneinanderreihen. Sie kann sie verschieben und/oder drehen. Dann fährt der Laser entlang einer geraden Linie, wobei alle Stücke, auf die der Laser trifft, zerschnitten werden. (Mit anderen Worten, in einem einzelnen Schritt kann Binna beliebig viele Stücke entlang einer geraden Linie zerschneiden.) Was ist die minimale Anzahl Schnitte, die Maus Binna machen muss?

Formal ist ein $n \times m$ Rechteck gegeben, und du musst die minimale Anzahl Operationen finden, um es in 1×1 Quadrate aufzuteilen. In einer einzelnen Operation kannst du zuerst jedes Stück ausrichten, das bedeutet jedes Stück verschieben und/oder drehen. Dann kannst du eine Linie auswählen, und jedes Stück, welches diese Linie schneidet, wird entlang der Linie in zwei Stücke geteilt. Damit ist eine einzelne Operation beendet.

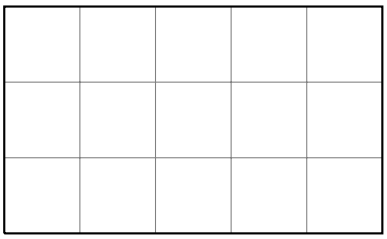
Eine 1×6 Tafel in 3 Schnitten zerteilen



- Die gestrichelte Linie zeigt wo der Laser entlangfährt.
- Von einer Zeile zur nächsten werden die Stücke umgeordnet.

Die Gesamtzahl der Schnitte für diese 1×6 Tafel war 3, und man kann beweisen, dass dies optimal ist.

Teilaufgabe 1: Eine 3×5 Tafel (10 Punkte)



Gib die minimale Anzahl Schnitte an, die Maus Binna braucht, um eine 3×5 Tafel Schokolade in 15 einzelne Stücke zu schneiden.

Teilaufgabe 2: Eine lange, dünne Tafel (35 Punkte)

Entwirf einen Algorithmus, der für eine $1 \times m$ Tafel Schokolade die minimale Anzahl Schnitte berechnet. Beweise die Korrektheit und gib die asymptotische Laufzeit und Speichernutzung an.

Teilaufgabe 3: Allgemeiner Fall (55 Punkte)

Entwirf einen Algorithmus, der für eine $n \times m$ Tafel Schokolade die minimale Anzahl Schnitte berechnet. Beweise die Korrektheit und gib die asymptotische Laufzeit und Speichernutzung an.

Flugzeugbestuhlung

Es gibt ein Flugzeug mit $R \cdot C$ Sitzen, die in R Zeilen und C Spalten organisiert sind. Jedem Sitzplatz können eindeutige Koordinaten (r, c) zugeordnet werden (wobei $1 \leq r \leq R, 1 \leq c \leq C$). Wir sagen, dass zwei Sitze Nachbarn sind, wenn sie in derselben Reihe direkt nebeneinander liegen, oder wenn einer vor dem anderen liegt. Formal sind zwei Plätze (r_i, c_i) und (r_j, c_j) genau dann Nachbaren, wenn $|r_i - r_j| + |c_i - c_j| = 1$.

Die große Familie von Stofl, bestehend aus $R \cdot C$ Mäusen, die praktischerweise von 1 bis $R \cdot C$ nummeriert sind, organisiert eine Reise nach Singapur. Da es sich um eine große Familie handelt, kann es sein, dass sich einige Mäuse untereinander nicht gut verstehen.

Namentlich sind zwei Funktionen gegeben: `likes(m)`, die die Menge der Mäuse zurückgibt, die Maus m mag ($1 \leq m \leq R \cdot C$) und `num(m)`, die die Größe von `likes(m)` zurückgibt (d. h. `num(m)` = |`likes(m)`|). Diese Beziehung ist garantiert symmetrisch: Wenn Maus a Maus b mag, dann mag Maus b auch Maus a, d. h. $a \in$ `likes(b)` $\iff b \in$ `likes(a)`. Alle anderen Paare mögen sich nicht.

Du willst Stofl helfen, dass der Ausflug ein Erfolg wird. Hilf Stofl, eine Sitzordnung zu finden, sodass die folgenden Bedingungen erfüllt sind:

- Jede Maus sitzt auf ihrem eigenen Sitzplatz.
- Alle Paare von Mäusen, die sich mögen, sitzen benachbart.
- Kein Paar von Mäusen, die sich nicht mögen, sitzt benachbart.

Wenn mehrere Sitzordnungen die obigen Kriterien erfüllen, kannst du eine beliebige von ihnen finden. Rufe die Funktion `assign(m, r, c)` auf, um die Maus m dem Sitzplatz (r, c) zuzuweisen. Wenn es keine solchen Anordnungen gibt, dann melde dies, indem du `impossible()` aufrufst. (dadurch werden alle vorherigen Aufrufe von `assign(m, r, c)` ignoriert).

Du kannst für die Analyse annehmen, dass `num`, `assign` und `impossible` in konstanter Laufzeit und mit konstantem Speicherbedarf laufen und dass `likes` eine Laufzeit und einen Speicherbedarf von $O($`num(m)`$)$ hat.

Teilaufgabe 1: Konkretes Beispiel (20 Punkte)

Sei $R = 3$ und $C = 3$ und die $N = 12$ Mäusepaare sehen wie folgt aus:

$$\{1,2\}, \{5,6\}, \{9,7\}, \{8,9\}, \{1,3\}, \{3,4\}, \{8,1\}, \{4,5\}, \{6,2\}, \{4,2\}, \{6,7\}, \{9,2\}$$

Können Mäuse unter Einhaltung der obigen Bedingungen im Flugzeug platziert werden?

Teilaufgabe 2: Ein Flugzeug mit einem einzigen Sitz in einer Reihe (30 Punkte)

In dieser Teilaufgabe nehmen wir an, dass $C = 1$ ist (d.h. es gibt nur einen Sitz in einer Reihe), aber es gibt keine weiteren Beschränkungen für R (d.h. das Flugzeug hat eine beliebige Anzahl Reihen).

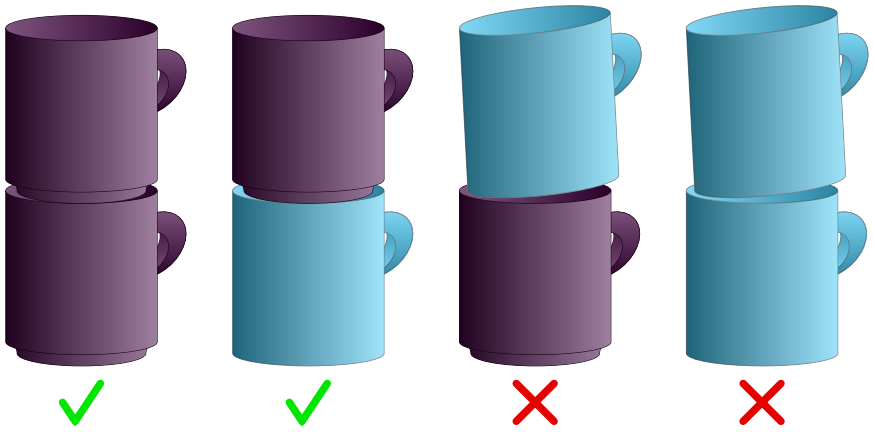
Teilaufgabe 3: Allgemeiner Fall (50 Punkte)

In dieser Teilaufgabe gibt es keine Einschränkungen für R oder C.

Tassen stapeln

Maus Stofl hat $2n$ Tassen im Schrank. Die Tassen stehen von links nach rechts in einer Reihe, und sind von 1 bis $2n$ nummeriert. Weil Maus Stofl in der Horizontalen der Platz ausgeht, möchte er die Tassen stapeln.

Einige Tassen haben einen komplett flachen Boden, während andere zur Mitte hin leicht gewölbt sind. Flache Tassen können nicht auf andere Tassen gestapelt werden, weil solch ein Stapel sehr instabil wäre. Gewölbte Tassen hingegen können gut auf andere Tassen gestapelt werden. In der Illustration unten sind die zwei Stapel links erlaubt, die zwei Stapel rechts nicht.



Maus Stofl will die Tassen in n Stapel der Höhe 2 stapeln. Um Tasse i auf Tasse j zu stellen, braucht Maus Stofl $|i - j|$ Sekunden. Finde die minimale Zeit, in der Stofl seine Tassen stapeln kann.

Für die Analyse kannst du annehmen, dass eine Funktion `type(i)` gegeben ist, welche den Typ der Tasse i ($1 \leq i \leq 2n$) in konstanter Laufzeit und mit konstantem Speicher zurückgibt.

Teilaufgabe 1: Löse ein Beispiel (10 Punkte)

Wie kann Stofl die folgende Anordnung von Tassen in minimaler Zeit stapeln? Du musst deine Antwort nicht beweisen.

Teilaufgabe 2: Perfekte Paarung (40 Punkte)

Genau die Hälfte der Tassen hat einen flachen Boden. Entwirf einen Algorithmus, der die minimale Zeit berechnet. Beweise die Korrektheit und gib die asymptotische Laufzeit und Speichernutzung an.

Teilaufgabe 3: Stofls Schrank (50 Punkte)

Stofls Schrank erfüllt nicht unbedingt die Einschränkung der vorherigen Teilaufgabe. Es ist allerdings garantiert, dass mindestens n Tassen einen gewölbten Boden haben. Entwickle einen Algorithmus, der im allgemeinen Fall die minimale Zeit berechnet. Beweise die Korrektheit und gib die asymptotische Laufzeit und Speichernutzung an.

Astrophysik

Dr. Mouse Binna ist eine bekannte experimentelle Astrophysikerin. Ihr allgemeiner Forschungsschwerpunkt ist die angewandte allgemeine Relativitätstheorie. Die Details ihrer neuesten Forschung können nur von den erfahrensten Forschern, die sich auf ihr Gebiet spezialisiert haben, verstanden werden. Ihre Forschung wurde aber wie folgt popularisiert: Das Ziel von Maus Binna ist es, Schwarze und Weisse Löcher zu finden.

Intuitiv ist ein Schwarzes Loch sehr *anziehend*: ausgehend von einem beliebigen Punkt in der Galaxie, existiert eine Geodäte, die im Schwarzen Loch endet. Es ist aber unmöglich das Schwarze Loch zu verlassen und einen beliebigen Punkt ausserhalb des Schwarzen Loches zu erreichen. [1]

Das Gegenteil eines Schwarzen Lochs ist ein Weisses Loch. Intuitiv ist ein Weisses Loch sehr abstossend: Es gibt eine Geodäte, die vom Weissen Loch ausgeht und zu einem beliebigen Punkt in der Galaxie führt, der sich nicht in einem Weissen Loch befindet. Es besteht aber keine Möglichkeit das Weisse Loch zu betreten, wenn es einmal verlassen wurde. [2]

Maus Binna betrachtet eine Menge von $n \geq 2$ Raum-Zeit-Orten innerhalb einer Galaxie, in der es höchstens ein Schwarzes Loch und höchstens ein Weisses Loch gibt.

Mit Hilfe einer komplizierten experimentell-theoretischen Methode (unter Einbeziehung von *Wissenschaft*) kann Maus Binna für zwei Orte a und b bestimmen, ob es einfach möglich ist, den Ort b zu erreichen, wenn man am Ort a startet.[3]

Wichtige Anmerkung: Maus Binnas Begriff der einfachen Erreichbarkeit basiert auf Astrophysik oberhalb jeder Ebene, die allgemein verstanden werden kann. Ein Ort ist einfach von sich aus erreichbar, aber das ist die einzige Regel, auf die man sich verlassen kann. Insbesondere, wenn man die einfache Erreichbarkeit für einige Paare von Orten kennt, sagt dies nichts über die einfache Erreichbarkeit für jedes andere Paar von unterschiedlichen Orten aus! (Insbesondere ist diese Beziehung nicht transitiv: es ist möglich, dass wir sowohl b von a als auch c von b aus einfach erreichen können, aber nicht c von a aus).

Inspiriert von ihrer Methode hat Maus Binna die Begriffe der *allgemeinen* Schwarzen Löcher und Weissen Löcher eingeführt (Trotzdem bezeichnet sie diese als Schwarze und Weisse Löcher):

Ein Schwarzes Loch ist eine Menge von Orten, die von überall einfach erreichbar sind, aber nicht einfach zu verlassen sind. D.h.:

- Falls x innerhalb des Schwarzen Lochs ist, dann gilt für *jeden* Ort y: x ist einfach erreichbar von y.
- Falls x innerhalb des Schwarzen Lochs ist, aber y *nicht* innerhalb des Schwarzen Lochs ist, dann ist y *nicht* einfach erreichbar von x.

Ein Weisses Loch ist eine Menge von Orten, von denen aus wir jeden anderen Ort einfach erreichen können. D.h.:

- Falls x innerhalb des Weissen Lochs ist, dann gilt für *jeden* Ort y: y ist einfach erreichbar von x.
- Falls x innerhalb des Weissen Lochs ist, aber y *nicht* innerhalb des Weissen Lochs ist, dann ist x *nicht* einfach erreichbar von y.

Da die Messung einfacher Erreichbarkeit teuer ist, möchte Mouse Binna davon so wenige wie möglich durchführen.

[1] Wie Maus Binna gerne betont ist dies der Fall, da nach dem Eintreten ins Schwarze Loch die Aussenwelt eher ein Punkt in der Zeit als ein Ort im Raum ist; das Verlassen des Schwarzen Lochs würde somit einer Rückwärtsreise in der Zeit entsprechen.

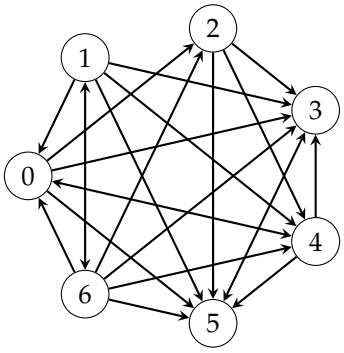
[2] Maus Binna sagt, dass dies der Fall ist, da das Innere des Weissen Lochs in der Vergangenheit liegt. Beispielsweise ist der Urknall ein Weisses Loch.

[3] Auch wenn Maus Binna es vorzieht zu sagen: Es gibt eine zulässige Geodäte mit hinreichend geringer Eigenzeit, die in a beginnt und in b endet, innerhalb einer Riemannschen Raumzeit-Mannigfaltigkeit mit einem geeigneten metrischen Tensor, obwohl für unsere Zwecke das Huhn sphärisch ist, wir die Entwicklung des Systems gemäss den Einstein-Feldgleichungen ignorieren, usw. [weitere Details im Interesse der Kürze weggelassen].

Teilaufgabe 1: Eine sehr kleine Galaxie (10 Punkte)

Um ihre Methode zu testen, hat Maus Binna bereits sämtliche Messungen einfacher Erreichbarkeit für eine kleine Menge von $n = 7$ Orten in einer kleinen Galaxie durchgeführt.

Es gibt einen Pfeil $a \to b$ genau dann, wenn Ort b einfach von Ort a aus erreichbar ist; und einen beidseitigen Pfeil $a \leftrightarrow b$ genau dann, wenn a von b, sowie b von a aus erreichbar ist.



Finde die Menge aller Orte, die sich im Schwarzen Loch befinden, sowie die Menge aller Orte, die sich im Weissen Loch befinden.

Teilaufgabe 2: Ein kleines Schwarzes Loch (15 Punkte)

In dieser Teilaufgabe gibt es kein Weisses Loch. Ausserdem ist garantiert, dass es ein Schwarzes Loch gibt und genau einer der angegebenen Orte innerhalb des Schwarzen Lochs liegt. D.h. dein Algorithmus muss nicht prüfen, ob es ein Schwarzes Loch gibt oder ob es wirklich die Grösse 1 hat. Dein Algorithmus kann annehmen, dass das Schwarze Loch existiert und dass es aus genau einem Ort besteht.

Du kannst Mause Binna fragen, die (unidirektionale) Erreichbarkeit für die Orte a nach b zu messen. D.h., Mouse Binna wird dir sagen, ob der Ort b vom Ort a aus einfach erreichbar ist. Du solltest mit möglichst wenigen Messungen den einzigen Ort innerhalb des Schwarzen Lochs finden.

Beachte, dass wir bei dieser Aufgabe Terme niedrigerer Ordnung ignorieren, aber die führende Konstante eine Rolle spielt. Zum Beispiel ist eine Lösung, die $4 \cdot n^2 + 3 \cdot n$ Messungen verwendet, äquivalent zu einer Lösung, die $4 \cdot n^2 + 2 \cdot n$ Messungen verwendet, aber eine Lösung, die $3 \cdot n + 100$ Messungen verwendet, ist strikt besser als eine Lösung, die $4 \cdot n$ Messungen verwendet. (Die beiden quadratischen Lösungen sind beide schlechter als jede Lösung mit linearer Zeit, unabhängig von der führenden Konstante).

Beachte, dass wir bei dieser Aufgabe (wie auch bei anderen Aufgaben) erwarten, dass du die Korrektheit und Effizienz deines Algorithmus zeigst, d. h. du sollst argumentieren, warum er immer die korrekten Mengen von Orten findet, und du sollst auch eine Obergrenze für die Anzahl der Messungen bestimmen, die er verwendet.

Andererseits musst du nicht beweisen, dass deine Lösung optimal ist, aber Lösungen, die weniger Messungen verwenden, erhalten mehr Punkte.

Teilaufgabe 3: Ein kleines Schwarzes Loch und Weisses Loch (20 Punkte)

In dieser Teilaufgabe wird garantiert, dass es sowohl ein Schwarzes als auch ein Weisses Loch gibt. Ausserdem befindet sich genau einer der gegebenen Orte innerhalb des Schwarzen Lochs und genau ein anderer Ort innerhalb des Weissen Lochs.

D.h. dein Algorithmus muss nicht prüfen, ob es ein Schwarzes Loch und ein Weisses Loch gibt und er muss auch nicht prüfen, ob sie wirklich die Grösse 1 haben. Er kann annehmen, dass beide Arten von Löchern existieren und dass jedes von ihnen aus genau einem Ort besteht.

Finde sowohl den Ort des Schwarzen Lochs als auch den Ort des Weissen Lochs, mit einer möglichst kleinen Gesamtzahl an Messungen.

Auch hier muss die Optimalität nicht bewiesen werden, versuch jedoch, den führenden Term möglichst klein zu machen.

Teilaufgabe 4: Ein grosses Schwarzes Loch (25 Punkte)

In dieser Teilaufgabe gibt es kein Weisses Loch. Ausserdem ist garantiert, dass es ein Schwarzes Loch gibt und mindestens einer der angegebenen Orte innerhalb des Schwarzen Lochs liegt. Finde die Menge aller Orte, die sich im Schwarzen Loch befinden, mit möglichst wenig Messungen. Auch hier muss die Optimalität nicht bewiesen werden, versuch jedoch, den führenden Term möglichst klein zu machen.

Teilaufgabe 5: Ein grosses Schwarzes Loch und Weisses Loch (30 Punkte)

In dieser Teilaufgabe ist garantiert, dass ein Schwarzes Loch und ein Weisses Loch existieren. Mindestens einer der angegebenen Orte ist innerhalb des Schwarzen Lochs und mindestens einer der angegebenen Orte ist innerhalb des Weissen Lochs. Das Schwarze Loch und das Weisse Loch haben keine gemeinsamen Orte. Finde die Menge aller Orte, die sich im Schwarzen Loch befinden und die Menge aller Orte die sich im Weissen Loch befinden. Benutze möglichst wenig Messungen. Wieder muss die Optimalität nicht bewiesen werden, versuch jedoch, den führenden Term möglichst klein zu machen.

Lösungen, die $4 \cdot n + O(1)$ Messungen verwenden, können bis zu 18 Punkte für diese Teilaufgabe erreichen.