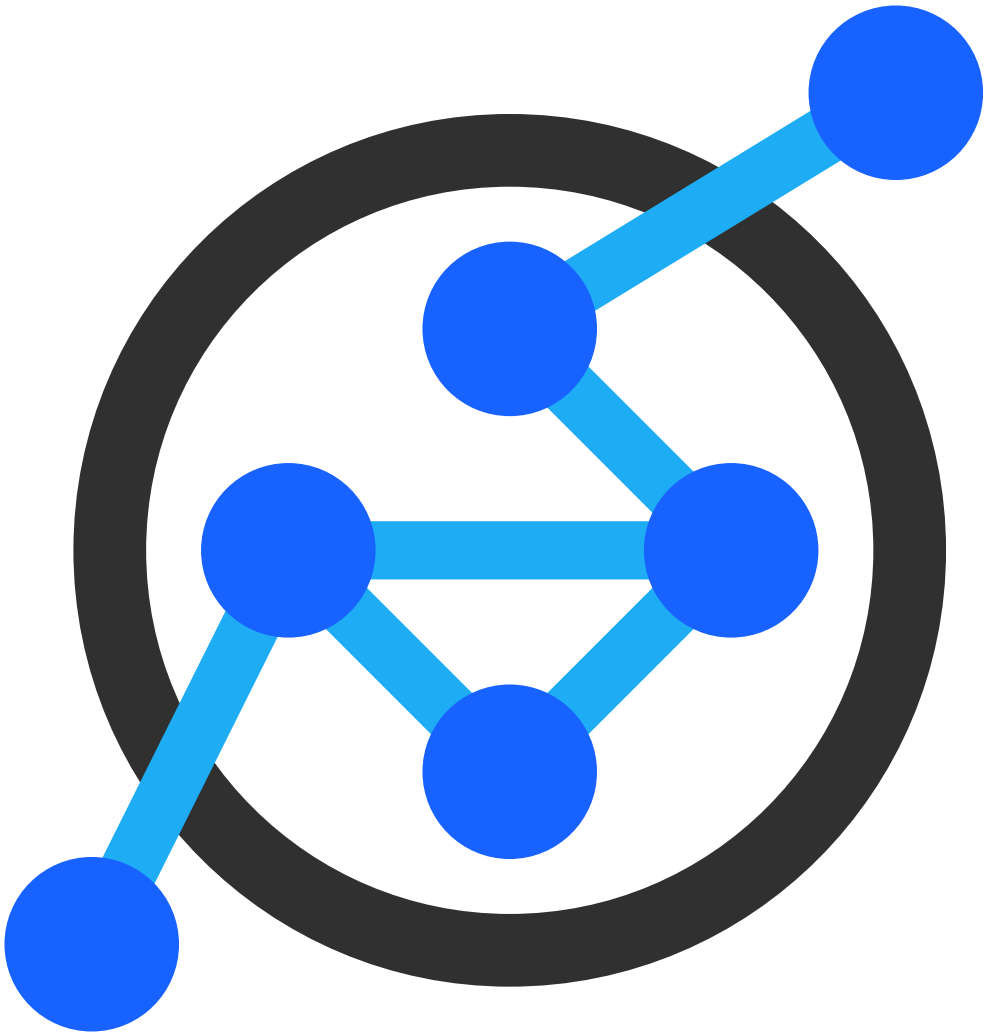
Zweite Runde Theorie

Aufgaben



Swiss Olympiad in Informatics

4. März 2023

Anweisungen

- Öffne die Prüfungs erst, wenn du dazu aufgefordert wirst. Die Prüfung beginnt für alle Teilnehmer gleichzeitig und dauert 5 Stunden.
- Mit Ausnahme von Uhren (keine Smartwatches) sind keine elektronischen Geräte auf den Tischen erlaubt. Schalte dein Mobiltelefon aus.
- Für alle Fragen zu den *Aufgaben* verwende bitte den beiliegenden Fragebogen.
- Beginne jede Aufgabe auf einem neuen Blatt und schreibe deinen Namen auf alle Blätter. Nummeriere deine Seiten und sortiere sie vor der Abgabe.
- Verwende keinen Bleistift und schreibe nicht mit rot.
- Schreibe lesbar.

Bewertung

Deine Lösung wird anhand ihrer Korrektheit und der asymptotische 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!

Gulasch

Als Vorbereitung auf die IOI in Ungarn will Maus Stofl einen grossen Topf Gulasch (eine traditionelle ungarische Suppe) für alle seine Freunde kochen. Leider hat er zu schnell mit dem kochen begonnen und vergessen, die Zutaten aus dem Keller zu holen. Jetzt, wo er angefangen hat, muss er das Gulasch ständig umrühren und kann den Topf nicht unbeaufsichtigt lassen. Natürlich haben sich seine Freunde angeboten ihm zu helfen, die Zutaten aus dem Keller zu holen.

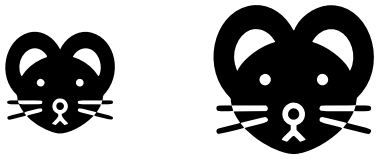
Es werden N Zutaten für das Gulasch benötigt und Stofl hat M Freunde, die ihm helfen, die Zutaten aus dem Keller zu holen. Die Zutaten in Stofls Keller sind in Regalen angeordnet. Jede Zutat i befindet sich in einer bestimmten Höhe h_i und jeder von Stofls Freunden j hat eine bestimmte maximale Höhe m_j, die er/sie erreichen kann.

Da alle sehr hungrig sind, wollen sie das Gulasch so schnell wie möglich auftischen. Das Gulasch ist fertig, wenn alle benötigten Zutaten in den Topf gegeben wurden. Die Reihenfolge, in der die Zutaten hinzugefügt werden, spielt keine Rolle. Jeder von Stofls Freunden kann genau eine Zutat auf einmal mitnehmen. Es ist nicht möglich, dass sie mehrere Zutaten auf einmal aus dem Keller holen. Praktischerweise gehen alle Stofls Freunde mit der gleichen Geschwindigkeit. Sie brauchen 1 Minute, um in den Keller zu gehen, und 2 Minuten, um mit dem, was sie tragen, wieder zurückzukommen.

Kannst du ihnen helfe, mit Hilfe einer Zutatenliste und einer Liste von Stofls Freunden herauszufinden, ob sie das Gulasch zubereiten können, und berechnen, wie lange sie dafür brauchen?

Ein Beispiel dafür, wie das aussehen kann, ist unten zu sehen.

i	Zutat	Höhe
0	Salz	2
1	Tomate	4
2	Kartoffel	3



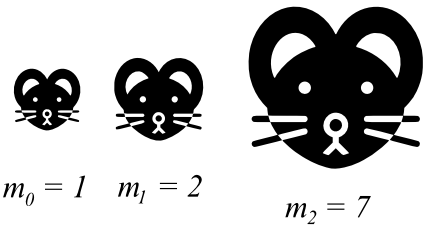
$m_0 = 1$ $m_1 = 4$

In diesem Beispiel hat Stofl zwei Freunde und es werden drei Zutaten für das Gulasch benötigt. Leider ist Freund 0 nicht gross genug, um irgendeine der Zutaten zu erreichen. Dadurch muss Freund 1 dreimal in den Keller gehen, um die drei Zutaten zu holen, was $3 \cdot (1 + 2) = 9$ Minuten dauert.

Teilaufgabe 1: Löse ein Beispiel (10 Punkte)

Löse die Aufgabe für das folgende Gulaschrezept mit Hilfe der drei abgebildeten Freunde von Stofl.

i	Zutat	Höhe
0	Olivenöl	1
1	Steak	4
2	Zwiebel	7
3	Knoblauch	1
4	Tomate	3
6	Paprika	5
7	Petersilie	2



Schreibe genau auf, wann welche Mäuse in den Keller gehen und was sie hinauftragen. Du brauchst deine Antwort in dieser Teilaufgabe nicht zu begründen.

Teilaufgabe 2: Prüfe, ob es möglich ist (10 Punkte)

Entwirf einen Algorithmus, der ermittelt, ob es möglich ist, das Gulasch zu kochen. Dieser Algorithmus sollte "*Yes*" ausgeben, wenn es für Stofls Freunde möglich ist, alle Zutaten aus dem Keller zu holen und "*No*" andernfalls.

Teilaufgabe 3: Löse den allgemeinen Fall (80 Punkte)

Entwirf einen Algorithmus, der Stofl hilft, die minimale Zeit zu bestimmen, die seine Freunde brauchen, um alle Zutaten aus dem Keller zu holen. Du kannst $N > M$ annehmen, um die Effizienz deines Algorithmus zu beurteilen.

Joggingkarte

Mausarnen ist ein ruhiges und abgelegenes Dorf mit L Häusern, die von riesigen Gärten umgeben sind, jedes von den Häusern ist mit einer einzigen Einfahrt zu einem Ort verbunden ist. *Strassen* in Mausarnen, jede davon genau 1 Mausmeter lang, verbinden zwei *Orte*, wobei ein Ort entweder eines der L *Häuser* oder eine der Y *Kreuzungen* in Mausarnen ist. (Ein *Auffahrt* ist also nur eine spezielle Art von *Strasse*.) Die Einwohner von Mausarnen machen sich nicht viel aus der Strasseninfrastruktur, daher gibt es nicht mehr Strassen als nötig. Wenn du einen Mausarner danach fragst, wird er nur mit den Schultern zucken und sagen: *"Warum sollten wir mehr haben? Egal, wo ich in Mausarnen bin, ich kann an jeden anderen Ort gelangen, also sollte das reichen."* Deshalb gibt es in Mausarnen genau $L + Y - 1$ Strassen.

Mausarnen ist der Ort, an dem Maus Binna wohnt und an dem das berühmte MOI[1]-Lager stattfindet. Maus Binna und Maus Stofl konnten sich dieses Jahr für das Lager qualifizieren. Sie sind sehr gute Freunde, könnten aber in einem Punkt nicht unterschiedlicher sein. Während Maus Binna täglich eine Runde um das Dorf joggt, findet Maus Stofl nur selten die Motivation, sich körperlich zu betätigen.

Die Leiter der MOI versorgen ihre Teilnehmer mit Unmengen von Snacks, und obwohl Stofl diese sehr geniesst, kommt ihm schliesslich der Gedanke, dass er seiner Gesundheit zuliebe vielleicht etwas Sport treiben sollte. Er will sich ein Beispiel an Maus Binna nehmen und ebenfalls mit dem Joggen beginnen. Um die perfekte Joggingstrecke zu finden, braucht Stofl eine Karte von Mausarnen. Genauer gesagt, will er wissen, welche Orte miteinander verbunden sind.

Zum Glück kommt Binna gerade vom Joggen zurück. Sie hat zwar keine Karte von Mausarnen, aber dank ihrer hochmodernen Armbanduhr kann sie die Entfernung zwischen zwei beliebigen der $L + Y$ Orte nachgucken. Aufgrund des kryptischen, proprietären Formats ist es leider nicht möglich, direkt an der Uhr zu erkennen, ob es sich bei dem Ort um ein Haus oder eine Kreuzung handelt. Allerdings kennen unsere Mäuse die Anzahl der Häuser L und die Anzahl der Kreuzungen Y. Maus Binna will so schnell wie möglich weiterjoggen, also bittet sie Stofl, seine Fragen auf ein Minimum zu beschränken.

Du kannst davon ausgehen, dass die Orte von $0, 1 \ldots L + Y - 1$ nummeriert sind. Man kann eine Frage der Form $d(a, b)$ stellen, worauf hin man die Anzahl der Strassen bekommt, die man beim Joggen vom Ort a zum Ort b durchläuft.

Teilaufgabe 1: Löse ein Beispiel (15 Punkte)

Konstruiere anhand der folgenden Fragen und Antworten alle möglichen Karten mit den Orten $0, 1, 2, 3, 4, 5$:

- $d(4, 2) = 1$
- $d(3, 3) = 0$
- $d(3, 0) = 2$
- $d(3, 5) = 2$
- $d(0, 2) = 3$

Begründe kurz, warum andere Karten nicht möglich sind.

[1] Mausolympiade in Informatik

Teilaufgabe 2: Lösen Sie den allgemeinen Fall (65 Punkte)

Kannst du Stofl helfen, die Karte von Mausarnen mit der geringsten Anzahl von Fragen asymptotisch im ungünstigsten Fall zu konstruieren?

Deine Priorität ist es, zuerst die Anzahl der Fragen, dann die Laufzeit und zuletzt den Speicher zu optimieren. Und das alles asymptotisch, d.h. in big-$O()$ und im schlimmsten Fall.

Mit "Worst Case" meinen wir, dass die Uhr ihre Ortsbezeichnungen nicht auswählen muss, bevor Sie die erste Frage stellen. Sie kann ihre "Lösungskarte" im Hintergrund anpassen, solange sie mit den bereits gegebenen Antworten konsistent bleibt[2].

Um die volle Punktzahl zu erhalten, musst du die Karte von Mausarnen in $O(Y \cdot L)$-Abfragen bestimmen. Wenn du $O((L+Y)^2)$ Fragen stellst, kannst du bis zu 10 Punkte erhalten. Beachte, dass du mehr als 10 Punkte bekommen kannst, wenn deine Anzahl der Fragen zwischen $O((L+Y)^2)$ und $O(Y \cdot L)$ liegt.

Wenn du das Problem für den Spezialfall von Karten löst, bei denen $L = 2$ ist, d.h. alle Orte auf einer Linie liegen, kannst du bis zu 10 Punkte erhalten.

Hinweis: In dieser Teilaufgabe kannst du beide Arten von Teilpunkten erhalten. Wenn du den Spezialfall effizient und den allgemeinen Fall in $O((L+Y)^2)$ Fragen richtig löst, kannst du 20 Punkte bekommen.

Teilaufgabe 3: Zeige eine untere Schranke (20 Punkte)

Maus Binna ist etwas verärgert, dass sie so lange warten musste und behauptet, dass Maus Stofl seine Frage ineffizient gestellt hat. Um sich zu verteidigen, nachdem Binna von ihrem nächsten Lauf zurückkommt, will Stofl Binna ein Beispiel für eine Karte zeigen, bei der man eine bestimmte Anzahl von Fragen stellen muss, um sicher zu sein, dass die Karte korrekt ist.

Angenommen, die Anzahl der Orte N ist gegeben. Kannst du ihm helfen, eine Karte zu konstruieren, bei der jeder Algorithmus im schlimmsten Fall mindestens $\frac{N^2}{2023}$ Fragen braucht? Du kannst die Anzahl der Häuser L und der Kreuzungen Y frei wählen, solange sie $N = L + Y$ erfüllen. Du kannst davon ausgehen, dass du Pech mit den Bezeichnungen hast, d.h. dass die Uhr die Bezeichnungen der noch nicht abgefragten Orte ändert.

Um die volle Punktzahl zu erhalten, musst du *begründen*, warum deine gewählte Karte nicht in weniger als $\frac{N^2}{2023}$ Fragen konstruiert werden kann.

[2] Damit der Hersteller die verbesserte Uhr im nächsten Jahr verkaufen kann

Programmiermissgeschick

Maus Binna nimmt an einem Programmierwettbewerb teil. Sie war gerade dabei, ihre 100-Punkte-Lösung für den letzten Task des Wettbewerbs abzugeben. Allerdings wurde sie beim Abgeben unglücklicherweise in ein kleines *Tabulatorenvervollständignungsmissgeschick* involviert.[1]

Dadurch wurden die Zeilen ihres Programms durchgemischt, sodass sie nun in zufälliger Reihenfolge dastehen. Da dies ein Programmierwettbewerb ist, unterliegt ihr Code leider nicht einer Versionskontrolle, daher muss sie nun den Inhalt der Datei manuell sortieren.

Maus Binnas fortgeschrittener Texteditor `emous` hat eine Reihe von speziellen Befehlen, um die Hackerelite, zu der auch Maus Binna gehört, bei der Erfüllung solcher Aufgaben zu unterstützen. Ein besonders effizienter Befehl (`M-x rot3`) erlaubt Maus Binna, drei unterschiedliche Zeilen i, j and k auszuwählen. Dieser Befehl ändert dann gleichzeitig alle drei Zeilen so, dass deren Inhalt *rotiert* wird: Zeile j enthält dann den Inhalt, den Zeile i vorher hatte, Zeile k jenen von Zeile j und Zeile i den vorherigen Inhalt von Zeile k.

Als eine leidenschaftliche Verfechterin vom DRY (Don't Repeat Yourself)-Prinzip hat Maus Binna selbstverständlich keine Codezeile wiederholt. Daher weiss sie bei jeder Zeile Code im gemischten Text ganz genau, wohin sie eigentlich gehört.

Rasch sortiert Maus Binna ihren Code. Dabei benutzt sie nur `rot3` Befehle, wie sie oben beschrieben sind. Natürlich tut sie das so effizient wie nur möglich und benutzt daher die kleinstmögliche Anzahl `rot3` Operationen.

Kannst du diese Art Aufgabe im Allgemeinen lösen?

`emous` fängt bei 0 an, um die Zeilen zu nummerieren. Gegeben ist eine Liste von N unterschiedlichen ganzen Zahlen zwischen 0 und $N-1$. Die i-te Zahl p_i zeigt an, dass die Zeile i des gemischten Quellcodes die Zeile p_i des originalen Codes enthält. Finde eine kürzestmögliche Abfolge von `rot3` Operationen, die den originalen Quellcode wiederherstellen, oder entscheide, dass es nicht möglich ist.

Betrachte als Beispiel folgende Eingabe mit $N = 6$:

- $p_0 = 3$
- $p_1 = 0$
- $p_2 = 4$
- $p_3 = 1$
- $p_4 = 5$
- $p_5 = 2$

In diesem Fall können wir die die Zeilen mit zwei `rot3` Operationen sortieren:

1. `rot3`$(0, 3, 1)$. Diese Operation verschiebt Zeile 0 zu Zeile 3, Zeile 3 zu Zeile 1 und Zeile 1 zu Zeile 0. Die folgende Eingabe beschreibt die neue Situation:
   - $p_0 = 0$
   - $p_1 = 1$
   - $p_2 = 4$
   - $p_3 = 3$
   - $p_4 = 5$
   - $p_5 = 2$
2. `rot3`$(2, 4, 5)$. Diese Operation verschiebt Zeile 2 zu Zeile 4, Zeile 4 zu Zeile 5 und Zeile 5 zu Zeile 2. Die folgende Eingabe beschreibt die neue Situation:
   - $p_0 = 0$
   - $p_1 = 1$
   - $p_2 = 2$
   - $p_3 = 3$
   - $p_4 = 4$
   - $p_5 = 5$

   Jetzt enthält jede Zeile der vermischten Datei die entsprechende Zeile der originalen Datei, wir haben also erfolgreich alles sortiert. Desweiteren gibt es keine Möglichket, das mit weniger Operationen zu erreichen.

[1] Sie hat aus Versehen das Bash-Skript `./shuffle.sh e.py` statt `./submit.sh e.py` ausgeführt.

Teilaufgabe 1: Kleines Beispiel (10 Punkte)

Gib explizit eine beliebige kürzeste Abfolge an `rot3` Operationen, welche jene Datei sortiert, die durch folgende Eingabe mit $N = 11$ beschrieben wird:

- $p_0 = 1$
- $p_1 = 0$
- $p_2 = 3$
- $p_3 = 4$
- $p_4 = 5$
- $p_5 = 2$
- $p_6 = 7$
- $p_7 = 8$
- $p_8 = 9$
- $p_9 = 10$
- $p_{10} = 6$

Im Gegensatz zu den anderen Teilaufgaben musst du hier keine Begründung der Korrektheit angeben, wenn deine Lösung korrekt ist. (Allerdings kann eine Begründung der einzelnen Schritte hilfreich sein, um Teilpunkte zu erhalten, falls das Gesamtresultat nicht korrekt ist.)

Teilaufgabe 2: Zyklische Verschiebung von 2023 Zeilen (10 Punkte)

Löse folgenden Testfall mit $N = 2023$ auf optimale Weise oder beweise, dass es nicht möglich ist:

- $p_0 = 1$
- $p_1 = 2$
- $p_2 = 3$
- $p_3 = 4$
- $p_4 = 5$
- ...
- ...
- $p_{2020} = 2021$
- $p_{2021} = 2022$
- $p_{2022} = 0$

Das heisst, für $0 \leq i < 2022$ haben wir $p_i = i + 1$ und $p_{2022} = 0$.

Du musst nicht alle Operationen explizit angeben, aber du solltest klarstellen, welche das sind. Beachte, dass du jetzt begründen musst, warum deine Lösung korrekt ist.

Teilaufgabe 3: Zyklische Verschiebung von 2024 Zeilen (10 Punkte)

Löse folgenden Testfall mit $N = 2024$ auf optimale Weise oder beweise, dass es nicht möglich ist:

- $p_0 = 1$
- $p_1 = 2$
- $p_2 = 3$
- $p_3 = 4$
- $p_4 = 5$
- ...
- ...
- $p_{2020} = 2021$
- $p_{2021} = 2022$
- $p_{2022} = 2023$
- $p_{2023} = 0$

Das heisst, für $0 \leq i < 2023$ haben wir $p_i = i + 1$ und $p_{2023} = 0$.

Du musst nicht alle Operationen explizit angeben, aber du solltest klarstellen, welche das sind. Beachte, dass du jetzt begründen musst, warum deine Lösung korrekt ist.

Teilaufgabe 4: Umkehrung (30 Punkte)

In dieser Teilaufgabe ist $N \geq 1$ beliebig und die Reihenfolge der Zeilen ist genau umgekehrt. Das bedeutet, wir haben immer $p_i = N - 1 - i$.

Entwirf einen Algorithmus, der in diesem besonderen Fall immer die richtige Antwort berechnet. (Er soll entweder eine kürzestmögliche Liste an `rot3` Operationen angeben oder ausgeben, dass

es keine Lösung gibt.)

Teilaufgabe 5: Allgemeiner Fall (40 Punkte)

In dieser Teilaufgabe gibt es keine weiteren Einschränkungen. Das heisst, die Zeilen können beliebig angeordnet sein. Entwerfe einen Algorithmus, der immer die richtige Antwort berechnet. (Er soll entweder eine kürzestmögliche Liste an `rot3` Operationen angeben oder ausgeben, dass es keine Lösung gibt.)

Du kannst bis zu 20 Punkte erhalten, wenn du eine Lösung findest, die nicht optimal ist, aber maximal um einen konstanten Faktor von dieser abweicht (du musst das in diesem Fall immer noch beweisen). Algorithmen, die mindestens dreimal die optimale Anzahl Operationen benötigen, erhalten maximal 10 Punkte.

Für die Beschreibung eines guten Algorithmus und seiner Korrektheit sind jeweils gleich viele Punkte vorgesehen.

Ungerade Kanäle

Ein Forschungsteam unter der Leitung von Maus Binna hat die Ruinen der ehemaligen Hauptstadt des Volkes der Soianer entdeckt, eines uralten Stammes, der für seine technologischen Fortschritte und seine Fähigkeit, kryptische Rätsel zu lösen, bekannt ist. Die Ruinen sind von einem Kanalsystem durchzogen, das wahrscheinlich einst dem Warentransport diente. Es besteht aus $2 \cdot N$ Stätten, die durch M Kanäle verbunden sind. Maus Binna hatte schon lange vermutet, dass die Soianer bestimmten Zahlen bestimmte Bedeutungen zuschrieben. Insbesondere Zahlen, die nach der Division durch 4 einen Rest von 1 haben, gelten als Glückszahlen. In der Tat ist die Anzahl der Kanäle, die mit einer Stätte verbunden sind, immer ein Vielfaches von 4 plus 1, so als hätten die Soianer ihre Glückszahlen sorgfältig in ihre Architektur eingearbeitet.

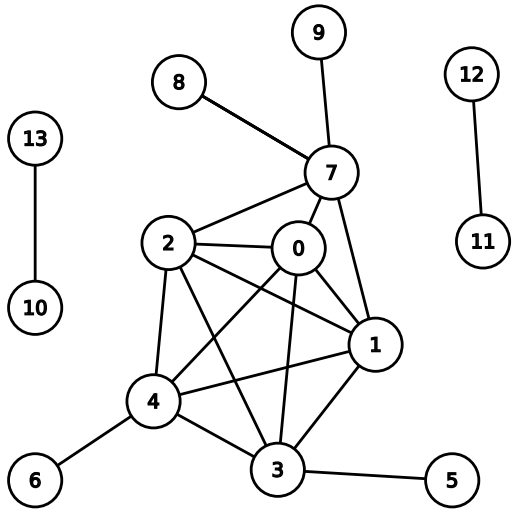
Leider hat der Mouseity Council beschlossen, die Ruinen als Touristenattraktion zugänglich zu machen, trotz des erbitterten Widerstands der Wissenschaftler. Maus Binna arbeitet nun an einem Konzept für Touristenbesuche, das einen respektvollen Umgang mit den soianischen Ausgrabungsstätten gewährleisten soll. In erster Linie möchte sie, dass die Touristen nur auf den Kanälen fahren und nicht durch die Ruinen laufen. Damit das funktioniert, muss jeder Kanal eine bestimmte Richtung haben, in die die Boote fahren müssen, um Kollisionen zu vermeiden. Ausserdem ist es ihr wichtig, den Respekt vor der soianischen Kultur zu wahren und zumindest einige ihrer Glaubensvorstellungen in das Konzept einfliessen zu lassen. Aus einer soianischen Legende erfahren wir, dass eine Reise nicht immer Glück bringen muss, sondern in vielerlei Hinsicht ausgeglichen sein sollte. Deshalb will Maus Binna, dass die Anzahl der wegführenden Kanäle für N Standorte ungerade und für die anderen N Standorte gerade ist.

Damit das Konzept eine Chance hat, vom Stadtrat genehmigt zu werden, braucht sie deine Hilfe, um die Richtungen der Kanäle zu bestimmen.

Teilaufgabe 1: Löse ein Beispiel (10 Punkte)

Maus Binna hat einige Baupläne gefunden, die auf alternative Grundrisse für die ehemalige Hauptstadt hindeuten. In diesen Zeichnungen sind die Orte durch Kreise und die Kanäle durch Linien dargestellt.

Sie hat eine kleinere Zeichnung genommen und versucht, zuerst eine Lösung für diese Zeichnung zu finden. Kannst du die Kanäle lenken (indem du die Linien in Pfeile umwandelst), so dass genau die Hälfte der Grundstücke eine ungerade Anzahl von Kanälen hat, die wegführen?



Da Binna dir eine Kopie gegeben hat, darfst du die Richtungen in diesem Bild direkt einzeichnen. Denke aber daran, es am Ende mit abzugeben.

Teilaufgabe 2: Ungerade Anzahl von Stätten? (5 Punkte)

Während du das Beispiel gelöst hast, hat sich Maus Binna einige weitere Layouts angeschaut, die S Stätten und M Kanäle haben. Sie stellte fest, dass die Anzahl der Grundstücke immer gerade ist, wenn die Anzahl der Kanäle, die ein Grundstück verlassen, ein Vielfaches von 4 plus 1 ist. Kannst du Maus Binna erklären, warum dies immer der Fall ist?

Teilaufgabe 3: Nur ungerade ist unmöglich (10 Punkte)

In einem sojanischen Volksmärchen gelten Vielfache von 4 plus 3 ebenfalls als Glücksbringer, aber die Mischung mit Vielfachen von 4 plus 1 bringt Unglück. Maus Binna will den Wert dieses Märchens zeigen.

Sie will ein Layout eines Kanalsystems konstruieren, das die folgenden zwei Eigenschaften erfüllt:

a) die Anzahl der Kanäle auf jeder Seite ist ein Vielfaches von 4 plus 1 oder ein Vielfaches von 4 plus 3, mit anderen Worten einfach ungerade.

b) ist es unmöglich, die Kanäle so anzuordnen, dass von genau N Standorten eine ungerade Anzahl von Kanälen wegführt (wie in der Soian-Legende).

Kannst du ihr helfen, ein solches Beispiel zu finden?

Teilaufgabe 4: Finde einen Algorithmus (75 Punkte)

Hilf Binna dabei, einen Algorithmus zu finden, der die Kanäle leitet, wenn ein Grundriss der Ruinen vorliegt.

Bei dieser Teilaufgabe geht es vor allem um die Korrektheit deines Algorithmus.

Deine Lösung kann bis zu 55 Punkte erreichen, wenn sie in polynomieller Zeit für ein beliebiges Polynom läuft. Mit polynomiell meinen wir etwas wie $O(n^{100})$ im Gegensatz zu $O(2^n)$.

Ausserdem kannst du bis zu 35 Punkte erreichen, wenn du eine besondere Art von Layouts

löst: Layouts mit $2 \cdot N - 1$ Kanälen, bei denen jeder Ort von jedem anderen Ort aus über Kanäle erreicht werden kann, bevor man ihn ansteuert.

Hinweis: Wenn du beide Varianten gesondert beweist, die polynomielle Schranke und das besondere Layout, dann erhältst du nur Punkte für den Teil, indem du mehr Punkte hast, und nicht die Summe aus beiden.