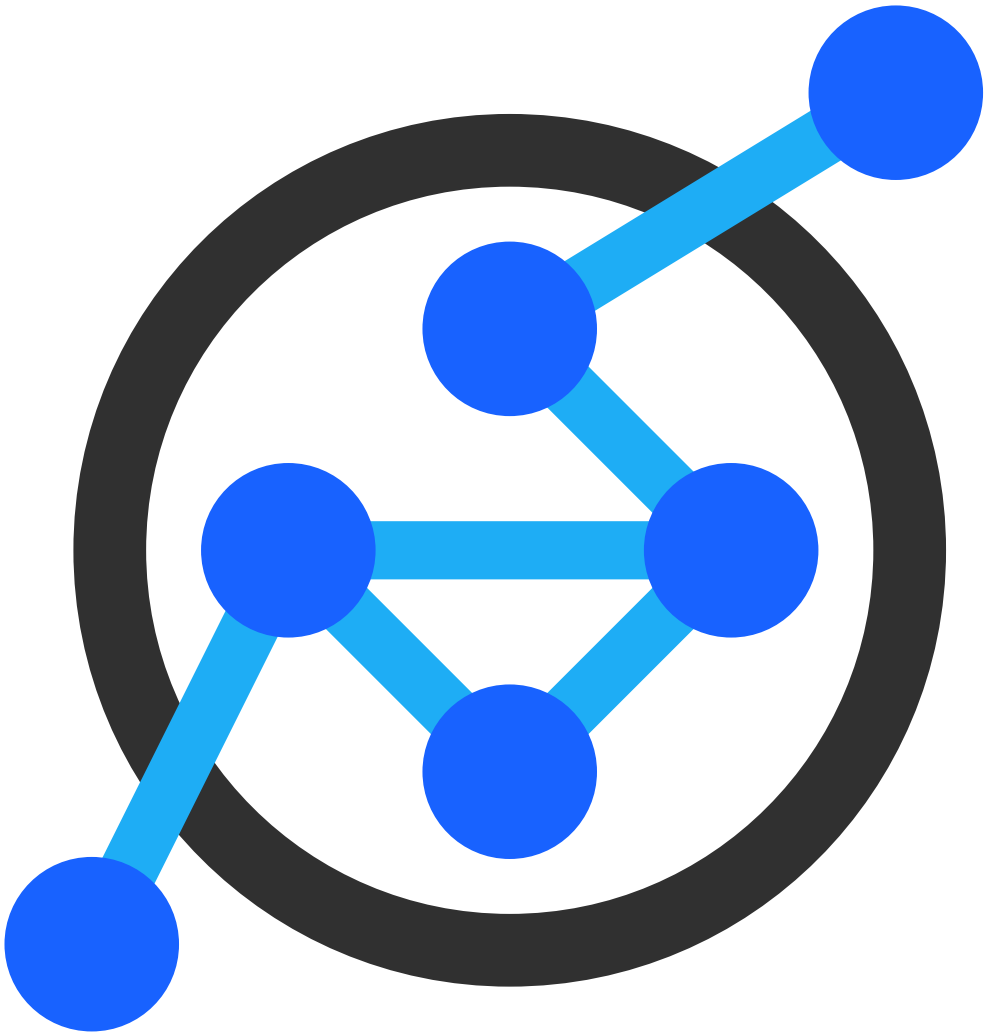
Deuxième Tour Théorique

Tâches



Swiss Olympiad in Informatics

4 mars 2023

Indications

— Ouvre le livret seulement quand tu y es invité. L'épreuve commence en même temps pour tous les participants et dure cinq heures.
— Tous les appareils électroniques sont interdits, à l'exception des montres (sauf montres intelligentes). Éteins ton téléphone portable.
— Pour toutes questions *relative aux problèmes*, merci d'utiliser le formulaire fourni.
— Commence chaque exercice sur une nouvelle feuille et écris ton nom sur chaque feuille. Numérote les feuilles et trie les avant la reddition.
— N'utilise pas de crayon et n'écris pas en rouge.
— Écris lisiblement.

Évaluation

Les solutions sont évaluées selon des critères similaires à ceux du premier tour théorique. Les deux critères les plus importants sont la correction et le temps d'exécution asymptotique. La qualité de la description et les arguments prouvent la correction sont également pris en compte.

Tu peux toujours te référer à du contenu du Wiki SOI ou de 2H. Tu ne dois pas expliquer pourquoi, par exemple, l'algorithme de Dijkstra fonctionne, mais tu dois expliquer pourquoi et comment il peut être appliqué. Dans le cas de Dijkstra, tu devrais expliquer clairement sur quel graphe tu l'utilises et noter que les poids des arêtes de celui-ci ne sont pas négatives.

l'algorithme doit être décrit en suffisamment de détail pour qu'il soit aisé de convertir la description en un programme. Écris également quelles structures de données tu utiliserais. Habituellement, le mieux est d'écrire un bref pseudocode.

Si un algorithme t'est demandé, tu devrais utiliser cette structure comme ligne directrice :

1. Décris l'idée derrière un algorithme qui résout le problème.
2. Donne du pseudocode ou explique comment on pourrait implémenter l'algorithme.
3. Prouve par des arguments la correction de l'approche utilisée.
4. Indique le temps d'exécution asymptotique ainsi que l'utilisation de mémoire (en les justifiant).

Si une partie de ta solution est utile pour plusieurs sous-tâches, il suffit de l'écrire une seule fois et de t'y référer à partir de là. Note, cependant, qu'un algorithme peut résoudre plusieurs sous-tâches mais pas nécessairement de façon optimale.

Bonne chance !

Goulasch

En guise de préparation à l'IOI en Hongrie, la souris Stofl veut préparer une grande marmite de goulasch (une soupe traditionnelle hongroise) pour tous ses amis. Malheureusement, il a commencé trop vite et a oublié de prendre les ingrédients à la cave. Maintenant qu'il a commencé, il doit continuer à remuer le goulasch et ne peut pas laisser la marmite sans surveillance. Bien sûr, ses amis se sont portés volontaires pour l'aider à aller chercher les ingrédients à la cave.

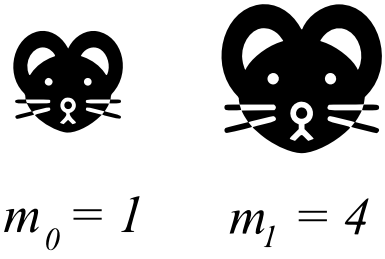
Il y a N ingrédients nécessaires pour le goulasch et Stofl a M amis pour l'aider à aller chercher les ingrédients à la cave. Les ingrédients qui se trouvent dans la cave de Stofl sont rangés sur des étagères. Chaque ingrédient i se trouve à une hauteur spécifique h_i et chacun des amis de Stofl j a une certaine hauteur maximale m_j qu'il peut atteindre.

Comme tout le monde a très faim, ils veulent finir le goulasch aussi vite que possible. Le goulasch sera terminé lorsque tous les ingrédients nécessaires auront été ajoutés à la marmite. Notez que l'ordre dans lequel les ingrédients sont ajoutés n'a pas d'importance. Chacun des amis de Stofl peut transporter exactement un ingrédient à la fois. Il ne leur est pas possible d'apporter plusieurs éléments en un seul voyage au sous-sol. Par chance, tous les amis de Stofl marchent à la même vitesse et il leur faut 1 minute pour aller au sous-sol et 2 minutes pour revenir avec ce qu'ils transportent.

Étant donné une liste d'ingrédients et des amis de Stofl, pouvez-vous les aider à déterminer s'ils peuvent faire le goulasch et calculer le temps qu'il leur faudra pour le terminer ?

Vous trouverez ci-dessous un exemple de ce que cela peut donner.

i	Ingrédient	Hauteur
0	Sel	2
1	Tomate	4
2	Patate	3

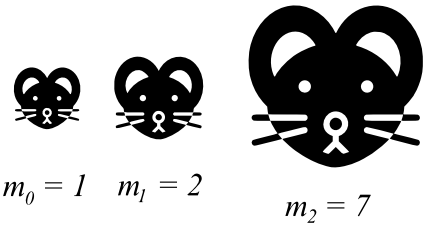


Dans cet exemple, Stofl a deux amis et il y a trois ingrédients nécessaires pour le goulasch. Malheureusement, l'ami 0 ne peut atteindre aucun des ingrédients. L'ami 1 doit donc se rendre trois fois à la cave pour récupérer les trois ingrédients, ce qui lui prend $3 \cdot (1 + 2) = 9$ minutes.

Sous-tâche 1: Résoudre un exemple (10 points)

Résolvez le problème de la recette de goulasch suivante avec l'aide des trois amis de Stofl représentés.

i	Ingrédient	Hauteur
0	Huile d'olive	1
1	Steak	4
2	Oignon	7
3	Ail	1
4	Tomate	3
6	Paprika	5
7	Persil	2



Notez exactement quand les souris vont au sous-sol et ce qu'elles en rapportent. Vous n'avez pas besoin d'expliquer votre réponse dans cette sous-tâche.

Sous-tâche 2: Déterminer si c'est possible (10 points)

Concevez un algorithme qui détermine s'il est possible de cuisiner le goulasch. Cet algorithme doit produire le résultat "*Yes*" s'il est possible pour les amis de Stofl de récupérer tous les ingrédients au sous-sol et "*No*" dans le cas contraire.

Sous-tâche 3: Résoudre le cas général (80 points)

Concevez un algorithme pour aider Stofl à déterminer le temps minimum dont ses amis ont besoin pour récupérer tous les ingrédients dans la cave. Vous pouvez supposer que $N > M$ pour juger de l'efficacité de votre algorithme.

Carte de jogging

Mousarnen est un village calme et isolé, composé de L maisons entourés de grands jardins, ayant chacune **exactement une** allée qui les connecte à un lieu. Les *routes* de Mousarnen, chacune mesurant exactement un sourismètre de long, relient deux *lieux*, où un lieu est soit une des L *maisons* ou une des Y *intersections* de Mousarnen. (Par conséquent, une *allée* est simplement un type de *route*.) Les habitants de Mousarnen préfèrent la sobriété dans leur infrastructure routière, il n'y a donc pas plus de routes que nécessaire. Si vous demandez pourquoi à un Mousarnien, il haussera les épaules et vous répondra : *"Pourquoi en aurait-on besoin de plus ? Peu importe où je suis dans Mousarnen, je peux me rendre à n'importe quel autre endroit, c'est largement suffisant."* C'est pourquoi il y a exactement $L + Y - 1$ routes à Mousarnen.

C'est à Mousarnen que vit la souris Binna et que se déroule le fameux camp de l'OIS[1]. La souris Binna et la souris Stofl ont réussi à se qualifier pour le camp cette année. Ce sont de très bons amis, mais qui ne pourraient pas être plus différents sur un point particulier. Alors que la souris Binna fait un jogging dans le vilage chaque jour, la souris Stofl trouve rarement la motivation de faire une activité sportive.

Les leaders de l'OIS fournissent des tonnes de snacks à leurs participants, et bien que Stofl les aime énormément, il vient finalement de se rendre compte que pour sa santé, il devrait peut-être faire au moins un peu d'exercice physique. Il veut prendre exemple sur la souris Binna et souhaite se mettre à faire du jogging lui aussi. Pour trouver le meilleur trajet, il lui faut une carte de Mousarnen. Plus précisément, il veut savoir quels lieux sont connectés entre eux.

Heureusement, Binna rentre tout juste d'un jogging. Bien qu'elle n'ait pas de carte de Mousarnen, elle peut récupérer la distance entre n'importe quelle paire parmis les $L + Y$ lieux, grâce à sa montre ultramoderne. À cause du format propriétaire ésotérique utilisé, il n'est malheureusement pas possible de savoir depuis la montre si un lieu est une maison ou une intersection. Cependant, nos souris connaissent le nombre de maisons L et le nombre d'intersections Y. La souris Binna veut repartir courir le plus rapidement possible, donc elle demande à Stofl de lui poser aussi peu de questions que possible, mais autant que nécessaire.

Vous pouvez supposer que les lieux sont numéroterés par $0, 1 ... L + Y - 1$. Vous pouvez envoyer des requêtes de la forme $d(a, b)$ et obtenir en retour le nombre de routes qu'il faut parcourir pour aller du lieu a au lieu b.

Sous-tâche 1: Résoudre un exemple (15 points)

Counstruisez toutes les cartes possibles avec les lieux $0, 1, 2, 3, 4, 5$ sachant les requêtes et réponses suivantes :

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

Expliquez brièvement pourquoi aucune autre carte n'est possible.

Sous-tâche 2: Résoudre le cas général (65 points)

Pouvez-vous aider Stofl à construire une carte de Mousarnen avec le nombre le plus faible asymptomatiquement de questions dans le pire cas ?

1. Olympiade d'Informatique des Souris

Votre priorité est d'optimiser le nombre de questions en premier, puis le temps d'exécution et enfin la mémoire. Chacun des trois asymptomatiquement, donc avec la notation grand-$O()$, et dans le pire cas.

Par pire cas, nous entendons pour la pire carte possible et que la montre n'a pas besoin de choisir les lieux à l'avance. Elle peut adapter sa *carte 'solution'* en arrière-plan, tant qu'elle reste cohérente avec les réponses déjà données. [2].

Pour obtenir tous les points, vous devez déterminer la carte de Mousarnen en $O(Y \cdot L)$ requêtes. Envoyer $O((L+Y)^2)$ requêtes vous rapportera au maximum 10 points. Notez qu'il est possible d'obtenir plus de 10 points, si le nombre de questions posées se situe entre $O((L+Y)^2)$ et $O(Y \cdot L)$.

Résoudre le cas particulier des cartes où $L = 2$, c'est-à-dire où tous les lieux sont sur une ligne, permet de remporter jusqu'à 10 points.

Note : vous **pouvez** obtenir les deux types de points partiels dans cette sous-tâche. Si vous résolvez le cas particulier de manière efficace et le cas général en $O((L+Y)^2)$ requêtes correctement, vous pouvez obtenir 20 points.

Sous-tâche 3: Minorer (20 points)

La souris Binna trouve qu'elle a trop dû attendre et affirme que Stofl posait ses questions de manière inefficace. Pour se défendre, Stofl veut montrer à Binna, lorsqu'elle rentrera de son jogging, un exemple de carte où il faut un certain nombre de questions pour être sûr d'avoir la bonne carte.

Étant donné le nombre de lieux N, pouvez-vous l'aider à construire une carte où n'importe quel algorithme nécessiterait au moins $\frac{N^2}{2023}$ requêtes dans le pire cas ? Vous êtes libre de choisir le nombre de maisons L et d'intersections Y, tant qu'ils respectent la contrainte $N = L + Y$ Vous pouvez supposer ne pas avoir de chance, c'est-à-dire que la montre peut changer les types des lieux pour lesquels aucune requête n'a été faite.

Pour obtenir tous les points, il faudra *montrer pourquoi* votre carte ne peut pas être construite en moins de $\frac{N^2}{2023}$ questions.

2. Comme ça, le fabricant pourra vendre le nouveau modèle amélioré l'année prochaine

Accident de code

La souris Binna participe à un concours de programmation. Elle était sur le point de soumettre sa solution qui devait lui permettre d'obtenir tous les points pour le dernier problème du concours. Malheureusment, au moment de soumettre arriva un *accident de complétion*. [1]

Le résultat est que les lignes de son code ont été mélangées aléatoirement. Comme il s'agissait d'un concours, son code n'était pas dans un logiciel de gestion de version, et elle doit maintenant remettre le fichier dans l'ordre manuellement.

L'éditeur de texte avancé de la souris Binna `souvim` a de nombreuses commandes créées spécialement pour aider des hackers d'élite comme Binna à accomplir ce genre de tâches. Une commande particulièrement efficace (`:rot3 i j k`) permet à Binna de choisir trois lignes différentes i, j et k. La commande effectue alors une *permutation circulaire* du contenu des lignes : la ligne j contiendra l'ancien contenu de la ligne i, la ligne k contiendra l'ancien contenu de la lgine j, et la ligne i contiendra l'ancien contenu de la ligne k.

Binna a horreur de la duplication de code, ce qui veut dire que toutes les lgines de son fichier de texte sont différentes. Binna sait donc exactement où chaque ligne doit aller.

La souris Binna remet très rapidement son code dans l'ordre, en utilisant uniquement des commandes `rot3` telles qu'expliquées plus haut. Bien évidemment, elle le fait de la manière la plus efficace, en utilisant le moins d'opérations `rot3` possible.

Arrivez-vous à résoudre ce type de tâche de manière générale ?

`souvim` numérote les lignes à partir de 0. Est donné un tableau de N entiers distincts entre 0 et $N-1$. Le i-ème entier p_i indique que la ligne i du code mélangé contient la ligne p_i du code source original. Trouvez la séquence la plus courte d'opérations `rot3` qui restorera le fichier source original, ou déterminez que cela est impossible.

Par exemple, considérez l'entrée suivante avec $N = 6$:

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

Dans ce cas, deux opérations `rot3` suffisent à remettre le code dans l'ordre :

1. `rot3`$(0, 3, 1)$. Cette opération déplace la ligne 0 à la ligne 3, la ligne 3 à la ligne 1 et la ligne 1 à la ligne 0. L'entrée suivante décrit la nouvelle situation :
   - $p_0 = 0$
   - $p_1 = 1$
   - $p_2 = 4$
   - $p_3 = 3$
   - $p_4 = 5$
   - $p_5 = 2$
2. `rot3`$(2, 4, 5)$. Cette opération déplace la ligne 2 à la ligne 4, la ligne 4 à la lgine 5 et la ligne 5 à la ligne 2. L'entrée suivante décrit la situation finale :
   - $p_0 = 0$
   - $p_1 = 1$
   - $p_2 = 2$
   - $p_3 = 3$
   - $p_4 = 4$
   - $p_5 = 5$

   Chaque ligne du fichier mélangé contient désormais la ligne correspondante du fichier original, nous avons donc remis le fichier dans l'ordre. De plus, il n'y a aucun moyen d'arriver à ce résultat avec moins d'opérations.

1. Plus précisément, elle a accidentellement exécuté le script bash `./shuffle.sh e.py` au lieu de `./submit.sh e.py`.

Sous-tâche 1: Petit exemple (10 points)

Fournissez explicitement n'importe laquelle des séquences les plus courtes d'opérations `rot3` qui remettent dans l'ordre un fichier décrit par l'entrée suivante avec $N = 11$:

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

Contrairement aux autres sous-tâches, il n'est pas nécessaire de justifier que votre solution est correcte si elle l'est effectivement. (Mais une explication des étapes intermédiaires peut permettre d'obtenir des points partiels si le résultat final est incorrect.)

Sous-tâche 2: Permutation cyclique de 2023 lignes (10 points)

Résolvez de façon optimale le cas suivant avec $N = 2023$, ou montrez que cela est impossible :

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

C'est-à-dire que pour $0 \leq i < 2022$, nous avons $p_i = i + 1$, et $p_{2022} = 0$.

Il n'est pas nécessaire de fournir une liste explicite des opérations effectuées, mais il faut néanmoins clairement les décrire. Notez qu'il faut maintenant argumenter pourquoi cette solution est correcte.

Sous-tâche 3: Permutation cyclique de 2024 lignes (10 points)

Résolvez de façon optimale le cas suivant avec $N = 2024$, ou montrez que cela est impossible :

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

C'est-à-dire que pour $0 \leq i < 2023$, nous avons $p_i = i + 1$, et $p_{2023} = 0$.

Il n'est pas nécessaire de fournir une liste explicite des opérations effectuées, mais il faut néanmoins clairement les décrire. Notez qu'il faut maintenant argumenter pourquoi cette solution est correcte.

Sous-tâche 4: Inversion (30 points)

Dans cette sous-tâche, $N \geq 1$ est arbitraire, et l'ordre des lignes a exactement été inversé : $p_i = N - 1 - i$.

Construisez un algorithme qui calcule une solution correcte dans ce cas précis. (Il doit soit produire une des plus courtes listes d'opérations `rot3` ou indiquer qu'il n'y a pas de solution.)

Sous-tâche 5: Cas général (40 points)

Dans cette sous-tâche, il n'y a pas de restriction supplémentaire : les lignes ont été mélangées de manière arbitraire. Construisez un algorithme qui calcule une solution correcte dans tous les cas. (Il doit soit produire une des plus courtes listes d'opérations `rot3` ou indiquer qu'il n'y a pas de solution.)

Vous pouvez obtenir jusqu'à 20 points en trouvant une solution qui est sous-optimale d'un facteur constant au plus (il faudra quand même montrer que c'est le cas). Les algorithmes qui utilisent plus de trois fois le nombre optimal d'opérations ne pourront remporter que jusqu'à 10 points.

La moitié des points est allouée à la description d'un bon algorithme, l'autre à la preuve de son exactitude.

Canaux impairs

Une équipe de recherche dirigée par la souris Binna a découvert les ruines de l'ancienne capitale du peuple Soien, une ancienne tribu connue pour ses avancées technologiques et ses capacités à résoudre des énigmes. Les ruines sont traversées par un système de canaux qui servaient probablement autrefois à transporter des marchandises. Il s'agit de $2 \cdot N$ sites reliés par M canaux. La souris Binna soupçonnait depuis longtemps que les Soiens attribuaient certaines significations à certains nombres. En particulier, les nombres qui ont un reste de 1 après avoir été divisés par 4 sont considérés chanceux. En fait, le nombre de canaux reliés à un site est toujours un multiple de 4 plus 1, comme si les Soiens avaient pris grand soin d'incorporer leurs chiffres porte-bonheur dans leur architecture.

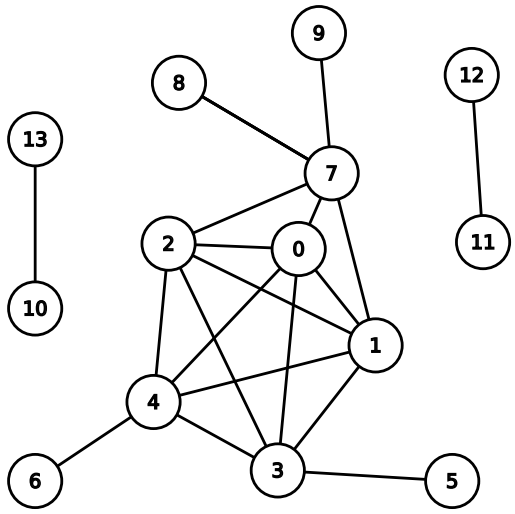
Malheureusement, le conseil de Sourinesse a décidé de rendre les ruines accessibles en tant qu'attraction touristique, malgré l'opposition farouche des scientifiques. La souris Binna travaille maintenant sur un concept de visite touristique qui garantira un traitement respectueux des vestiges soiens. Elle souhaite avant tout que les touristes ne circulent que sur les canaux et ne traversent pas les ruines. Pour que cela fonctionne, chaque canal doit recevoir un sens de passage pour les bateaux afin d'éviter les collisions. En outre, il est important pour elle de maintenir le respect de la culture soïenne et d'intégrer au moins certaines de leurs croyances dans le projet. Une légende soïenne nous apprend qu'un voyage ne doit pas toujours être chanceux, mais plutôt équilibré de toutes sortes de façons. C'est pourquoi la souris Binna souhaite que le nombre de canaux quittant un site soit impair pour N sites et pair pour les N autres sites.

Pour que ce concept ait une chance d'être approuvé par le conseil municipal, elle a besoin de votre aide pour déterminer la direction des canaux.

Sous-tâche 1: Résoudre un exemple (10 points)

La souris Binna a trouvé quelques dessins de constructions qui laissent entrevoir des tracés alternatifs de l'ancienne capitale. Dans ces dessins, les sites sont représentés par des cercles et les canaux par des lignes.

Elle en prend un de petite taille et essaie d'abord de trouver une solution à celui-ci. Pouvez-vous diriger les canaux, (en transformant les lignes en flèches), de sorte qu'exactement la moitié des sites aient un nombre impair de canaux qui s'en éloignent ?



Comme Binna vous a donné une copie, vous êtes autorisé à dessiner directement sur cette feuille. Mais n'oubliez pas de la rendre à la fin.

Sous-tâche 2: Nombre impair de sites ? (5 points)

Pendant que vous résolviez l'exemple, la souris Binna a examiné d'autres tracés, qui ont S sites et M canaux. Elle a remarqué que le nombre de sites est toujours pair lorsque le nombre de canaux quittant un site est un multiple de 4 plus 1. Pouvez-vous aider la souris Binna à expliquer pourquoi c'est toujours le cas ?

Sous-tâche 3: Impossible de n'être qu'impair (10 points)

Dans un conte populaire Soien, les multiples de 4 plus 3 sont également considérés comme chanceux, mais les mélanger avec des multiples de 4 plus 1 porte malheur. La souris Binna veut montrer la valeur de ce conte.

Pouvez-vous construire le plan d'un système de canaux remplissant les deux propriétés suivantes :

a) le nombre de canaux sur chaque site est un multiple de 4 plus 1 ou un multiple de 4 plus 3, en d'autres termes juste impair.
b) il est impossible de diriger les canaux de telle sorte que, à partir d'exactement N sites, un nombre impair de canaux s'éloignent (comme dans la légende soïenne).

Pouvez-vous l'aider à trouver un tel exemple ?

Sous-tâche 4: Trouver un algorithme (75 points)

Aidez Binna à trouver un algorithme pour diriger les canaux, à partir d'un plan des ruines.

Dans cette sous-tâche, l'accent est mis sur l'exactitude de votre algorithme.

Votre solution peut obtenir jusqu'à 55 points, s'il s'exécute en temps polynomial pour tout polynôme. Par polynôme, nous entendons quelque chose comme $O(n^{100})$ par opposition à $O(2^n)$.

De plus, vous pouvez marquer jusqu'à 35 points en résolvant un type spécial de configurations : Les tracés avec $2 \cdot N - 1$ canaux et où chaque site peut être atteint à partir de chaque autre site via des canaux avant de les diriger.

Remarque : Si vous tentez les deux, le polynôme et le type spécial, nous ne compterons que celui où vous aurez obtenu le plus de points, et non la somme des deux.