Deuxième Tour Théorique

Tâches

Swiss Olympiad in Informatics

3 mars 2018

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.
— 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.
— Regarde bien chaque sous-tâche, même quand tu n'as pas pu résoudre une des précédentes. Quelques sous-tâches peuvent être résolues sans l'aide des précédentes.

Évaluation

Ta solution est évaluée en fonction de son exactitude, de sa complexité spatio-temporelle ainsi que de la justification de ces éléments. Nous attendons de toi que tu fournisses une preuve ou une esquisse de preuve de l'exactitude et de la complexité de ta solution.

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

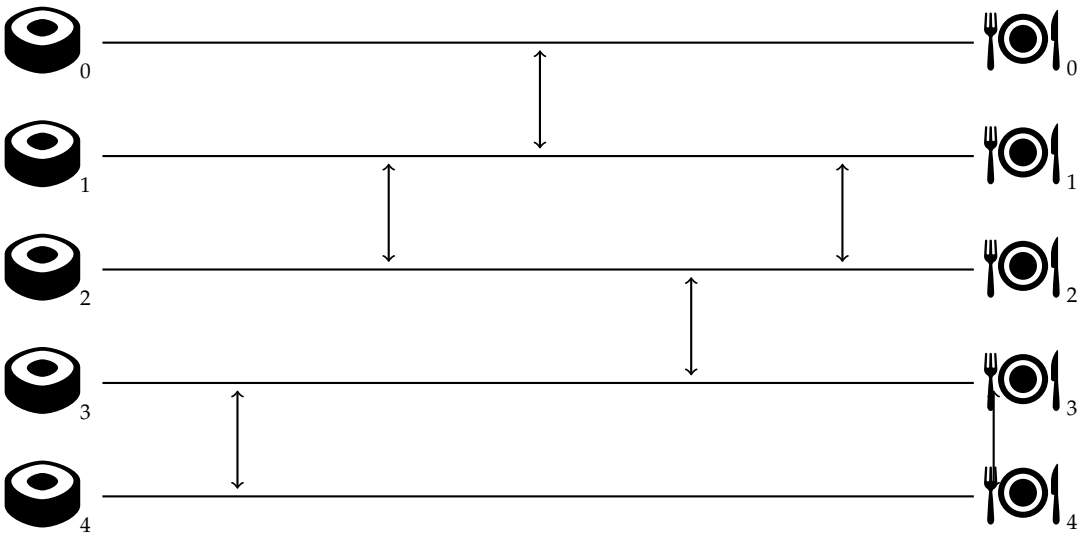
1. Décris l'idée derrière ton algorithme aussi clairement que possible.
2. Explique pourquoi ton approche résout correctement l'exercice.
3. Analyse la complexité spatio-temporelle asymptotique de ta solution.
4. Écris en pseudocode une solution. Tu peux omettre les parties faciles comme l'entrée et la sortie et utiliser des expressions mathématiques.

Si une partie de ta solution est utile pour plusieurs sous-tâches, il suffit de l'écrire une seule fois et de s'y référer à partir de là.

Bonne chance !

Tapis à suhshi

Au restaurant de Takano, il y a b tapis roulants à sushi qui sont utilisés pour distribuer les suhsi aux clients. Les tapis vont tout droit, d'ouest en est. Ils sont numérotés de 0 à $b - 1$ du nord au sud.



A l'extrémité ouest de chaque tapis se trouve un chef qui prépare un repas de sushi. Chaque chef a son propre plat de prédilection – on considère que le repas préparé par le chef du tapis i est différent des plats des chefs de tous les autres tapis.

A l'extrémité est de chaque tapis se trouve un siège pour un client. Puisque les tapis sont très espacés, le client assis au bout du tapis i ne peut consommer que les plats qui arrivent sur le tapis i.

Cette organisation est très rapidement devenue ennuyeuse pour les clients – peu importe où ils s'asseyaient, ils ne pouvaient goûter qu'un seul plat. C'est pour cette raison que le restaurant a décidé d'employer w clients qui peuvent déplacer les repas entre des tapis adjacents.

Les serveurs sont numérotés de 0 à $w - 1$. Pour chaque j, le serveur j est stationné à la coordonnée x_j entre les deux tapis adjacents y_j et y_{j+1}. Ce serveur peut prendre n'importe quel plat qui arrive sur le tapis y_j et le placer sur le tapis y_{j+1} ou vice versa. Tu peux supposer que tous les serveurs ont des coordonnées en x différentes et donc qu'ils n'interfèrent pas entre eux (par exemple il n'est pas possible que deux d'entre eux puissent essayer de prendre un plat en même temps. Cependant, il se peut que plusieurs serveurs déplacent des plats entre les deux mêmes tapis. Quand un client passe une commande, tous les serveurs coordinent leurs efforts pour leur livrer le repas de leur choix (si possible) en le déplaçant entre tapis adjacents.

Sous-tâche 1: Servir tous les repas à tous les clients (10 Points)

Pour $b = 10$ tapis roulants à sushi, trouve n'importe quel placement de 47 serveurs ou moins de sorte à ce que chaque repas puisse être livré à chaque siège dans le restaurant. Explique pourquoi ta façon de placer les serveurs fonctionne.

Sous-tâche 2: Nombre minimal de serveurs (30 Points)

Takano aimerait employer aussi peu de serveurs que possible, tout en étant toujours capable de servir chaque plat à chaque siège dans le restaurant. Pour un $b \geq 2$ arbitraire, quel est le nombre minimal de serveurs requis (en fonction de b) et comment ces serveurs devraient-ils être placés ?

Pour 20 points, prouve que ta solution pour la sous-tâche précédente utilise bien le nombre minimal de serveurs.

Sous-tâche 3: Serveurs sous-optimaux (60 Points)

Les serveurs chez Takano n'ont pas été capables de résoudre les deux sous-tâches précédentes. Ils restent donc à certaines positions arbitraires et font de leur mieux pour servir les plats aux clients.

Tu reçois b, w et les nombres x_j et y_j qui dénotent le placement des serveurs comme décrit plus haut. Soit m_i le nombre de différents plats que les serveurs peuvent amener au siège au bout du tapis roulant i. Trouve un algorithme qui calcule les nombres $m_0, \ldots, m_{b-1}$.

Pour 20 de ces points, prouve que ta solution est correcte.

Métro d'Osaka

La souris Stofl vient d'arriver à Osaka. Pour utiliser les transports publics, Stofl achète une SOI-Card (Smart Osaka Inter Card). La ville est représentée à l'aide de n arrêts de métro qui sont connectés avec m lignes de métro. On suppose que chaque ligne n'a que deux arrêts : il n'existe aucun arrêt intermédiaire. Suivant les lignes, les temps de parcours peuvent être différents. Par contre, le temps de parcours d'une ligne est indépendant du sens de parcours de la ligne. Grâce à l'ingéniosité des concepteurs du réseau de transport, le temps nécessaire pour changer de ligne dans une station est nul. De plus, il est garanti qu'il est toujours possible de joindre chaque station à partir de chaque station, éventuellement à l'aide de changements.

Le prix d'un trajet est calculé de la manière suivante. Chaque arrêt est situé dans exactement une zone de trajet (il y a exactement z zones différentes). Faire une partie du trajet dans la zone i coûte p_i yens. Le prix du trajet entier à travers plusieurs arrêts différents est égal à la somme des prix des zones traversées (chaque zone est compté seulement une fois). Par exemple, si Stofl passe par les zones 0, 0, 3, 0, 4, 4 et 3, elle devra payer $p_0 + p_3 + p_4$.

Un trajet le *moins* cher est un trajet avec un prix minimal.

Quand Stofl utilise la SOI-Card, alors elle met sa carte dans le lecteur au premier arrêt (d'où elle part) et au dernier arrêt (là où elle désire aller). Le prix facturé est le coût d'un trajet le moins cher entre les deux arrêts.

La souris Stofl aime bien la manière dont les trajets sont facturés. Cependant, elle se demande si cela est consistant. Un réseau est un réseau consistant si il n'y a aucune paire d'arrêts telle que le trajet le plus court entre eux (en terme de temps) est strictement plus cher que un trajet le moins cher entre les deux arrêts.

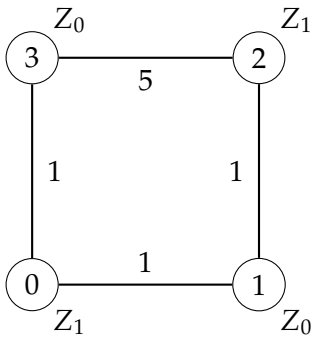
Comme le réseau de transports publics de Osaka est malheureusement assez complexe, la souris Stofl a besoin de ton aide.

Un exemple de réseau peut être observé dans la figure. Il y a deux zones ($z = 2$) qui sont mises entre parenthèses dans l'exemple. Le temps de trajet est donné à côté de chaque ligne de métro. La zone 0 coûte $p_0 = 1$ yen et la zone 1 coûte $p_1 = 2$ yens. L'unique trajet le plus court entre les arrêts 3 et 2 est décrit par la séquence d'arrêts $3 \to 0 \to 1 \to 2$. Un trajet le moins cher de l'arrêt 3 à l'arrêt 2 est la ligne directe entre ces deux arrêts ou aussi la séquence $3 \to 0 \to 1 \to 2$.

Sous-tâche 1: Trouver un réseau qui n'est pas consistant (10 points)

Trouve un réseau et une répartition des arrêts dans les zones tel que le réseau n'est *pas* consistant.

Le réseau doit être donné comme un graphe non orienté. Le temps de trajet pour chaque arc doit être noté directement au sein du graphe à côté de chaque arc (le temps de trajet est indépendant de la direction de trajet). De plus, chaque noeud doit appartenir



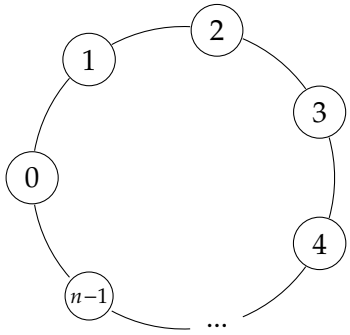
à exactement une zone. Finalement, les prix de chaque zone doivent être listés dans un tableau (notons que comme on ne paye qu'une fois par zone, il ne faut pas marquer le prix pour chaque arrêt mais pour chaque zone).

Sous-tâche 2: Un réseau circulaire (30 points)

Supposons maintenant que le réseau de transport est un cycle. Cela veut dire que l'arrêt i a une ligne directe vers l'arrêt $i + 1$ pour $0 \leq i < n - 1$, et qu'il y a une ligne directe entre les arrêts $n - 1$ et 0.

Propose un algorithme avec une complexité temporelle et spatiale minimale qui décide si le réseau est consistant.

Un réseau circulaire est donné pour illustration dans la figure suivante.



Sous-tâche 3: Le réseau de transport d'Osaka (60 points)

Malheureusement, le réseau à Osaka n'est pas un cycle, mais comporte heureusement peu de zones (par rapport au nombre d'arrêts n).

Propose un algorithme qui détermine pour un réseau donné s'il est consistant. Optimise la complexité temporelle et spatiale par apport à n et m en priorité, puis par rapport à z.

Dandyshow

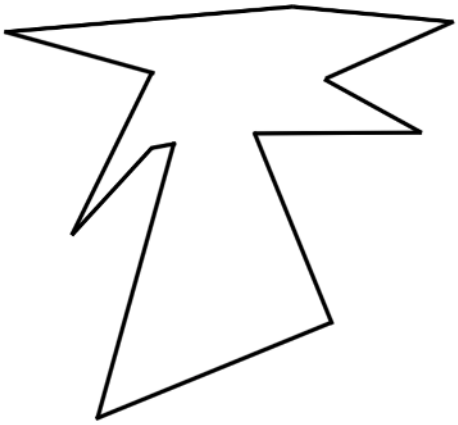
La souris Stofl est vraiment célèbre dans son univers et détient légitimement le titre de dandy de l'espace. Dans l'une de ses innombrables aventures, il a compris que l'univers dans lequel il vit est un polygone bidimensionnel (du moins, il ne peut pas traverser les murs de cet univers, dont il a créé une carte). Il est devenu riche et célèbre dans son univers et n' a plus qu'un seul désir : qu'on se souvienne de lui. Pour ne pas être oublié, il a l'intention de créer la plus grande soirée disco de tout son univers ! Cependant, Stofl veut que la soirée disco ait lieu à un endroit où tout le monde dans l'univers puisse le voir directement. Aidez Stofl à trouver un tel endroit dans son univers.

Description formelle On te donne un polygone simple bidimensionnel (où "simple" signifie que les arêtes du polygone ne se croisent pas entre elles et ne se rejoignent qu'aux sommets du polygone). Trouve un point dans ce polygone tel que tous les autres points du polygone peuvent y être reliés par une ligne droite, sans croiser ou toucher les arêtes du polygone.

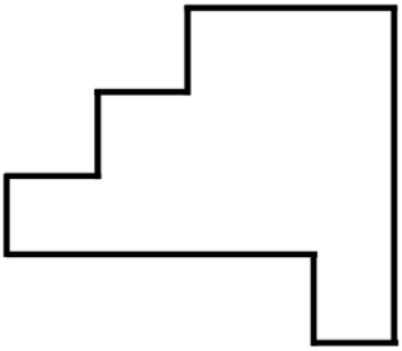
Sous-tâche 1: Résoudre la tâche pour certains univers donnés (10 points)

Marque tous les endroits où Stofl pourrait organiser sa soirée disco.

1.



2.



Sous-tâche 2: Résoudre la tâche pour les univers rectilignes (35 points)

Construis un algorithme qui trouve un point dans l'univers où Stofl peut organiser sa soirée disco, ou renvoie "Dommage" s'il n'y en a pas.

Le polygone est donné sous la forme d'une liste de coordonnées x et y indiquant les points du polygone dans le sens des aiguilles d'une montre.

De plus, Stofl a découvert que chacun des bords du polygone est horizontal ou vertical (en d'autres termes, tous les angles entre les bords sont de 90° ou 270° degrés). Nous supposons que l'univers n'est pas tordu.

Sous-tâche 3: Résoudre le cas général (55 points)

Résous le cas général, où le polygone peut être n'importe quel polygone simple défini plus haut.

Approvisionnement en gaz

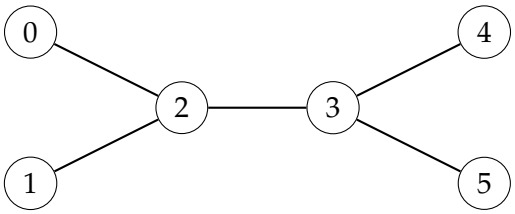
La souris Stofl a changé de métier et est maintenant responsable pour l'approvisionnement en gaz dans Tsukaba. Le réseau de gaz est formé d'intersections et de tuyaux (qui n'ont pas de direction) qui connectent les intersections. Dans le réseau, il est possible d'aller de chaque intersection à chaque autre intersection en suivant les tuyaux. De plus, les tuyaux ne forment pas de cycles. Cette structure est appelée un arbre en théorie des graphes. Finalement, dans chaque réseau il y a au maximum 3 tuyaux qui sont connectés à une intersection (on dit que le degré maximal est 3).

Malheureusement, lors du premier jour de travail dans ce métier, la souris Stofl rencontre des problèmes techniques. En effet, la demande en gaz est anormale, elle est beaucoup plus importante que d'habitude. Stofl est convaincue qu'une intersection a une fuite. Cependant, elle n'arrive pas à déterminer simplement quelle intersection est cassée. Heureusement, elle a préparé un plan : Dans la nuit où la demande de gaz est nulle, tout le gaz va aller vers l'intersection avec la fuite et va s'échapper. Stofl peut déterminer pour chaque tuyau la direction du parcours du gaz. Combien de tuyaux faut-il que Stofl vérifie dans le pire des cas afin de déterminer l'intersection avec la fuite ?

Note : Le but de l'exercice est d'optimiser le cas où le nombre d'intersections est important. Par exemple, il est mieux d'avoir à vérifier $\sqrt{N} + 100$ tuyaux par rapport à $2 \cdot \sqrt{n} + 2$ tuyaux. Les termes non-dominante seront ignorés lors de la correction (une solution qui vérifie $n^{\frac{1}{2}} + 6n^{\frac{1}{4}} + 9$ tuyaux est considérée équivalente à une solution qui vérifie $\sqrt{n} - 42$ tuyaux).

Description formelle Pour un arbre donné où le degré maximal est 3, trouve l'intersection L recherchée. On peut faire des requêtes du type suivant : pour un tuyau représenté par les deux intersections A, B adjacentes qu'il relie, on reçoit l'intersection (soit A, soit B) la plus proche de L.

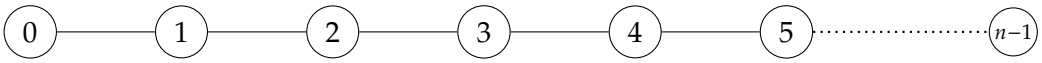
Sous-tâche 1: Résoudre un exemple (10 points)



Quel est le nombre minimal de requêtes qu'il faut faire afin de trouver la fuite à coup sûr ? Pourquoi est-ce que ces requêtes ne contiennent pas deux fois la même requête ?

Sous-tâche 2: Un réseau un peu plus simple (30 points)

Observons en premier un cas un peu plus simple, où chaque intersection est adjacente à au plus 2 tuyaux. De plus, on part du principe qu'il y a un tuyau de l'intersection i à l'intersection $i + 1$ pour tout $0 \leq i < n - 1$ (n représente le nombre d'intersections). Le réseau a donc l'aspect suivant :



Pour un réseau de ce type donné, quel est le nombre minimal de requêtes nécessaires pour trouver la fuite ? Propose un algorithme que Stofl peut utiliser pour déterminer ces requêtes. Détermine la complexité temporelle de l'algorithme proposé. En d'autre termes, fournis une preuve constructive qui montre que le nombre minimal de requêtes proposé est atteignable.

Sous-tâche 3: Le cas général (60 points)

Propose un algorithme pour le cas général (où le degré maximal est 3). Montre que le nombre minimal proposé de requêtes pour trouver la fuite est atteignable. Décris un algorithme pour Stofl qui trouve la fuite en utilisant ce nombre de requêtes. Analyse la complexité temporelle de l'algorithme proposé.