Deuxième Tour Théorique

Tâches

Swiss Olympiad in Informatics

4 mars 2017

Instructions

Les solutions seront évaluées d'après des critères similaires aux ceux des exercices théoriques du premier tour. Ce qui est le plus important, c'est que les solutions soient correctes et que la complexité en temps soit juste. La qualité de la description ainsi que les arguments pour la justesse de la solution seront également pris en compte.

Si un algorithme est demandé, l'approche suivante t'aidera à le formuler :

1. Décris l'idée générale de ton algorithme le plus clairement possible.
2. Explique pourquoi ton approche est correcte.
3. Analyse la complexité en temps et en mémoire.
4. Écris un programme en pseudocode. Tu peux ignorer les opérations simples telles que l'entrée et la sortie et tu peux utiliser des expressions mathématiques.

Si une partie de ta solution peut être appliquée à plusieurs exercices, il suffit de l'écrire une seule fois et de simplement indiquer que tu la réutilises dans les autres exercices.

Fleptons

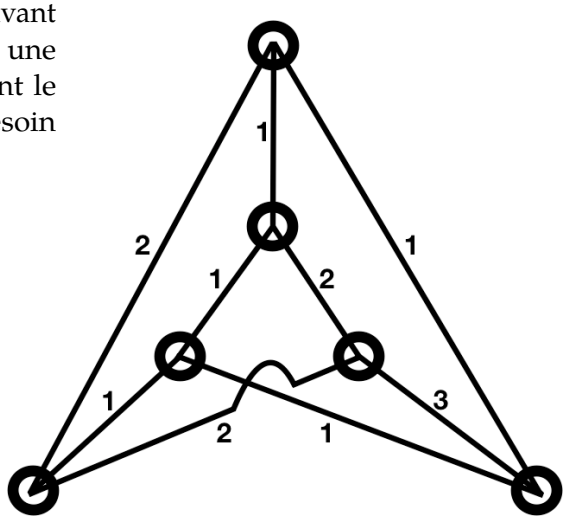
Stofl la souris est un physicien théorique brillant et vient de t'informer de sa nouvelle découverte : le flepton ! Un paire de fleptons peut apparaître spontanément à un endroit dans l'espace, suite à quoi les deux fleptons vont se séparer. Après séparation, ces deux fleptons vont suivre une certaine trajectoire dans l'espace et finir par entrer en collision et disparaître. Un flepton ne va jamais être situé à un endroit que lui-même ou son flepton homologue a déjà occupé plus tôt. Lorsque les deux fleptons se trouvent au même endroit au même moment, ils disparaissent.

Ta tâche est d'aider Stofl à obtenir le prix Nobel de Physique, qu'il ne peut obtenir que s'il parvient à prouver l'existence des fleptons à l'aide d'une expérience. Stofl a déjà construit un réseau de m tubes moléculaires et n détecteurs, avec lesquels il espère détecter la paire de fleptons. Un tube moléculaire se situe toujours entre deux détecteurs et pour des raisons nanophysiques, on ne peut pas mettre plus d'un tube entre deux détecteurs. Les fleptons se déplacent à la vitesse de la lumière et les tubes moléculaires sont tellement petits que les fleptons ne peuvent s'y déplacer que dans une seule direction. Par simplicité, nous décrivons la longueur des tubes par le temps nécessaire à un flepton pour le traverser. Tu tâche est d'aider Stofl à déterminer deux détecteurs où pourraient apparaître ou disparaître des fleptons, de sorte que Stofl puisse se concentrer juste sur ces deux détecteurs-là.

Description formelle : Étant donné un graphe simple orienté et pondéré, trouve les deux chemins *les plus courts* qui commencent tous deux au même noeud et se terminent tous deux au même noeud, et couvrent la même distance (somme des segments) et n'ont aucun autre noeud en commun.

Sous-tâche 1 : Résous l'exemple (10 points)

Trouve deux tels chemins dans le système suivant de tubes moléculaires. Chaque cercle indique une intersection (noeud) et les nombres indiquent le temps en nanosecondes dont un flepton a besoin pour traverser un tube.



Sous-tâche 2 : Développe un algorithme (50 points)

Développe un algorithme qui puisse, pour n'importe quel système de tubes moléculaires, trouver deux chemins possibles pour une paire de fleptons. Mentionne la complexité en *temps* et en *mémoire* de ton algorithme.

Sous-tâche 3 : Configuration pour chaque paire de détecteurs (40 points)

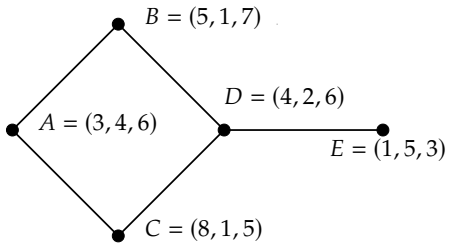
Oh non ! Stofl a utilisé presque tout son argent pour des détecteurs et n'a toujours détecté aucun flepton. À cause de son budget limité, il n'est possible d'acheter que des tubes de la même taille. De plus, il souhaite augmenter la probabilité que les détecteurs trouvent un flepton. Aide-le à construire une nouvelle configuration de tubes à n noeuds de sorte que chaque paire de noeuds non-adjacents peut former et faire disparaître des fleptons. Moins de tubes tu utilises, plus ta solution recevra des points. Par exemple, une solution avec $m = 2 \cdot n$ tubes recevra plus de points qu'une solution avec $m = 5 \cdot n$ tubes.

Téléportation

Les souris d'une planète lointaine se déplacent de ville en ville à l'aide d'un réseau de téléports (stations de téléportation). Un téléport comprend trois émetteurs opérant sur des fréquences différentes.

À cause des interférences, une souris ne peut se déplacer entre deux téléports que si leurs émetteurs n'utilisent pas la même fréquence.

Par exemple, la figure suivante illustre 5 téléports A, B, C, D et E ainsi que leurs fréquences d'opérations. Les liens montrent toutes les paires possibles que les souris peuvent utiliser pour leurs déplacements.



Comme on peut le voir, parfois il est nécessaire de passer par une série de téléports intermédiaires pour se rendre d'une ville à une autre. Par exemple, pour se rendre de B à C, il faut passer par A (ou D). Formellement, nous définissons $d(x, y)$, la distance entre deux téléports x et y, comme le nombre minimum de téléportations nécéssaires pour se rendre de l'un à l'autre. Dans le cas mentionné ci-dessus, $d(B, C) = 2$.

Les souris sont très fières de leur réseau et prétendent qu'il est très efficace : on ne doit pas passer par trop de téléports intermédiaires pour se rendre de n'importe quelle ville à n'importe quelle autre.

Stofl te demande de l'aider à vérifier l'efficacité du réseau. Le mésure pour ça est la distance maximale. Dans l'exemple ci-dessus $d(A, E) = 3$ est la valeur maximale recherchée.

Sous-tâche 1 : Résous l'exemple (10 points)

Il y a 7 téléports opérant avec les fréquences suivantes : $A = (6, 8, 3)$, $B = (4, 7, 0)$, $C = (5, 6, 2)$, $D = (1, 7, 3)$, $E = (2, 4, 5)$, $F = (6, 8, 1)$, $G = (4, 8, 0)$. Liste toutes les paires de téléports entre lesquels il est possible de voyager directement.

De plus, trouve une paire de téléporteurs x et y dans ce réseau de téléportation pour que $d(x.y)$ soit la distance maximale la plus courte.

Sous-tâche 2 : Réseau avec distance maximale (45 points)

Stofl se demande s'il existe un réseau pour lequel la distance entre deux téléports est grande.

Tâche Dessine un réseau d'au moins deux téléports et liste leurs fréquences d'opération de sorte que la distance entre une paire de téléports x et y est la plus grande possible.

Points Pour un réseau avec une distance maximale ℓ, tu recevras $\min(5\ell - 15, 45)$ points. C'est-à-dire que si la distance est 12 ou plus, tu recevras le maximum de points.

Exemple Les téléports dans l'exemple ci-dessus ne sont pas très éloignés. La distance maximale de 3 correspond aux téléports A et E. Ce réseau te vaudrait 0 points.

Sous-tâche 3 : Limite pour la distance maximale (45 points)

Stofl s'efforce de trouver un réseau avec une distance maximale de 2017 et commence à penser qu'il est impossible de parvenir à une telle distance.

Tâche Donne une preuve que la distance maximale dans un réseau de téléports est de moins de 2017. Essaie d'améliorer cette limite le plus possible.

Points Toute preuve complèment correcte avec une constante arbitraire (< 2017) recevra 20 points. Si la constante est au plus deux fois l'optimum, tu recevras 25 points. Les autres limites recevront un nombre de points selon une échelle de points correspondante.

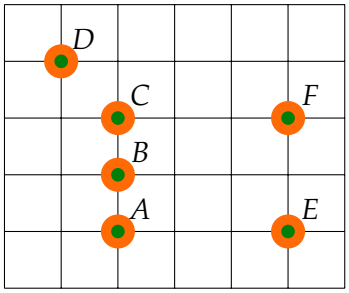
Infectionpoints

La souris Stofl a planté des carrotes. Malheureusement, Stofl est un scientifique et faire pousser des plantes l'ennuie rapidement. Il a donc commencé à s'intéresser aux parasites présents dans son jardin. Stofl est convaincu qu'il a trouvé un nouveau type de parasite et lui a donné son propre nom : "Stoffulus". Ce qui est particulièrement intéressant chez Stoffulus est sa manière de se propager. Une infection débute toujours par deux carottes qui sont infectées. Une carotte saine devient infectée lorsque la distance entre cette carotte et une carotte déjà infectée est plus petite que la distance entre deux carottes infectées.

La souris Stofl veut maintenant faire la chose suivante : lorsqu'il infecte 2 carottes avec Stoffulus, le parasite va se propager jusqu'à ce qu'il ne parvienne plus à atteindre de nouvelles carottes. Nous appellerons cette situation un état final. Selon quelles carottes furent choisies au début, les carottes infectées seront différentes. La souris Stofl souhaiterait apprendre combien d'états finaux il existe dans son jardin. Or, il existe de nombreuses possibilités de choisir les 2 carottes initialement infectées (pour être exact, il en existe $\binom{N}{2} = \frac{N(N+1)}{2}$). La souris Stofl n'a pas envie de mener autant d'expériences et te demande donc ton aide.

Description formelle Donné sont n points $p_1, \ldots, p_n$ dans le plan (le jardin de Stofl). Chaque expérience commence avec deux points infectés : p_a, p_b. Un point p_x devient infecté lorsqu'il existe 3 points infectés (pas forcément distincts, c'est-à-dire pas forcément différents les uns des autres) p_v, p_w, p_y tels que $dist(p_x, p_y) \leq dist(p_v, p_w)$ [1]. Lorsqu'il ne reste plus aucun point qui pourrait être infecté, l'état final est atteint. Un état final existe pour chaque paire p_a et p_b. Combien il y a-t-il d'état finaux **différents** ?

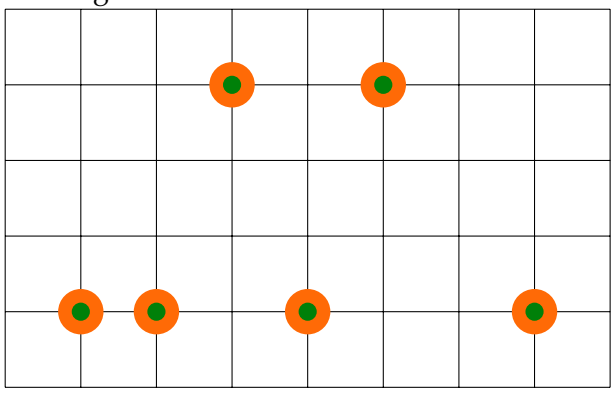
Exemple Supposons que toutes les carottes se trouvent sur un quadrillage. Il existe 3 états finaux différents. Si les deux premières carottes infectées se trouvent d'un même côté, alors aucune autre carotte ne sera infectée. Dans tous les autres cas, lorsqu'une carotte est dans la partie gauche et l'autre dans une partie droite, toutes les 4 carottes seront infectées.



1. $dist$ note la distance euclidienne $dist((x_1, y_1), (x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

Sous-tâche 1 : Résoud l'exemple (10 points)

Pour cette exemple, suppose de nouveau que toutes les carottes sont placées sur un quadrillage. Combien d'états finaux existe-t-il ?



Sous-tâche 2 : Développe un algorithme (90 points)

Développe un algorithme pour Stofl qui accepte comme entrée une liste de points $L = [(x_1, y_1), \ldots, (x_n, y_n)]$ and donne en sortie un seul nombre entier, le nombre d'états finaux différents.

Indique aussi la *complexité en temps* (le temps que met ton algorithme en fonction du nombre de carottes n) et la *quantité de mémoire utilisée* de ton algorithme.

Cake Protection Agency

Dans les forces spéciales policières de Sourisland, il y a plusieurs unités qui ont chacune une tâche principale. L'une des plus importantes parmi elles est la CPA (*Cake Protection Agency* en anglais), qui protège tous les gâteaux dans le pays contre les attaques terroristes.

La CPA vient d'apprendre que la souris Gehr, souris terroriste sophistiquée et extrêmement dangereuse, planifie une attaque. Elle compte sortir de chez elle, marcher jusqu'à un gâteau et le jeter par terre. Les agents de la CPA peuvent arrêter la souris terroriste soit en arrivant au gâteau avant elle (et non pas au même temps, sinon elle aura juste assez temps pour le pousser par terre), soit en l'interceptant avant qu'elle arrive au gâteau.

Ta tâche est d'aider les agents de la CPA à gâcher le plan machiavellique de la souris Gehr.

Description formelle Sourisland est représenté en tant que graphe orienté et pondéré $G = (V, E)$ avec $n + 3$ arêtes $V = \{v_{\text{cpa}}, v_{\text{gehr}}, v_{\text{cake}}, v_1, \ldots v_n\}$: la position de départ des agents de la CPA, la position de départ de la souris Gehr, la position du gâteau et le nombre de croisements restants à Sourisland.

Une arête $e = (u, v, t) \in E$ lie $u \in V$ à $v \in V$ avec le temps de voyage $t \geq 0$, qui est un nombre entier. Il est important de savoir que toutes les rues à Sourisland sont des rues à sens unique ; l'arête (u, v, t) peut seulement être utilisée pour aller de u à v et non pas d'aller de v à u. Il faudrait qu'il y ait une arête (v, u, t') pour aller de v à u.

Les agents de la CPA peuvent arrêter la souris Gehr soit en arrivant avant elle au gâteau, soit en arrivant en même temps qu'elle sur le sommet $v \neq v_{\text{cake}}$.

Sous-tâche 1 : Dernier moment (15 Punkte)

La souris Gehr n'a pas encrypté ses communications et son plan secret avec la route $p = (v_{\text{gehr}}, \ldots, v_{\text{cake}})$ qui indique son chemin pour aller au gâteau, ainsi que son temps de départ T, un nombre entier, ont été interceptés par la CPA.

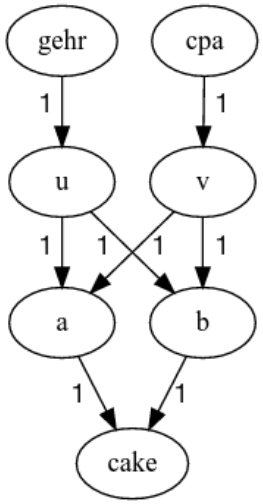
Décris un algorithme qui trouve le dernier temps T' (de nouveau un nombre entier) où les agents peuvent sauver le gâteau en partant de la position v_{cpa}, soit en arrivant avant la souris Gehr au gâteau, soit en l'arrêtant sur le chemin. Argumente pourquoi ton algorithme est correct et analyse sa complexité en temps.

Sous-tâche 2 : Nombre quelconque d'agents (35 Punkte)

Dans ce scénario-ci, la souris Gehr avait appris de la surveillance à grande échelle de la CPA et a réussi à garder son plan secret. Suppose que la souris Gehr part de chez elle au moment T (toujours un nombre entier) et que la CPA peut envoyer autant d'agents qu'elle veut. Trouve le dernier moment T' auquel les agents peuvent être envoyé pour arrêter la souris Gehr. Argumente pourquoi ton algorithme est correct et analyse sa complexité en temps.

Tout le monde connaît la position de tout le monde à chaque moment. De plus, si la souris Gehr et un agent de la CPA doivent prendre une décision au même moment (choisir leurs prochaines arêtes respectives), la souris Gehr doit d'abord prendre son choix que l'agent connaîtra avant d'avoir choisi sa prochaine arête à lui.

Dans l'exemple suivant, toutes les arêtes ont un poids de 1 et un seul agent de la CPA qui part en même temps que la souris Gehr sera toujours capable de l'arrêter avant qu'elle arrive au gâteau. D'abord, la souris Gehr ira au sommet u et l'agent ira au sommet v. Puis, la souris Gehr doit prendre une décision entre le sommet a et le sommet b et l'agent pourra la suivre (soit sur a, soit sur b) et l'arrêter.



Sous-tâche 3 : Un Agent (50 Punkte)

Même chose que dans l'exercice précédent avec une seule différence : la CPA ne peut envoyer qu'un seul agent. Détermine de nouveau le dernier moment T' (nombre entier, si jamais) auquel l'agent de la CPA peut quitter le QG de la CPA v_{cpa} pour qu'il puisse arrêter la souris Gehr. Argumente pourquoi ton algorithme est correct et analyse sa complexité en temps.