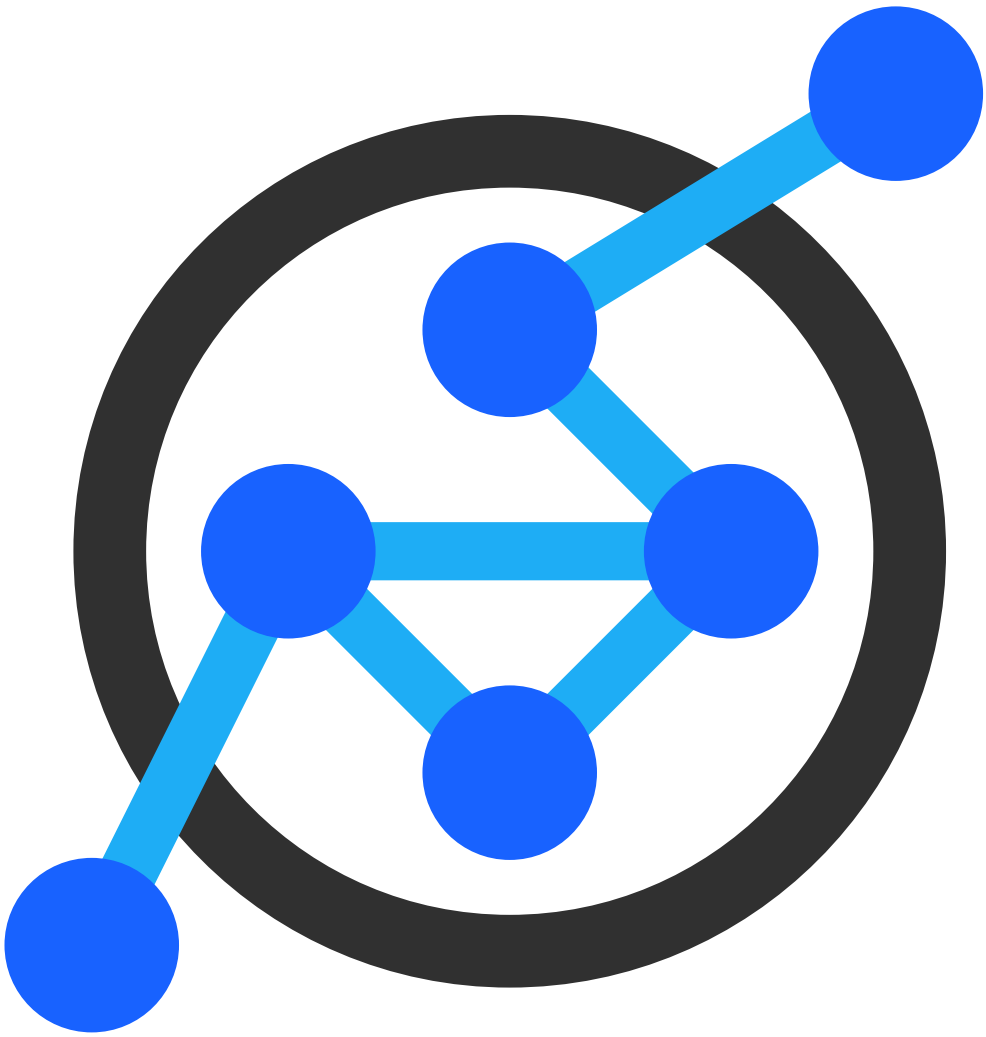
Deuxième Tour Théorique

Tâches



Swiss Olympiad in Informatics

7 mars 2020

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.

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 !

evaporation

Souris Binna est super heureuse de pouvoir participer à l'IOI 2020 à Singapour. Cependant, une fois arrivée, elle s'est vite rendu compte qu'il fait bien trop chaud à Singapour pour une petite souris comme elle. Afin de ne pas mourir de déshydratation, Souris Binna a décidé d'acheter beaucoup de verres d'eau, à savoir n. Le verre i-ème contient a_i millilitres d'eau. Malheureusement, en raison du temps chaud, chaque minute, un millilitre d'eau s'évapore de chaque verre. Comme Mouse Binna ne veut pas passer tout son temps à aller au supermarché pour acheter de l'eau, au début de chaque minute, Mouse Binna a décidé de verser toute l'eau d'un verre dans un autre verre. Aidez la souris Binna à maximiser la quantité d'eau après n minutes.

Description formelle Étant donné est une liste de n entiers non négatifs $a_0, \ldots, a_{n-1}$. Chaque minute, voici ce qui se passe : vous pouvez transformer a_i à $a_i + a_j$ et a_j à 0 (pour un i et j de votre choix), puis tous les entiers $a_i > 0$ deviennent $a_i - 1$. Trouvez la somme maximale de tous les entiers après n minutes.

Sous-tâche 1: Un exemple (10 points)

On te donne la liste suivant : $\{1, 2, 3, 4\}$. Quelle est la quantité maximale d'eau encore présente après $n = 4$ minutes ? Énumérez les opérations et l'heure à laquelle vous effectuez ces actions afin d'obtenir cette valeur à la fin. Vous n'avez pas besoin de justifier votre réponse.

Sous-tâche 2: Tous pareil (20 points)

Nous supposons maintenant que tous les verres contiennent au début exactement k millilitre d'eau ($x_i = k$ pour tous les $0 \leq i < n$). Construisez un algorithme qui calcule la valeur maximale dans ce cas particulier.

Sous-tâche 3: Permutations (30 points)

Cette fois, tous les entiers a_i forment une permutation des nombres $\{1, 2, \ldots, n\}$. Construisez un algorithme dans ce cas.

Sous-tâche 4: Cas général (40 points)

Il n'y a plus de restrictions sur les valeurs des a_i. Construisez un algorithme pour ce cas général.

Chasse au nombre

Stofl participe à une "chasse au nombre". Le but est de trouver un nombre manquant. Il y a un tableau – dont Stofl n'a pas connaissance – de $2^n - 1$ entiers uniques entre 0 et $2^n - 1$. Il ne manque donc qu'un seul nombre. Aide Stofl à déterminer quel est le nombre manquant et ainsi à gagner la chasse au nombre.

Chaque nombre dans le tableau secret peut être écrit comme une suite de n bits. Par exemple, $6_{(10)} = 4+2 = 1\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0 = 110_{(2)}$ ou $13_{(10)} = 8+4+1 = 1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 = 1101_{(2)}$. Le 0-e bit est le coefficient de 2^0 (c'est-à-dire 1 si le nombre est impair et 0 s'il est pair), le 1-er bit est le coefficient de 2^1, et ainsi de suite.

Il y a $(2^n - 1)\cdot n$ quêtes disponibles pour Stofl. Résoudre la quête $q(i, j)$ lui révèle le j-e bit du i-e nombre. Dans le tableau ci-dessous, résoudre $q(5, 2)$ révèle 1 et résoudre $q(0, 2)$ révèle 0. Les quêtes sont résolues une à une, donc Stofl peut décider quelle quête entreprendre en se basant sur les résultats des quêtes précédentes.

Aide Stofl à gagner le chasse au nombre en passant par le moins de quêtes possible.

Nombre	Bits		
$a_0 = 3$	0	1	1
$a_1 = 1$	0	0	1
$a_2 = 6$	1	1	0
$a_3 = 7$	1	1	1
$a_4 = 0$	0	0	0
$a_5 = 4$	1	0	0
$a_6 = 2$	0	1	0

Exemple d'un tableau secret

Nombre	b_2	b_1	b_0
a_0	$q(0,2)$	$q(0,1)$	$q(0,0)$
a_1	$q(1,2)$	$q(1,1)$	$q(1,0)$
a_2	$q(2,2)$	$q(2,1)$	$q(2,0)$
a_3	$q(3,2)$	$q(3,1)$	$q(3,0)$
a_4	$q(4,2)$	$q(4,1)$	$q(4,0)$
a_5	$q(5,2)$	$q(5,1)$	$q(5,0)$
a_6	$q(6,2)$	$q(6,1)$	$q(6,0)$

Tableau des quêtes disponibles

Description formelle Soit $a_0, a_1, \ldots, a_{2^n-1}$ une permutation des nombres $\{0, 1, \ldots, 2^n - 1\}$. Tu peux faire des requêtes $q(i, j)$ pour $0 \le i \le 2^n - 2$ et $0 \le j < n$ pour accéder au j-e bit du i-e nombre (note que tu ne peux pas faire de requête à propos du $(2^n - 1)$-e nombre). Trouve la valeur de a_{2^n-1} en utilisant le nombre minimal de requêtes.

Sous-tâche 1: Analyse le tableau de Stofl (10 points)

Stofl a déjà terminé certaines quêtes et a obtenu le tableau suivant :

Nombre	b_2	b_1	b_0
a_0		1	0
a_1	1		
a_2	1		1
a_3		1	0
a_4			0
a_5	1		1
a_6			

Résous les tâches suivantes (aucune justification n'est requise) :

— Pour chaque élément de la liste a_i, $0 \le i \le 6$, note l'ensemble des valeurs possibles.

— Le nombre manquant peut être absolument déterminé par le biais d'une seule quête. Laquelle et comment déterminer quel est ce nombre après cela ?

Sous-tâche 2: Algorithme optimal (90 points)

Développe un algorithme capable de résoudre le problème optimalement. Le nombre n t'est donné et tu dois suivre certaines quêtes pour découvrir quel est le nombre manquant. Note que tu peux adapter ta stratégie selon les résultats des quêtes.

Optimise d'abord le nombre de quêtes. Donne la formule exacte du nombre de quêtes dont tu as besoin dans le pire des cas (par exemple $2^n + n + 13$ ou $\frac{1}{2}n \cdot (n-1)$). Optimise ensuite le temps d'exécution (asymptotique), puis pour l'utilisation de mémoire (asymptotique).

Binna à vélo

Mouse Binna vient d'arriver à Singapour pour l'IOI 2020. Elle est actuellement à son hôtel et aimerait aller à la cérémonie d'ouverture à vélo. La souris Binna loue un Binnabike pour voyager à travers la ville. Il fonctionne comme suit :

— Comme Singapour veut que plus de gens utilisent des transports respectueux de l'environnement, le gouvernement a décidé que les premières d minutes à vélo depuis n'importe quelle station seront toujours gratuites - cela peut être fait plusieurs fois par jour !
— Chaque minute après les premières d minutes, un dollar doit être payé par minute.
— Le vélo doit toujours être ramené à une station officielle.

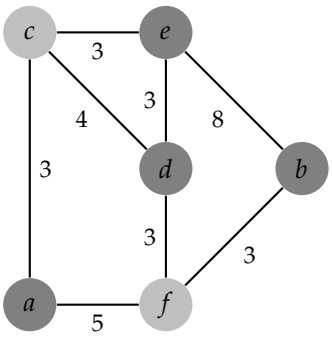
À Singapour, il y a n carrefours reliés par$m$ rues. Pour chaque rue, tu sais combien de minutes Binna prendrait pour la suivre de bout en bout. ce temps t'est donné comme un nombre entier positif. Binna sait aussi que k de ces carrefours sont équipés de stations Binnabike où des vélos peuvent être empruntés et rendus.

Binna découvre que la station a est la plus proche de son hôtel et la station b la plus proche de la cérémonie d'ouverture. Comme celle-ci commence au temps t, Binna veut arriver à la station b avant t. Additionnellement, elle souhaite dépenser le moins d'argent possible en chemin. Aide-la !

Description formelle On te donne un graphe avec n sommets et m arêtes. Les arêtes ont pour poids des entiers positifs. k des sommets sont des "stations". Il y a un paramètre de réduction d. Si le temps de trajet entre les stations u et v est w, le coût égale $\max(w - d, 0)$, et cette réduction peut être utilisée plusieurs fois. Trouve le coût minimal pour un voyage de a à b de manière à ce que le temps nécessaire soit au plus de t.

Limites Pour optimiser ton algorihtme, suppose que n et t sont très grands et d et k plutôt petits. En particulier, tu peux supposer que $d \cdot k$ (et donc d et k aussi) est plus petit que t et que k^3 est plus petit que n.Les poids des arêtes sont des nombres entiers positifs qui peuvent être arbitrairement grands.

Sous-tâche 1: Un exemple (5 points)



Donnez le chemin que Binna devrait prendre afin de minimiser le coût de son voyage pour le graphe ci-dessus, avec $d = 3$ et $t = 14$. Les $k = 4$ stations sont a, b, d et e. Donnez le montant qu'elle doit dépenser. Aucune justification n'est nécessaire.

Sous-tâche 2: Durée illimitée (35 points)

Comme Binna pense que la cérémonie d'ouverture est à peu près la même chaque année, elle pense que c'est acceptable d'être en retard. Construisez un algorithme qui calcule le coût minimum étant donné que Binna a un temps illimité.

Sous-tâche 3: Cas général (60 points)

Malheureusement, le leader suisse a ordonné à Binna de le rencontrer lors de la cérémonie d'ouverture avant que celle-ci ne commence. Binna devra donc arriver encore plus tôt que prévu. Il n'y a plus de restrictions sur quoi que ce soit. Décrit ton algorithme.

Venise

Malheureusement, Venise est engloutie en raison de la montée des eaux et il est désormais impossible de la visiter. Stofl a toujours voulu la voir une fois mais n'en a jamais eu l'occasion, donc la nouvelle l'a grandement attristé. Binna a donc commencé à recréer la ville. Elle a déjà construit n monuments et creusé des canaux entre ces monuments. Vu la pénibilité de la réalisation de ces canaux, elle a creusé exactement assez de canaux pour que tous les monuments soient reliés mais pas un de plus.

Les gondoles sont la chose la plus importante à Venise. Chaque gondole peut faire des allers et retours entre deux monuments et utilise pour ce faire le chemin direct et s'arrête auprès de chaque monument sur le chemin.

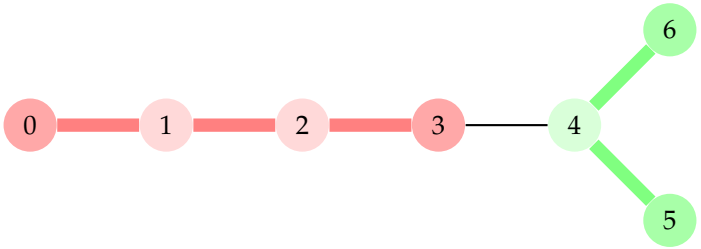
La construction des gondoles est un procédé complexe et Binna aimerait donc établir des routes de façon à utiliser aussi peu de gondoles que possible en s'assurant qu'il n'y ait pas de monument auprès duquel aucune gondole ne s'arrête. Aide-la à déterminer combien de gondoles sont nécessaires et quelles routes établir.

Description formelle Le graphe dont les sommets sont les monuments et les arêtes les canaux t'est donné. Il est garanti que ce graphe est un arbre. Cela signifie que pour n'importe quels monuments a et b, il y a exactement un chemin entre a et b gibt.

Une répartition des gondoles est une liste de paires (a_i, b_i). Cela signifie que la gondole i fait des allers et retours entre le monument a_i et le monument b_i. Une répartition des gondoles est valide s'il n'y a aucun moment qu'aucune gondole ne visite. Une répartition des gondoles est optimal si elle est valide et qu'il n'y a aucune répartition des gondoles valide qui utilise moins de gondoles.

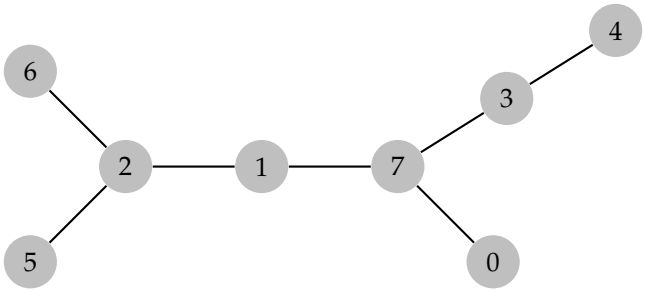
Exemple $n = 7$ et la liste des canaux est $\{(0,1),(1,2),(2,3),(3,4),(4,5),(4,6)\}$. Une solution est d'utiliser une gondole entre les monuments 0 et 3 et une autre entre 5 et 6.

Un planning optimal pour cet exemple est dépeint ci-dessous :



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

Il y a huit monuments à Venise, reliés par des canaux comme illustré ci-dessous. Trouve une répartition optimale des gondoles et prouve par des arguments qu'elle est en effet optimale.



Sous-tâche 2: Plus de gondoles (25 points)

Binna est contente si elle trouve une répartition des gondoles qui est presque optimale. Plus précisément, si une répartition optimale des gondoles en utilise k, elle aimerait trouver une répartition des gondoles qui en utilise au plus $2k$. Développe un algorithme qui trouve une telle répartition et prouve par des arguments qu'il est correct.

Sous-tâche 3: Répartition optimale (65 points)

Construire des gondoles n'est pas aussi aisé que Binna l'espérait. C'est pourquoi elle n'est finalement satisfaite que par une répartition optimale des gondoles. Développe un algorithme qui trouve une répartition optimale des gondoles et prouve par des arguments qu'il est correct.