Olympiades Régionales d'Informatique Franco-Australiennes

10–12 Mars, 2006

Durée: 4 heures

4 questions

Problème 1
Gardes

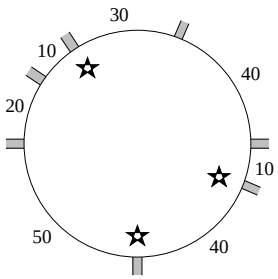
Limites de temps et de mémoire : 3 secondes, 30 Mo

C'est l'heure des Olympiades annuelles de lancer de Boomerang et de Croissant, et les hôtes ont dépensé sans compter. Une grande partie de la ville a été bouclée entièrement pour accueillir les athlètes et les journalistes, sans oublier les responsables des barbecues et les chefs pâtissiers Français. Avec toute cette agitation, vous vous êtes retrouvés responsable de la sécurité de l'événement.

Chaque route qui arrive dans la zone réservée à l'Olympiade a été condamnée, pour assurer qu'il n'y ait pas d'intrus indésirables. Cependant, les hôtes ont dépensé tellement pour les cérémonies d'ouverture et de clôture qu'il ne reste plus assez d'argent pour embaucher assez de gardes de sécurité. Votre but est d'embaucher aussi peu de gardes que possible, tout en garantissant la sécurité de la zone.

La zone des Olympiades peut être vue comme une grande région circulaire, avec des routes qui arrivent à différents points du contour. Le gouvernement requiert que tout point où une route entre dans la zone soit au maximum à k mètres d'un agent de sécurité, la distance étant mesurée le long du cercle. Les gardes ne peuvent être placés que là où arrivent des routes, ils ne peuvent jamais se trouver dans les zones se trouvant entre ces positions.

Un exemple est donné dans le diagramme ci-dessous. Sept routes arrivent dans la zone, les distances entre-elles étant indiquées sur le diagramme. Supposez que le gouvernement ait déclaré que $k = 30$. Dans ce cas, en plaçant trois gardes aux routes marquées par des étoiles, vous pouvez vous assurer que toute route est gardée.



Votre objectif est, étant donné les distances entre les routes et la valeur de k, de déterminer le plus petit nombre de gardes nécessaire.

Contraintes

- $1 \leq n \leq 1\,000\,000$, où n est le nombre de routes qui arrivent sur la zone des Olympiades;
- $1 \leq d \leq 1\,000$, où d est la distance entre deux routes adjacentes;
- $1 \leq k \leq 10\,000\,000$, où k est la valeur donnée par le gouvernement, comme décrit plus haut.

De plus, 30% des points attribués sont consacrés à des tests pour lesquels le nombre de routes satisfait $1 \leq n \leq 1\,000$.

Entrée

Votre programme doit lire sur l'entrée standard. La première ligne contiendra les entiers n et k, séparés par un espace, comme décrit plus haut.

Les n lignes suivantes décrivent chacune la distance entre deux routes adjacentes. Ces distances seront données dans l'ordre de parcours autour du cercle.

Plus précisément, supposons que les routes sont numérotées $1, 2, \ldots, n$ dans l'ordre des aiguilles d'une montre autour du cercle. Alors la $i$ème de ces lignes sera la distance, selon l'ordre des aiguilles d'une montre, entre la $i$ème et la $(i+1)$ème route, la $n$ème ligne donnant la distance entre la $n$ème et la première route.

Sortie

Votre programme doit afficher une seule ligne sur la sortie standard. Cette ligne doit contenir un simple entier, qui donne le plus petit nombre d'agents de sécurité requis.

Exemple d'entrée

```
7 30
30
40
10
40
50
20
10
```

Exemple de sortie

```
3
```

Score

Le score pour chaque test d'entrée sera de 100% si la bonne réponse est affichée, et de 0% sinon.

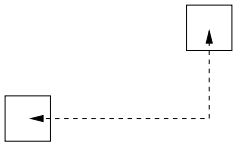
Problème 2
Fuir la Civilisation

Limites de temps et de mémoire : 3 secondes, 30 Mo

Après trop d'années passées en tant que programmeur, vous décidez qu'il est temps de prendre votre retraite et de renouer avec la nature. Vous empaquetez toutes vos possessions non-technologiques (c'est à dire pas grand chose) et vous éloignez de la ville, d'Internet, de la civilisation et allez vers la campagne pour vivre une vie simple.

Malheureusement, avant de partir, vous devez écrire un dernier programme. Vous souhaitez aller aussi loin que possible de la civilisation — dans l'une des zones les plus isolées qui soient connues de l'humanité. Vous possédez une carte de largeur w et de hauteur h. Cette carte vous indique où les zones de civilisation peuvent être trouvées.

Les zones les plus isolées sont les carrés qui sont à la distance Manhattan la plus éloignée de toute civilisation. La distance Manhattan entre deux carrés est la somme des distances haut/bas et gauche/droite, comme illustré ci-dessous.



Votre objectif est de déterminer à quelle distance se trouvent les zones les plus isolées de la civilisation.

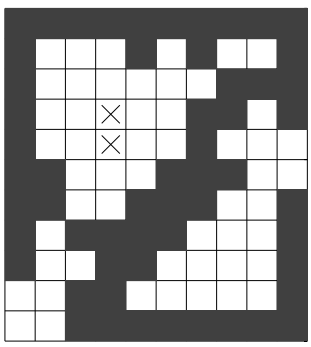
Contraintes

- $1 \leq w, h \leq 1\,000$, où w est la largeur de la carte, et h la hauteur;
- on vous garantit qu'il y aura toujours au moins un carré de civilisation, et un carré sans civilisation.

De plus, 30% des points seront attribués à des tests dans lesquels les dimensions de la carte satisfont $w, h \leq 50$.

Exemple

Considérez la carte ci-dessous, de largeur 10 et hauteur 11. Les zones de civilisation sont noircies. Les deux carrés marqués avec un $\times$ sont les carrés les plus isolés : aucun autre carré de la carte ne se situe plus loin de la civilisation que ceux-ci. Les carrés marqués ont chacun une distance manhattan de 3 par rapport à la civilisation la plus proche, donc la réponse au problème est 3.



Entrée

Votre programme doit lire sur l'entrée standard. La première ligne contiendra deux entiers positifs w et h, séparés par un espace. Les h lignes suivantes contiendront chacune w entiers. Chacun de ces entiers sera soit un 1, indiquant la présence de civilisation, soit un 0, indiquant l'absence de civilisation.

Sortie

Votre programme doit écrire une seule ligne sur la sortie standard. Cette ligne doit contenir un simple entier, donnant la distance Manhattan de la civilisation au carré le plus isolé de la carte.

Exemple d'entrée

```
10 11
1 1 1 1 1 1 1 1 1 1
1 0 0 0 1 0 1 0 0 1
1 0 0 0 0 0 0 1 1 1
1 0 0 0 0 0 1 1 0 1
1 0 0 0 0 0 1 0 0 0
1 1 0 0 0 1 1 1 0 0
1 1 0 0 1 1 1 0 0 1
1 0 1 1 1 1 0 0 0 1
1 0 0 1 1 0 0 0 0 1
0 0 1 1 0 0 0 0 0 1
0 0 1 1 1 1 1 1 1 1
```

Exemple de sortie

```
3
```

Score

Le score pour chaque test d'entrée sera de 100% si la bonne réponse est affichée, ou 0% sinon.

Problème 3
Ceintures

Limites de temps et de mémoire : 0.1 seconde, 30 Mo

C'est la semaine de la mode, et vous essayez les dernières ceintures design de Paris. Malheureusement, aucune des ceintures n'est à votre taille, ce qui indique clairement une chose : vous avez besoin de faire plus d'exercice.

Vous décidez que le meilleur moment pour faire de l'exercice est lors de chacun de vos trajets de retour à la maison en tramway. Le tramway va en ligne droite de l'école à la maison, et la ligne contient plusieurs arrêts. Vous pouvez faire de l'exercice en descendant du tramway à un arrêt, marchant vers un autre stop plus loin le long de la ligne, puis remontant dans le prochain tramway qui passe par là. Vous pouvez faire ceci autant de fois que vous souhaitez (utiliser le tramway, marcher, utiliser le tramway, et ainsi de suite). Vous pouvez choisir de commencer le trajet en marchant, et vous pouvez décider de descendre d'un tramway à un arrêt, et finir le trajet à pied.

Les tramways passent toutes les t minutes, et le premier tramway quitte votre école précisément quand vous commencez votre trajet. Les tramways se déplacent à vitesse fixe, et vous marchez également à une vitesse fixe, plus lente. Vous ne pouvez marcher que vers chez vous (vous ne pouvez pas retourner vers l'cole ni tourner en rond). Si vous marchez de l'arrêt a à l'arrêt b et attrapez le tramway suivant, vous attendez simplement à l'arrêt b et écoutez votre lecteur de mp3 jusqu'à-ce que le prochain tramway passe. Si vous arrivez à l'arrêt b précisément au moment où un tramway passe, vous pouvez monter à bord.

Après avoir parcouru quelques magazines de santé, vous décidez que vous avez besoin de marcher au moins k mètres lors de vorte trajet. Votre objectif est de trouver le temps de trajet le plus petit possible, allant du début à la fin et qui respecte les conditions décrites plus haut.

Contraintes

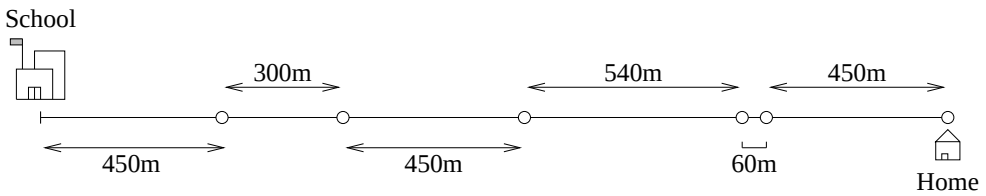
Tous les temps ci-dessous sont mesurés en millisecondes et toutes les distances sont mesurées en mètres.

- $30\,000 \leq t \leq 1\,000\,000$, où t est le temps entre chaque tramway et le suivant;
- $1 \leq m_t < m_w \leq 100$, où m_t est le temps nécessaire à un tramway pour avancer d'un mètre et m_w est le temps dont vous avez besoin pour marcher d'un mètre;
- $1 \leq k \leq 10\,000$, où k est la distance minimale que vous devez marcher;
- $1 \leq s \leq 100$, où s est le nombre total d'arrêts de tramway (sans compter l'école où votre trajet commence);
- Il n'y a jamais deux arrêts séparés de plus de 1 000 mètres;
- Le dernier arrêt est au moins à k mètres de l'école (i.e., il est possible de marcher assez pour obtenir l'exercice dont vous avez besoin).

De plus, 30% des points seront attribués à des tests dans lesquels la distance de marche minimale satisfait $k \leq 2\,000$.

Exemple

Considérez la ligne de tramway illustrée ci-dessous. Elle commence par l'école du côté gauche, après laquelle se trouvent $s = 6$ arrêts. Les distances entre les arrêts consécutifs sont indiquées sur le diagramme, et votre maison se trouve au tout dernier arrêt.



Supposez que la gestion du tramway soit très performante et que les tramway passent toutes les 30 secondes, i.e., toutes les $t = 30\,000$ millisecondes. Les tramways prennent $m_t = 1$ milliseconde pour se déplacer d'un mètre et il ne vous faut que $m_w = 100$ millisecondes pour marcher un mètre. Votre régime minceur vous impose de marcher au moins $k = 870$ mètres durant votre trajet.

Une stratégie optimale pour rentrer chez vous est la suivante :

Action	Arrêts	Distance	Temps
Utiliser le tramway	Ecole–1	450m	450ms
Marcher	1–2	300m	30 000ms
Attendre le tramway	2		300ms
Utiliser le tramway	2–3	450m	450ms
Marcher	3–5	600m	60 000ms
Attendre le tramway	5		600ms
Utiliser le tramway	5–6	450m	450ms
			92 250ms

Votre distance totale de marche selon ce plan est de 900m, ce qui est suffisant car c'est plus que les $k = 870$m requis. Le temps de trajet total est de 92 250 millisecondes.

Entrée

Votre programme doit lire sur l'entrée standard. La première ligne contiendra un simple entier t, qui donne le temps entre chaque tramway et le suivant. La deuxième ligne contiendra les deux entiers m_t et m_w séparés par un espace, donnant le temps nécessaire à un tramway pour se déplacer d'un mètre, et le temps qu'il vous faut pour marcher un mètre. La troisième ligne contiendra un simple entier k, donnant la distance minimale que vous avez besoin de marcher.

La quatrième ligne de l'entrée contiendra l'entier s, donnant le nombre total d'arrêts de tramway (à l'exception de l'école où le trajet commence). Les s lignes suivantes donnent les positions des arrêts individuels dans l'ordre, de l'école à votre maison. La $i$ème de ces lignes contiendra l'entier d_i, donnant la distance de l'école au $i$ème arrêt. Votre maison est au dernier de ces arrêts.

Une fois de plus, tous les temps sont mesurés en millisecondes et toutes les distances sont mesurées en mètres.

Sortie

Votre programme doit écrire une simple ligne sur la sortie standard, contenant un entier : la durée totale de trajet la plus courte possible, mesurée en millisecondes.

Exemple d'entrée

```
30000
1 100
870
6
450
750
1200
1740
1800
2250
```

Exemple de sortie

```
92250
```

Score

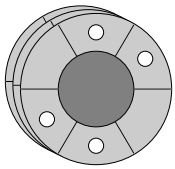
Le score pour chaque test d'entrée sera de 100% si la bonne réponse est fournie, et 0% sinon.

Problème 4
Les roues du destin

Limites de temps et de mémoire : 3 secondes, 30 Mo

C'est l'heure de votre visite hebdomadaire chez la voyante. La semaine dernière elle a lu dans les lignes de votre main; cette semaine, elle utilise les Roues du Destin.

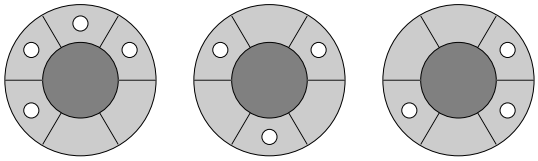
Les Roues du Destin consistent en un jeu de w roues circulaires, placées l'une derrière l'autre comme illustré ci-dessous. Le contour de chaque roue est divisé en s segments identiques, dont certains sont percés d'un trou.



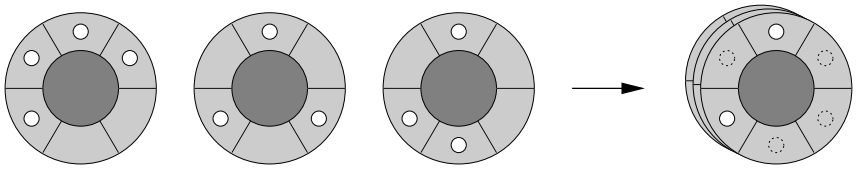
Votre voyante va vous demander de faire tourner chaque roue l'une après l'autre. Lorsque les roues s'arrêteront de tourner, elle cherchera des trous qui sont alignés sur toute la longueur, de l'avant à l'arrière (de telle sorte que vous puissiez voir au travers de l'ensemble des roues). Pour chaque ensemble de trous alignés, on vous garantit une année de prospérité et de code sans bug.

Heureusement, vous avez utilisé ce système de nombreuses fois, et êtes capables de faire tourner chaque roue exactement comme vous le souhaitez. Votre but est de décider jusqu'où faire tourner chaque roue, pour qu'un maximum de groupes de trous puisse être aligné.

Par exemple, considérez l'ensemble de $w = 3$ roues illustré ci-dessous, chacun étant divisé en $s = 6$ segments.



En faisant tourner la deuxième roue d'un segment dans le sens des aiguilles d'une montre, et en faisant tourner la troisième roue de deux segments dans ce même sens, vous pouvez placer les trois roues comme indiqué ci-dessous. Lorsque les roues sont placées l'une derrière l'autre, il y a deux ensembles de trous alignés sur toute la longueur, de l'avant à l'arrière. Ceci est illustré sur le dernier diagramme, du côté droit.



On ne vous demande pas de donner la meilleure solution existante. Votre but est d'aligner autant de groupes de trous que possible. Votre programme va alors affronter d'autres programmes, les meilleures solutions obtenant plus de points.

Contraintes

- $1 \leq w \leq 50$, où w est le nombre total de roues;
- $1 \leq s \leq 50$, où s est le nombre total de segments divisant le contour de chaque roue.

De plus, 30% des points sont associés à des tests pour lesquels les segments satisfont $s \leq 20$.

Entrée

Votre programme doit lire sur l'entrée standard. La première ligne contiendra les deux entiers w et s séparés par un espace, comme décrit plus haut.

Les w lignes qui suivent décrivent chacune une roue. Chaque ligne contient s entiers séparés par des espaces, décrivant les segments de la roue dans l'ordre des aiguilles d'une montre, en commençant en haut. Pour chaque segment, l'entier 0 représente un trou dans la roue et l'entier 1 représente l'absence de trou.

Sortie

Votre programme doit écrire $w + 1$ lignes sur la sortie standard. Parmi les w premières lignes, la $i$ème ligne correspond à la $i$ème roue des données d'entrée, et doit contenir un simple entier qui décrit le nombre de segments dont la roue doit tourner, dans le sens des aiguilles d'une montre. Chaque entier affiché doit être entre 0 et $s - 1$ inclus.

Une fois que ces w lignes ont été écrites, votre programme doit écrire une ligne supplémentaire sur la sortie. Cette ligne doit contenir un seul entier : le nombre total de groupes de trous qui sont alignés sur toute la longueur, de l'avant à l'arrière.

Exemple d'entrée

```
3 6
0 0 1 1 0 0
1 0 1 0 1 0
1 0 0 1 0 1
```

Exemple de sortie

```
0
1
2
2
```

Score

Il n'y a pas de "meilleure solution" particulière, que l'on vous demande d'obtenir. Votre score sera en effet déterminé relativement aux solutions des autres candidats contre lesquels vous concourrez (ainsi que la solution des juges).

Une solution simple consiste à examiner chaque roue tour tour du haut en bas, et à chaque étape, la faire tourner pour aligner autant de trous que possible avec toutes les roues au dessus d'elle. Une telle solution est définie comme obtenant 30% des points.

Pour chaque scénario d'entrée, le candidat qui aligne le plus de groupes de trous sera identifié. Supposons que ce candidat aligne h ensembles de trous au total. Votre score pour ce scénario d'entrée sera alors :

- 100% si votre programme trouve une solution qui a également h groupes de trous alignés;
- 30% si votre programme trouve une solution égale à la solution simple décrite ci-dessus;
- 0% si votre programme génère une solution incorrecte (i.e., vous calculez incorrectement le nombre de groupes de trous qui sont alignés, ou vous ne suivez pas le format de sortie précisément);
- sinon, il sera déterminé selon une échelle linéaire, en fonction du nombre de groupes de trous que votre programme aligne, les scores de 100% et 30% sur l'échelle, correspondant aux solutions décrites ci-dessus.

Par exemple, considérez un scénario d'entrée pour lequel la solution simple aligne 26 groupes de trous. Si la meilleure solution trouvée par un candidat (ou les juges) aligne 40 ensembles de trous, alors l'échelle de score pour une solution *correcte* sera la suivante :

Trous alignés	40	36	32	28	26	24	20	16
Score	100%	80%	60%	40%	30%	20%	0%	0%