|  |  |
| --- | --- |
| **n1** | **Un jeu à la Tetris (+++)** |
| **2** | **Rallonges audio (+++)** |
| **3** | **Snake (++)** |
| **4** | **Cartes perforées (++)** |
| **5** | **Coffre-fort (++)** |
| **6** | **Banderole (++)** |
| **7** | **Billard (+++)** |
| **8** | **Fragments d'ADN (+++)** |

**Un jeu à la Tetris**

**PROBLEME**

Vous devez simuler un jeu à la TetrisTM. Dans cette variante, toutes les formes sont des bâtons d'épaisseur 1 et de longueur variable. Le cadre du jeu est un rectangle infiniment haut, et des bâtons tombent du ciel les uns après les autres.

On vous dit pour chaque bâton s'il est orienté horizontalement ou verticalement, on vous donne sa longueur, et on vous indique enfin sa position latérale. Le bâton tombe alors tout droit vers le bas, sans se retourner d'aucune manière, et s'arrête dès qu'il touche le sol ou bien repose sur un bâton déjà placé.

Voici par exemple d'état d'un jeu, après que les bâtons numérotés de 'a' à 'h' sont tombés :

| |

| |

| f |

| f |

|g hhhh f |

|g eeeeeeeee|

|g b |

|g b ddddd|

|g b c |

|aaaaa c |

-------------

Supposons qu'un bâton 'i' de longueur 7 tombe verticalement dans la seconde colonne, alors la nouvelle situation est :

| |

| |

| i f |

| i f |

|gi hhhh f |

|gieeeeeeeee|

|gi b |

|gi b ddddd|

|gi b c |

|aaaaa c |

-------------

On part d'un cadre de jeu initialement tout vide, et on vous donne la suite des bâtons qui tombent. Votre tâche est de calculer la hauteur maximale atteinte par les bâtons, c'est-à-dire la hauteur de la case la plus haute contenant un morceau de bâton. Dans l'exemple ci-dessus, le résultat est 8, et est atteint à la fois par le bâton 'i' et par le bâton 'f'.

**LIMITES DE TEMPS ET DE MEMOIRE**

* Temps : 1 s sur une machine à 1Ghz.
* Mémoire : 64000 Ko.

**CONTRAINTES**

Le cadre du jeu contiendra toujours 100 colonnes, numérotées de 0 à 99 inclus. Il n'y a *a priori* pas de limite au nombre de lignes.

**ENTREE**

La première ligne de l'entrée contient un entier *N* : le nombre de bâtons qui vont tomber, avec 1 <= *N* <= 300.

Chacune des *N* lignes suivantes contient la description d'un bâton.

Un bâton est décrit par une lettre puis deux entiers, séparés par des espaces :

* une lettre qui donne l'orientation du bâton : 'H' pour horizontale, 'V' pour verticale.
* un entier qui donne la longueur *L*i du bâton, avec 1 <= *L*i <= 100.
* un entier qui donne la colonne *C*i du bord le plus à gauche du bâton, avec 0 <= *C*i <= 99.

Remarque à propos du bord le plus à gauche du bâton :

* Si le ième bâton est vertical, *C*i donne l'indice de la colonne dans laquelle tombe le bâton.
* Si le ième bâton est horizontal, le bâton s'étend entre les colonnes d'indice *C*i et *C*i + *L*i - 1 incluses.

**SORTIE**

Affichez un unique entier : la hauteur maximale atteinte.

**EXEMPLE(S) D'ENTREE / SORTIE**

*Exemple 1 :*

en entrée ...

|  |
| --- |
| 9 H 5 0 V 3 4 V 2 7 H 5 6 H 9 2 V 3 9  V 5 0 H 4 3 V 7 1 |

en sortie ...

|  |
| --- |
| 8 |

**Rallonges audio**

**PROBLEME**

|  |  |
| --- | --- |
| Vous souhaitez raccorder votre nouvel ordinateur à l'amplificateur de votre super chaîne hi-fi afin de pouvoir mettre à fond l'album "Wish You Were Here" (du groupe Pink Floyd). Vous espérez que la mélodie des guitares électriques fera venir en vous l'inspiration nécessaire pour résoudre les sujets du concours France-IOI auquel vous êtes en train de participer. | D:\Cours\Cours IGA\IGA [2009-2010]\Olympiades Informatique\olymp-info\info\www.france-ioi.org\data\sujets\rallonges_audio\rallonges_audio2.png |

En même temps, vous souhaitez vous rapprocher le plus possible de la borne wifi par laquelle vous accédez à Internet afin de pouvoir télécharger tous les autres albums des Pink Floyd avec un débit maximal (télécharger légalement, bien sûr). Cette borne wifi est commune à votre immeuble et peut théoriquement vous fournir une très bonne bande passante. Malheureusement, elle se situe un peu loin de chez vous et cela est un facteur limitant votre débit. Vous avez déjà remarqué qu'il vous était possible d'améliorer le débit rien qu'en vous déplaçant de quelques mètres en direction de la borne wifi.

Votre but va donc être de combiner les rallonges audio dont vous disposez afin d'obtenir un maximum de longueur, et pouvoir ainsi écouter de la musique tout en étant aussi près que possible de la borne wifi.

Vous avez à votre disposition un certain nombre de rallonges. Chaque rallonge est caractérisée par sa longueur, et par la nature des connecteurs à ses deux extrémités : chaque connecteur peut être soit mâle, soit femelle. Remarque technique : on ne peut pas connecter des fiches mâles entre elles, ni deux fiches femelles entre elles. Notez enfin que la carte son de votre ordinateur ainsi que l'amplificateur de votre chaîne hi-fi portent tous deux des connecteurs femelles (il faut donc arriver dessus avec des connecteurs mâles).

**LIMITES DE TEMPS ET DE MEMOIRE**

* Temps : 1 s sur une machine à 1Ghz.
* Mémoire : 16000 Ko.

**CONTRAINTES**

* 1 <= *N* <= 1 000, où *N* est le nombre de rallonges.
* 1 <= *L* <= 100 000, où *L* est la longueur d'une rallonge (en millimètres).

**ENTREE**

* La première ligne de l'entrée contient l'entier *N*.
* Chacune des *N* lignes suivantes contient trois entiers séparés par des espaces : *C1*, *C2* et *L*. *C1* et *C2* décrivent les connecteurs (1 pour mâle, 0 pour femelle), et *L* donne la longueur de la rallonge.

Notez que le signal peut passer dans une rallonge dans le sens que l'on veut. Ainsi le signal peut traverser la rallonge de C1 vers C2 ou de C2 vers C1.

**SORTIE**

Vous devez écrire un seul entier sur la sortie : la distance maximale à laquelle vous pouvez placer votre ordinateur de la chaîne hi-fi tout en lui envoyant le signal musical à travers les rallonges. S'il est impossible de relier les deux appareils avec les rallonges dont vous disposez, affichez -1.

**EXEMPLE(S) D'ENTREE / SORTIE**

*Exemple 1 :*

en entrée ...

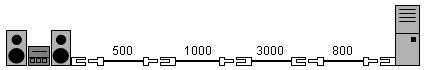
|  |
| --- |
| 5 1 0 1000 1 1 500 0 0 2000 1 1 800 0 0 3000 |

en sortie ...

|  |
| --- |
| 5300 |
|  |

**COMMENTAIRES**

Utilisez la séquence suivante par exemple :



Notez que la seconde rallonge a été retournée par rapport à sa description dans l'entrée. La longueur totale obtenue est 500+1000+3000+800=5300.

**Snake**

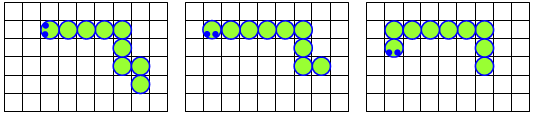
**PROBLEME**

Vous connaissez sûrement le jeu Snake, datant des années 70, mais revenu à la mode récemment sur de nombreux téléphones portables : un serpent se balade sur un terrain rectangulaire et doit manger divers aliments sans jamais se mordre la queue ni sortir du terrain, sachant qu'il s'allonge à chaque fois qu'il mange quelque-chose. Nous vous demandons tout simplement de programmer une partie d'une version simplifiée de ce jeu.

Le jeu se déroule sur une grille rectangulaire. Chaque case peut contenir une certaine quantité de nourriture, auquel cas on marque avec un nombre entier cette quantité, ou bien la case est vide auquel cas on la marque avec 0. Lorsque le serpent passe sur une case contenant de la nourriture, il la mange entièrement (la case sera donc vide après son passage), et il aura ainsi acccumulé de l'énergie qui lui permettra de grandir.

Au début, le serpent est de longueur un (il occupe donc exactement une case de la grille), il se trouve sur la case tout en haut à gauche, et sa tête regarde vers la droite. Cette case ne contient jamais de nourriture. À chaque tour de jeu (sauf au premier), le serpent va bouger. Soit il a de l'énergie en stock et dans ce cas il va s'allonger en avançant sa tête, ce qui consomme une unité d'énergie, soit il n'a pas d'énergie en stock et avance tout son corps d'une case, sans changer de longueur. Dans les deux cas, la tête du serpent avance d'exactement une case dans la direction où il regarde. A la fin de chaque tour, une fois le déplacement effectué, le joueur a la possibilité de faire tourner le regard du serpent d'un quart de tour vers la gauche ou vers la droite, et modifier ainsi la direction du déplacement suivant. On peut faire tourner le regard du serpent aussi à la fin du premier tour, bien qu'il n'y ait pas de déplacement.

L'illustration suivante montre deux déplacements successifs possibles d'un serpent de longueur 9.



Une fois que le serpent s'est déplacé, si sa tête se trouve sur la même case qu'une autre partie de son corps, il se mord la queue, et meurt. Le serpent meurt également si sa tête est en dehors de la grille. Si par contre sa tête se trouve sur une case contenant de la nourriture (un nombre positif), il la mange et accumule ainsi de l'énergie, qu'il utilisera aux tours suivants pour grandir. Notez qu'il n'y a pas de limite à la quantité d'énergie que le serpent peut accumuler.

On vous fournit une liste d'actions du joueur, en vous indiquant, pour chacune, à quel tour elle se produit, et le type d'action, qui peut consister à tourner le regard d'un quart de tour soit vers la gauche, soit vers la droite. Votre but est de déterminer à quel tour le serpent va mourir. Il finit en effet par mourir quoi qu'il arrive, car lorsqu'il n'y a plus d'actions du joueur, il finit nécessairement par sortir de la grille quelques tours plus tard.

**LIMITES DE TEMPS ET DE MEMOIRE**

* Temps : 1 s sur une machine à 1Ghz.
* Mémoire : 16000 Ko.

**CONTRAINTES**

* 1 <= *H*, *L* <= 100, où *H* et *L* sont respectivement la hauteur et la largeur de la grille.
* 0 <= *N* <= 100, où *N* est la quantité de nourriture contenue dans une case de la grille.
* 1 <= *A*, *T* <= 100 000, où *A* est le nombre d'actions effectuées par le joueur, et *T* est le numéro du tour auquel est faite une action.

De plus, dans 30% des tests, on a 1 <= *H*, *L* <= 10.

**ENTREE**

La première ligne de l'entrée contient deux entiers *H* et *L* séparés par un espace : la hauteur et la largeur de la grille.

Chacune des *H* lignes suivantes décrit une rangée de la grille et contient *L* entiers séparés par des espaces : les valeurs de cette rangée dans la grille, de gauche à droite.

La ligne suivante contient un entier *A* : le nombre d'actions effectuées par le joueur pendant la partie.

Chacune des *A* lignes suivantes contient deux entiers séparés par un espace. Le premier entier indique le numéro du tour auquel le joueur effectue une action (le premier tour a pour numéro 1). Le deuxième entier est soit -1 si le joueur tourne le regard du serpent vers la gauche, soit 1 s'il tourne le regard du serpent vers la droite à la fin de ce tour. Les actions sont données dans l'ordre dans lequel elles sont effecutées.

**SORTIE**

Vous devez afficher un simple entier sur la sortie : le numéro du tour auquel le serpent meurt.

**EXEMPLE(S) D'ENTREE / SORTIE**

*Exemple 1 :*   
en entrée ...

|  |
| --- |
| 4 5 0 4 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 6 4 1 6 1 7 1 9 -1 10 -1 11 -1 |

en sortie ...

|  |
| --- |
| 12 |
|  |

*Exemple 2 :*   
en entrée ...

|  |
| --- |
| 5 4 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 1 1 1 |

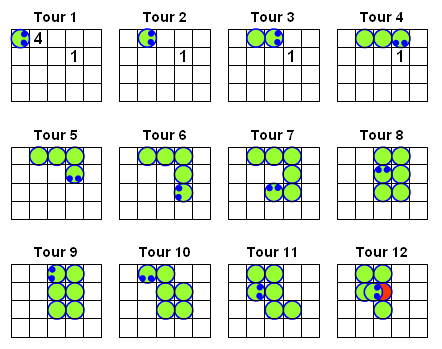
en sortie ...

|  |
| --- |
| 6 |
|  |

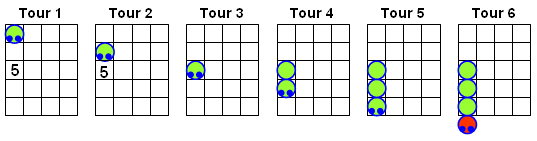
**COMMENTAIRES**

**Conseil :** pour bien vérifier que votre programme fonctionne, pensez à afficher l'état du jeu à la fin de chaque tour. (mais désactivez cet affichage avant de soumettre votre solution).

Voici les différentes étapes de la partie correspondant au premier exemple d'entrée. Elles représentent l'état du jeu à la fin de chaque tour :



Ci-dessous, les différentes étapes de la partie correspondant au deuxième exemple d'entrée :



**Cartes perforées**

**PROBLEME**

|  |  |
| --- | --- |
| Vous avez retrouvé un stock de cartes perforées, que vous avez toutes scannées. Chaque carte contient un message constitué de lettres, codé dans une grille à 26 colonnes. Votre but va être de retrouver ce message. | D:\Cours\Cours IGA\IGA [2009-2010]\Olympiades Informatique\olymp-info\info\www.france-ioi.org\data\sujets\cartes_perforees\carte_perforee.png |

Une carte perforée contient une grille, et chaque colonne de la grille correspond à une lettre de l'alphabet. L'encodage du message sur la carte a été effectué à la main à l'aide du procédé suivant :

* Les lettres du mot sont encodées une par une, dans l'ordre, chacune à l'aide d'un trou sur la carte. Les règles suivantes expliquent comment trouver la colonne et la ligne à laquelle faire le trou.
* La colonne dans laquelle on fait le trou est celle qui correspondant à la lettre. Pour A, on fait donc un trou à la première colonne, pour B à la seconde, pour Z à la dernière.
* Maintenant qu'on connaît la colonne, il faut trouver sur quelle ligne faire le trou. La règle est qu'on choisit la ligne où faire le trou de telle sorte :
  + qu'il n'y ait aucun trou déjà réalisé sur cette même ligne à droite de la colonne que l'on a sélectionnée,
  + qu'il n'y ait aucun trou sur les lignes d'en-dessous,
  + qu'on prenne la ligne la plus haute possible parmi celles qui respectent ces contraintes, de sorte à économiser du papier.

Noter que ce système empêche de faire deux trous au même endroit.

Les cartes sont représentées dans un format texte. Les # représentent les cases non trouées, les lettres 'O' en majuscule représentent les trous. Voici par exemple les étapes de l'encodage du mot LETTRES :

ABCDEFGHIJKLMNOPQRSTUVWXYZ

###########O##############

Lettres placées: L

ABCDEFGHIJKLMNOPQRSTUVWXYZ

###########O##############

####O#####################

Lettres placées: LE

ABCDEFGHIJKLMNOPQRSTUVWXYZ

###########O##############

####O##############O######

Lettres placées: LET

ABCDEFGHIJKLMNOPQRSTUVWXYZ

###########O##############

####O##############O######

###################O######

Lettres placées: LETT

ABCDEFGHIJKLMNOPQRSTUVWXYZ

###########O##############

####O##############O######

###################O######

#################O########

Lettres placées: LETTR

ABCDEFGHIJKLMNOPQRSTUVWXYZ

###########O##############

####O##############O######

###################O######

#################O########

####O#####################

Lettres placées: LETTRE

ABCDEFGHIJKLMNOPQRSTUVWXYZ

###########O##############

####O##############O######

###################O######

#################O########

####O#############O#######

Lettres placées: LETTRES

**LIMITES DE TEMPS ET DE MEMOIRE**

* Temps : 1 s sur une machine à 1Ghz.
* Mémoire : 16000 Ko.

**CONTRAINTES**

* 1 <= *N* <= 1000, où *N* est le nombre de lignes de la grille.

**ENTREE**

* La première ligne de l'entrée contient un entier : le nombre de lignes de la grille, *N*.
* Les *N* lignes suivantes contiennent chacune 26 caractères. Chaque caractère est soit # (pas de trou), soit O (trou).

**SORTIE**

Vous devez écrire le mot codé par la carte, en lettres majuscules.

**EXEMPLE(S) D'ENTREE / SORTIE**

*Exemple 1 :*   
en entrée ...

|  |
| --- |
| 5 ###########O############## ####O##############O###### ###################O###### #################O######## ####O#############O####### |

en sortie ...

|  |
| --- |
| LETTRES |

**Coffre-fort**

**PROBLEME**

|  |  |
| --- | --- |
| Vous êtes preneur de son, spécialisé dans l'enregistrement du cri d'animaux tels que la Carpe Muette De Chine Orientale et le Ver Annelé Tubicole Des Fumeurs Noires. Ne parvenant à effectuer que très peu de prises correctes, vous avez décidé de vous reconvertir dans la redistribution de richesses au profit de soi, profession bien plus lucrative. Plus précisément, vous vous spécialisez dans le cambriolage de coffres-forts, en particulier ceux dont le code doit être fourni en tournant une roue graduée vers diverses positions successives. | D:\Cours\Cours IGA\IGA [2009-2010]\Olympiades Informatique\olymp-info\info\www.france-ioi.org\data\sujets\coffre_fort\coffre.jpg |

Vous cachez un micro à côté du coffre-fort de la victime, vous notez le numéro que porte la roue, puis vous enregistrez les clics produits par ladite roue lorsque le propriétaire du coffre effectue la combinaison. Grâce à votre ouïe fort développée, vous parvenez à compter les clics de chaque déplacement, ainsi qu'à deviner dans quel sens la roue est tournée.

Votre objectif est maintenant de retrouver la combinaison du coffre, à partir de la position initiale de la roue, ainsi que du nombre de crans et du sens de chaque déplacement.

**LIMITES DE TEMPS ET DE MEMOIRE**

* Temps : 1.5 s sur une machine à 1Ghz.
* Mémoire : 32000 Ko.

**CONTRAINTES**

* 2 <= *N* <= 10 000, où *N* est le nombre de crans sur la roue du coffre-fort. La roue est numérotée de 0 à *N-1* inclus.
* 0 <= *P* < *N*, où *P* est la position initiale de la roue du coffre-fort.
* 0 <= *M* <= 10 000, où *M* est le nombre de mouvements effectués sur la roue du coffre-fort.
* -*N* < *K*i < *N*, où *K*i est le nombre de crans dont la roue est tournée au mouvement *i*. *K*i > 0 si la roue est tournée dans le sens des numéros de position croissants, *K*i < 0 si la roue est tournée dans l'autre sens. Les *K*i sont tous différents de zéro.

**ENTREE**

* La première ligne de l'entrée contient trois entiers séparés par des espaces : *N*, *P* et *M*.
* Chacune des *M* lignes suivantes contient un entier *K*i par mouvement, le nombre de crans dont la roue est tournée au mouvement *i*.

**SORTIE**

Vous devez écrire *K* entiers à raison d'un par ligne. La ligne numéro *i* doit donner la position de la roue du coffre-fort après le mouvement *K*i. La dernière ligne doit donc contenir la position finale de la roue du coffre-fort, après tous les mouvements. Chaque position doit être un entier compris entre 0 et *N-1* inclus.

**EXEMPLE(S) D'ENTREE / SORTIE**

*Exemple 1 :*   
en entrée ...

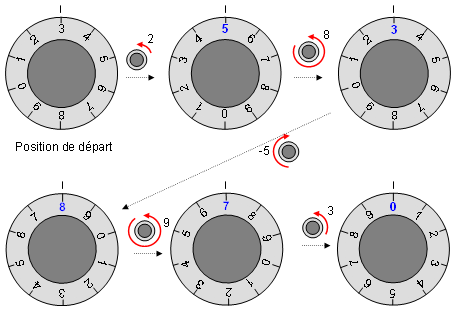
|  |
| --- |
| 10 3 5 2 8 -5 9 3 |

en sortie ...

|  |
| --- |
| 5 3 8 7 0 |
|  |

**COMMENTAIRES**

L'illustration suivante correspond à l'exemple d'entrée :



**Banderole**

**PROBLEME**

Fin août 2042, une grande fête est organisée au retour des candidats de l'équipe de France aux IOI. En effet, grâce à une préparation sans faille, ils ont réussi à décrocher 4 médailles d'or aux IOI. Vous proposez de participer à la décoration en accrochant une belle banderole aux couleurs de France-IOI.

Vous avez repéré qu'il y avait des piquets plantés bien alignés le long du chemin d'accès. Tous les piquets sont bien verticaux et plantés à intervalles réguliers (tous les mètres). Par contre les piquets ont des hauteurs très variées, que vous avez mesurées soigneusement.

Vous souhaitez que la banderole soit bien horizontale et bien tendue, sans piquets la cachant. Plus précisément, la banderole doit être accrochée sur deux piquets de même hauteur entre lesquels la distance est exactement la longueur de la banderole. De plus tous les piquets situés sous la banderole doivent avoir une hauteur inférieure ou égale à la hauteur des deux piquets sur lesquels est accrochée la banderole.

Écrivez un programme qui lit la longueur de la banderole, et la hauteur des piquets dans l'ordre de leur placement, puis qui détermine le nombre de positions différentes auxquelles il est possible d'accrocher la banderole.

**LIMITES DE TEMPS ET DE MEMOIRE**

* Temps : 1 s sur une machine à 1Ghz.
* Mémoire : 4000 Ko.

**CONTRAINTES**

* 1 <= *L* <= 100, où *L* est la longueur de la banderole, en mètres.
* 2 <= *N* <= 100 000, où *N* est le nombre de piquets, disposés tous les mètres.
* 1 <= *Hi* <= 1 000 000, où *Hi* est la hauteur du ième piquet.

**ENTREE**

* La première ligne de l'entrée contient deux entiers séparés par un espace *L* et *N*.
* La seconde ligne de l'entrée contient *N* entiers séparés par des espaces : les hauteurs *Hi*.

**SORTIE**

Affichez un entier : le nombre de positions différentes où l'on peut accrocher la banderole.

**EXEMPLE(S) D'ENTREE / SORTIE**

*Exemple 1 :*   
en entrée ...

|  |
| --- |
| 3 20 2 5 4 3 5 3 3 3 5 4 1 4 4 3 6 3 3 3 3 3 |

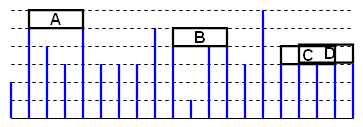
en sortie ...

|  |
| --- |
| 4 |
|  |

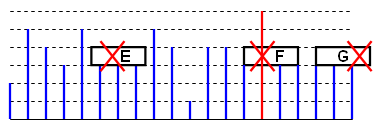
**COMMENTAIRES**

L'illustration ci-dessous correspond à l'exemple, et montre les quatre positions valides :

* la position A entre les piquets 2 et 5, à la hauteur 5
* la position B entre les piquets 10 et 14, à la hauteur 4
* la position C entre les piquets 16 et 19, à la hauteur 3
* la position D entre les piquets 17 et 20, également à la hauteur 3.



Les positions E, F et G représentées sur l'illustration suivante ne sont pas valides. La position E car elle ne correspond pas à une position entière, et que les extrémités de la banderole ne sont pas posées précisément sur des piquets. La position F car un piquet est plus haut que les extrémités de la banderole, et la position G car son extrémité droite ne repose sur aucun piquet.

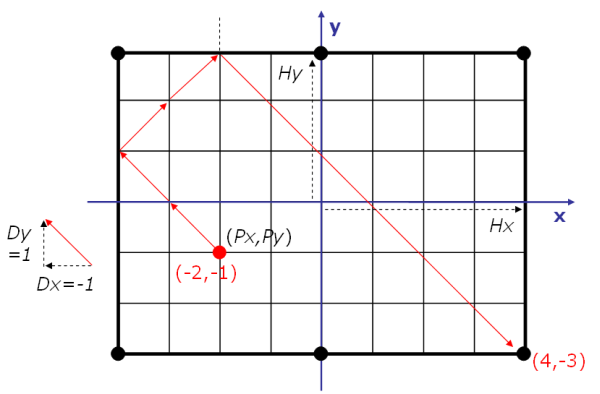


**Billard**

**PROBLEME**

|  |  |
| --- | --- |
| Vous êtes passionné de billard, et souhaitez programmer le meilleur jeu de billard jamais créé. Vous souhaitez cependant faire les choses petit à petit, en commençant par une version simplifiée.  Votre toute première version est très limitée : une seule boule (la blanche) est placée sur un billard américain. Il s'agit d'une table rectangulaire autour de laquelle se trouvent 6 trous : un dans chaque coin, et deux au milieu de chacun des bords horizontaux. Le joueur n'a pour l'instant que quatre directions de tir possibles : il tire toujours à un angle de 45 degrés par rapport aux bords de la table. Il ne peut pas appliquer le moindre effet sur la boule, mais peut régler la quantité d'énergie transmise à la boule lors du tir. | D:\Cours\Cours IGA\IGA [2009-2010]\Olympiades Informatique\olymp-info\info\www.france-ioi.org\data\sujets\billard\billard.png |

Votre objectif est d'écrire un programme qui détermine où s'arrête la boule de billard, soit parce qu'elle tombe dans un trou, soit parce qu'elle n'a plus d'énergie.



On vous décrit la table de billard en vous donnant les distances *Hx* et *Hy* entre le centre et les bords verticaux et horizontaux, respectivement. Les coordonnées initiales (*Px*,*Py*) de la boule blanche vous sont données dans un repère dont l'origine (0,0) est le centre de la table. Ainsi, le trou situé en haut à gauche du billard a pour coordonnées (*-Hx*, *Hy*), et le trou situé en haut et au milieu, a pour coordonnées (*0*, *Hy*), et ainsi de suite.

La direction de tir vous est donnée sous la forme d'un vecteur (*Dx*, *Dy*), indiquant le nombre d'unités *Dx* et *Dy* dont la balle se déplace respectivement horizontalement et verticalement juste après le départ, à chaque étape. *Dx* comme *Dy* ne peuvent valoir que -1 ou 1 : *Dx* vaut 1 si la boule se déplace plutôt vers la droite, et -1 si elle se déplace vers la gauche. De même, *Dy* vaut 1 si la boule se déplace plutôt vers le haut, et -1 vers le bas.

Lorsque la boule touche un bord (que son centre se trouve exatement sur le bord), elle rebondit contre celui-ci, et change de direction, en repartant toujours à un angle de 45 degrés par rapport à celui-ci.

Une fois le tir effectué, la boule blanche avance case par case en les traversant en diagonale, jusqu'à-ce qu'elle ait épuisé son énergie ou qu'elle atteigne exactement l'un des 6 trous de la table de billard, et y tombe. L'énergie initiale de la boule vous est donnée sous la forme du nombre de cases que peut traverser la boule avant de s'arrêter.

Étant donné la position initiale de la boule, sa direction son énergie initiale, vous devez indiquer les coordonnées exactes de la boule lorsqu'elle s'arrête, que ce soit parce qu'elle a épuisé son énergie, ou parce qu'elle est tombée dans un trou.

**LIMITES DE TEMPS ET DE MEMOIRE**

* Temps : 0.1 s sur une machine à 1Ghz.
* Mémoire : 32000 Ko.

**CONTRAINTES**

* 1 <= *Hx,Hy* <= 100 000, où *Hx* est la distance du centre de la table aux bords verticaux et *Hy* est la distance du centre aux bords horizontaux.
* *-Hx* < *Px* < *+Hx*, où *Px* est l'abscisse initiale de la boule blanche.
* *-Hy* < *Py* < *+Hy*, où *Py* est l'ordonnée initiale de la boule blanche.
* 1 <= *L* <= 1 000 000, où *L* est l'énergie initiale de la boule, donc le nombre maximal de cases qu'elle peut traverser.

**ENTREE**

* La première ligne contient deux entiers séparés par des espaces : *Hx* et *Hy*.
* La deuxième ligne contient deux entiers séparés par des espaces : *Px* et *Py*.
* La troisième ligne contient deux entiers séparés par des espaces : *Dx* et *Dy*.
* La quatrième et dernière ligne contient un seul entier : *L*.

**SORTIE**

Vous devez afficher deux entiers séparés par des espaces : l'abscisse puis l'ordonnée de la position finale de la boule.

**EXEMPLE(S) D'ENTREE / SORTIE**

*Exemple 1 :*   
en entrée ...

|  |
| --- |
| 4 3 -2 -1 -1 1 20 |

en sortie ...

|  |
| --- |
| 4 -3 |
|  |

**COMMENTAIRES**

L'exemple correspond exactement à l'illustration donnée plus haut. La boule pourrait traverser 20 cases, mais n'en traverse que 10, car le 10ème déplacement la fait tomber dans le trou en bas à droite, qui a pour coordonnées (4, -3).

**Fragments d'ADN**

**PROBLEME**

Depuis que les biologistes disposent de machines capables de séquencer des morceaux de génome, ils récupèrent des giga-octets de données. Cependant des données brutes ne présentent aucun intérêt si l'on ne dispose pas d'algorithmes suffisamment efficaces pour les analyser. Ce sujet vous propose un de ces nombreux problèmes d'analyse de séquence d'ADN.

D:\Cours\Cours IGA\IGA [2009-2010]\Olympiades Informatique\olymp-info\info\www.france-ioi.org\data\sujets\motifs_adn\dna.png

On commence par vous décrire une grande chaîne d'ADN, c'est-à-dire une suite de lettres parmi C,G,T,A. Ensuite on vous donne des motifs d'ADN correspondant à des gènes, c'est-à-dire une suite de quelques dizaines de lettres (toujours parmi C,G,T,A). Le but est de déterminer pour chaque gène quelle est la zone de la grande chaîne d'ADN qui ressemble le plus au motif de ce gène.

Concrètement, on veut minimiser le nombre de différences entre le code du gène et le code d'un fragment de la grande chaîne d'ADN (le fragment doit être de même longueur que le code du gène et doit être entièrement inclus dans la chaîne d'ADN).

Pour simplifier, on va travailler en considérant que l'ADN est simplement une suite de 0 et de 1 (remarquez que cela ne change pas fondamentalement le problème). On supposera aussi que tous les gènes considérés sont des séquences de même longueur. Ce qu'on vous demande précisément dans ce sujet, c'est de déterminer pour chaque gène quel est le nombre minimum de différences que l'on peut obtenir entre le code du gène et le code d'un fragment de même longueur entièrement inclus dans la grande chaîne d'ADN.

Notez que vous ne pouvez pas retourner les gènes ou la chaîne d'ADN.

**LIMITES DE TEMPS ET DE MEMOIRE**

* Temps : 0.4 s sur une machine à 1Ghz.
* Mémoire : 32000 Ko.

**CONTRAINTES**

* 1 <= *G* <= 100, où *G* est le nombre de gènes que l'on cherche à reconnaître approximativement dans la grande chaîne.
* 1 <= *L* <= 30, où *L* est la longueur des segments d'ADN des gènes considérés.
* L <= *N* <= 100 000, où *N* est la longueur de la grande chaîne d'ADN à étudier.

**ENTREE**

* La première ligne de l'entrée contient trois entiers séparés par des espaces : *G*, *L* et *N*.
* Chacune des *G* lignes suivantes contient *L* entiers (0 ou 1) séparés par des espaces décrivant un gène.
* La dernière ligne contient *N* entiers (0 ou 1) séparés par des espaces décrivant la grande chaîne.

**SORTIE**

Vous devez écrire *G* lignes contenant chacune un entier. La ligne *i*ème ligne doit donner le nombre minimal de différences que l'on peut trouver entre le code du *i*ème gène de l'entrée et un fragment de même longueur dans la grande chaîne d'ADN fournie.

**EXEMPLE(S) D'ENTREE / SORTIE**

*Exemple 1 :*   
en entrée ...

|  |
| --- |
| 3 6 12 0 0 1 1 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 0 1 1 0 0 0 0 1 1 0 0 |

en sortie ...

|  |
| --- |
| 1 3 0 |
|  |

**COMMENTAIRES**

Premier motif : 1 différence au moins

1 0 1 1 0 0 0 0 1 1 0 0

0 0 1 1 0 1

Second motif : 3 différences au moins

1 0 1 1 0 0 0 0 1 1 0 0

1 1 1 1 1 1

Troisième motif : 0 différence est possible

1 0 1 1 0 0 0 0 1 1 0 0

1 0 0 0 0 1