Olympiades Régionales d'Informatique Franco-Australiennes

11 mars 2007

Durée : 4 heures

4 problèmes

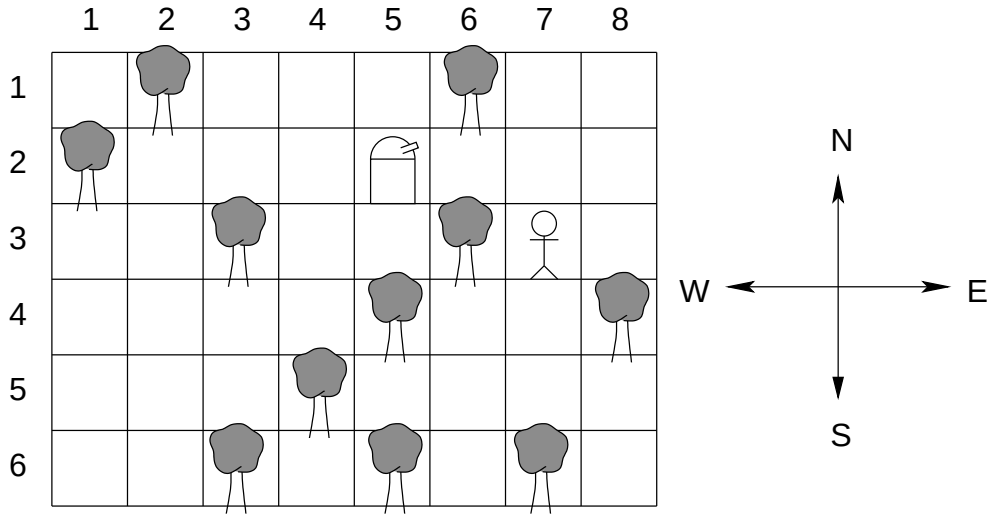
Problème 1
Robot sauveteur

Limites de temps et de mémoire : 1 seconde, 32 Mo

Vous êtes perdu dans une grande forêt, avec pour seule compagnie des singes et des perruches. Depuis plusieurs jours, vous tentez de sortir de la forêt, mais malgré tous vos efforts vous finissez toujours par revenir à votre point de départ. Résigné, vous décidez d'attendre qu'une équipe de sauvetage vous retrouve. Plusieurs jours s'écoulent avant qu'un robot ne tombe sur vous, apportant de la nourriture, de l'eau, et surtout une batterie pour votre ordinateur portable ! Ce n'est pas l'équipe de sauvetage que vous espériez, mais au moins, vous êtes tiré d'affaire. Enfin, presque.

Le robot n'est pas capable de vous dire comment revenir à l'équipe de sauvetage. Il ne peut que vous donner le plan de la forêt, la position où il vous a trouvé, et les mouvements qu'il a faits pour aller de l'équipe de sauvetage jusqu'à vous. Malheureusement vous tombez rapidement sur un bug dans le programme du robot : quand le robot essaye d'aller sur un arbre ou de sortir de la forêt, il n'avance pas mais enregistre quand même le mouvement dans son journal de bord.

Imaginez par exemple que la forêt soit représentée par la grille ci-dessous, dont certaines cases contiennent un arbre. Si le robot part de la case $(5, 2)$ et effectue les mouvements `E`, `E`, `E`, `S`, `S`, `W`, il finira dans la case $(7, 3)$ où vous vous trouvez (notez qu'il n'a pas réellement effectué le deuxième mouvement `S` car un arbre bloque le passage).



Supposez maintenant que le robot est parti de la case $(7, 1)$ et a effectué les mêmes mouvements `E`, `E`, `E`, `S`, `S`, `W`. Là aussi, il arrivera en $(7, 3)$ (cette fois, il n'a pas effectué les deuxième et troisième `E` car cela l'aurait fait sortir de la forêt). En fait le robot peut être parti de n'importe laquelle des cases suivantes : $(7, 1)$, $(8, 1)$, $(5, 2)$, $(6, 2)$, $(7, 2)$, $(8, 2)$, $(7, 3)$ ou $(8, 3)$, et dans tous les cas arriver à la case où vous vous trouvez.

Notez que le robot peut être parti de la case où vous vous trouvez actuellement, et/ou y passer un nombre quelconque de fois pendant son trajet (à ce moment là, vous étiez probablement ailleurs dans la forêt à bavarder avec les singes). Le robot ne peut cependant pas partir d'une case contenant un arbre, ni d'une case située hors de la forêt.

Étant donnés la carte de la forêt, votre position actuelle, et les mouvements effectués par le robot pour vous trouver, votre objectif est de trouver le nombre de positions de départ possibles pour le robot.

Contraintes

- $1 \leq w, h \leq 100$, où w est la largeur de la carte et h sa hauteur (mesurées en nombre de cases) ;
- $1 \leq p \leq 1\,000$, où p est le nombre de mouvements enregistrés par le robot.

Entrée

Votre programme doit lire sur l'entrée standard. La première ligne contiendra les entiers w et h, séparés par un espace, dont la signification est indiquée plus haut.

Les h lignes suivantes décrivent chacune une rangée de la carte de la forêt. Une ligne contiendra w caractères, chacun étant choisi parmi :

- '`.`' – une zone vide de la forêt ;
- '`T`' – un arbre ;
- '`U`' – la position à laquelle le robot vous a trouvé.

Le caractère '`U`' apparaîtra exactement une fois sur la carte.

La ligne suivante contiendra l'entier p. Ensuite, les p lignes qui suivent décriront les mouvements effectués par le robot pour vous trouver. Chacune de ces lignes contiendra une lettre – `N`, `E`, `S` ou `W` – correspondant respectivement aux déplacements vers le nord, l'est, le sud ou l'ouest. On vous garantit qu'il existe au moins un point de départ pour le robot tel qu'il arrive jusqu'à vous en effectuant cette séquence de mouvements.

Toutes les lettres de l'entrée seront données en majuscule.

Sortie

Votre programme doit écrire une seule ligne sur la sortie standard. Cette ligne doit contenir un seul entier, le nombre de positions de départ possibles pour le robot.

Exemple d'entrée

```
8 6
.T...T..
T.......
..T..TU.
....T..T
...T....
..T.T.T.
6
E
E
E
S
S
W
```

Exemple de sortie

```
8
```

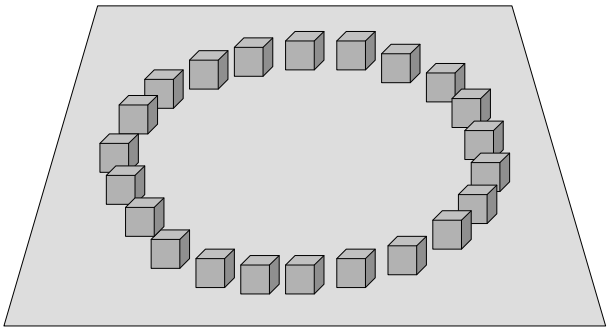
Score

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

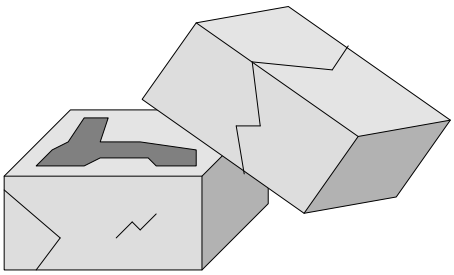
Problème 2
La chambre secrète de la pyramide de Gizeh

Limites de temps et de mémoire : 3 secondes, 32 Mo

Vous êtes membre d'une équipe d'archéologues qui vient juste de découvrir une chambre secrète dans la grande pyramide de Gizeh. À votre grande déception, cette chambre est complètement vide, à part 22 petites pierres cubiques disposées en cercle sur le sol, au milieu de la chambre.



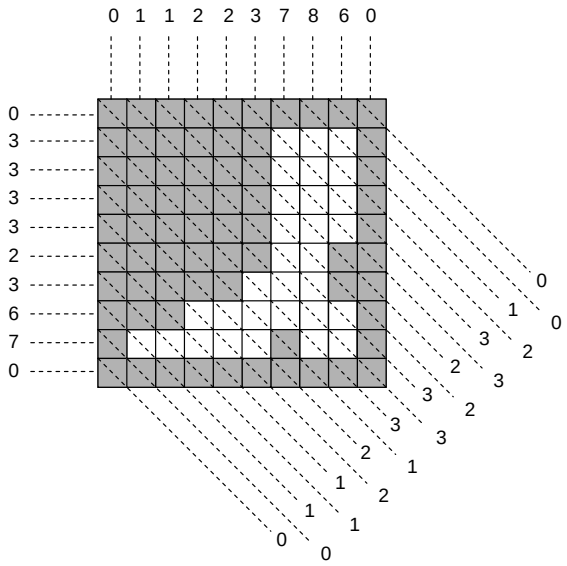
Vous remarquez que deux de ces cubes sont brisés, et après une observation plus approfondie, vous vous rendez compte qu'à l'intérieur de chacun d'eux, au milieu et parallèlement au sol, est gravé un hiéroglyphe. Un message caché du plus haut intérêt historique attend d'être découvert. Cependant, vous ne pouvez pas casser ni même déplacer les 20 cubes restants pour lire le message, donc vous décidez de recourir à la radiographie, en prenant des images aux rayons X de chaque cube.



Les résultats montrent clairement que, comme vous vous y attendiez, un hiéroglyphe est gravé à l'intérieur de chaque cube, selon un plan parallèle au sol. Mais comme les radiographies sont prises à partir des côtés du cube et non du haut, vous ne pouvez pas lire le hiéroglyphe directement. Vous avez besoin de votre ordinateur pour vous y aider.

Pour chaque cube, la tranche sur laquelle est gravé le hiéroglyphe peut être représentée par une grille de $N \times N$ cases, chacune étant pleine ou vide. À partir de vos trois radiographies, vous connaissez le nombre exact de cases vides dans chaque colonne, ligne et diagonale (les diagonales sont dans la direction haut-gauche vers bas-droite). Les coordonnées des cases de la grille sont telles que la case du coin supérieur gauche a les coordonnées $(0, 0)$, et celle du coin inférieur droit les coordonnées $(N-1, N-1)$. Les lignes, colonnes et diagonales sont numérotées $0, 1, 2, \ldots$

Une case de coordonnées (x, y) appartient à la colonne x, à la ligne y, et à la diagonale $x - y + (N-1)$; par conséquent, la petite diagonale du coin inférieur gauche est la diagonale 0, la longue diagonale passant par les cases $(0, 0)$ et $(N-1, N-1)$ est la diagonale $N-1$, et la petite diagonale du coin supérieur-droit est la diagonale $2N-2$.



Le schéma ci-dessus est un exemple de hiéroglyphe avec le résultat des trois radiographies ; il indique le nombre de cases vides dans chaque colonne, ligne et diagonale.

Étant donné le nombre de cases vides dans chaque colonne, ligne et diagonale de la grille, votre objectif est de trouver un hiéroglyphe dont le nombre de cases vides dans chaque direction est le plus proche possible de celui qui correspond dans l'entrée.

Contraintes

- $1 \leq N \leq 100$, où N est la longueur du côté de la grille, en cases.

Entrée

Votre programme doit lire sur l'entrée standard. La première ligne contiendra l'entier N, la taille de la grille $N \times N$.

La deuxième ligne contiendra N entiers séparés par des espaces. Le i^{e} de ces entiers est le nombre de cases vides dans la i^{e} colonne de la grille, en partant de la gauche.

La troisième ligne contiendra N entiers séparés par des espaces. Le i^{e} de ces entiers est le nombre de cases vides dans la i^{e} ligne de la grille, en partant du haut.

La quatrième ligne contiendra $(2N - 1)$ entiers séparés par des espaces. Le i^{e} de ces entiers est le nombre de cases vides dans la i^{e} diagonale de la grille, en partant du coin inférieur gauche.

Sortie

Votre programme doit écrire N lignes sur la sortie standard, décrivant un hiéroglyphe possible. Chaque ligne doit contenir N caractères décrivant une rangée de la grille, où le caractère '1' indique une case pleine, et '0' une case vide. La sortie ne doit pas contenir d'espace.

Exemple d'entrée

```
10
0 1 1 2 2 3 7 8 6 0
0 3 3 3 3 2 3 6 7 0
0 0 1 1 1 2 2 1 3 3 3 2 2 3 3 2 1 0 0
```

Exemple de sortie

```
1111111111
1111110001
1111110001
1111110001
1111110001
1111110011
1111100011
1110000001
1000001001
1111111111
```

Score

Pour un test d'entrée donné, les nombres de cases vides sur chaque colonne, ligne et diagonale sont comparés aux valeurs d'entrée. L'erreur d'une colonne, ligne ou diagonale est la valeur absolue de la différence entre la valeur du test d'entrée et la valeur obtenue pour votre sortie. L'erreur totale de votre sortie sera la somme de ces erreurs. Votre score pour le test d'entrée sera :

- 0 %, si l'erreur totale de votre sortie est supérieure à l'erreur totale obtenue avec une grille ne comportant aucune case vide ;
- 100 %, si l'erreur totale de votre sortie est de zéro ;
- sinon, il sera déterminé selon une loi linéaire en fonction de l'erreur totale de votre sortie, les scores 0 % et 100 % sur l'échelle correspondant aux erreurs totales décrites ci-dessus.

Problème 3
Architecture

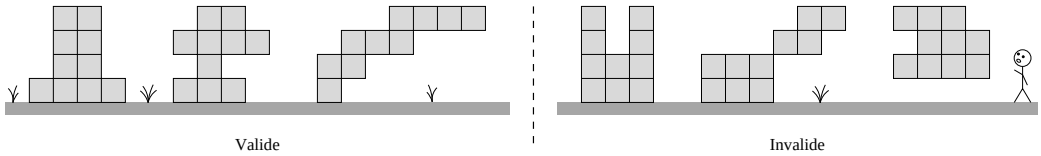
Limites de temps et de mémoire : 2 secondes, 32 Mo

Vous êtes un architecte d'avant-garde, connu pour vos belles et hautement instables constructions. Votre travail actuel est de construire un complexe d'appartements aux profils des plus étonnants et aux vues non moins superbes.

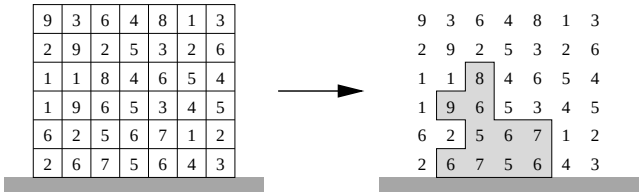
Votre complexe d'appartements doit comporter un certain nombre d'étages, dans lesquels se trouveront un certain nombre d'appartements. Il est vital que :

- Chaque étage soit composé d'une seule rangée d'appartements connexes. Avoir des sections non-connexes sur un étage serait d'un style années 90 les plus rétrogrades.
- Au moins un appartement sur chaque étage doit reposer sur un appartement de l'étage inférieur, sauf l'étage du bas qui doit reposer sur le sol.

Quelques exemples d'immeubles valides et invalides sont montrés sur le schéma ci-dessous. Les immeubles de gauche suivent les règles indiquées plus haut. Les trois immeubles de droite sont invalides : dans le premier immeuble, le troisième étage contient deux sections non-connexes, dans le deuxième immeuble, le troisième étage ne repose pas sur le second, et dans le dernier immeuble, l'étage inférieur ne touche pas le sol.



Malheureusement, votre génie créatif incomparable est bridé par des contraintes purement commerciales : les appartements ne peuvent être vendus s'ils n'ont pas la meilleure vue possible. Pour chaque point de l'espace, vous disposez d'un nombre indiquant à quel point la vue y est splendide. Étant donné un nombre d'appartements N, votre objectif est de concevoir un immeuble comportant exactement N appartements et tel que la somme des « qualités de vue » pour les N appartements soit aussi grande que possible.



Par exemple, considérez la grille située à gauche du schéma ci-dessus. Elle indique les qualités de vue pour un terrain, la ligne du bas contenant les scores des appartements du rez-de-chaussée, la deuxième ligne ceux du premier étage, etc.

Supposez qu'on vous ait demandé de concevoir un immeuble contenant $N = 10$ appartements. La partie de droite du schéma contient un exemple d'immeuble qui convient. La somme de ses qualités de vue est $8+9+6+5+6+7+6+7+5+6 = 65$. On peut montrer qu'il s'agit du meilleur immeuble que vous pouvez concevoir, c'est-à-dire que vous ne pouvez concevoir aucun immeuble ayant un score total supérieur à 65.

Contraintes

- $1 \leq N \leq 80$, où N est le nombre d'appartements dans votre immeuble ;
- $1 \leq W, H \leq 80$, où la grille des qualités de vue est de largeur W et de hauteur H ;
- Il est possible de construire un immeuble ayant N appartements, c'est-à-dire que l'on a $N \leq W \times H$.

De plus, pour 30 % des immeubles à concevoir, l'entrée satisfera $1 \leq N, W, H \leq 20$.

Entrée

Votre programme doit lire sur l'entrée standard. La première ligne contiendra l'entier N, le nombre d'appartements que doit contenir votre immeuble.

La deuxième ligne de l'entrée contiendra les entiers W et H, séparés par un espace : les dimensions de la grille de nombres.

Les H lignes suivantes contiendront chacune W entiers séparés par des espaces : les qualités de vue pour les différents points de l'espace. La dernière ligne de l'entrée contient les qualités de vue du rez-de-chaussée, l'avant-dernière ligne ceux du premier étage, etc. Toutes les qualités de vue seront comprises entre 1 et 100 000 inclus.

Sortie

Votre programme doit écrire une seule ligne sur la sortie standard. Cette ligne doit contenir un seul entier, la plus grande somme de qualités de vue possible pour votre immeuble.

Exemple d'entrée

```
10
7 6
9 3 6 4 8 1 3
2 9 2 5 3 2 6
1 1 8 4 6 5 4
1 9 6 5 3 4 5
6 2 5 6 7 1 2
2 6 7 5 6 4 3
```

Exemple de sortie

```
65
```

Score

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

Problème 4
Réchauffement climatique

Limites de temps et de mémoire : 1 seconde, 16 Mo

Le réchauffement climatique est un sujet plutôt chaud en ce moment, un grand nombre de scientifiques tentent de modéliser l'évolution du climat pour les décennies à venir. L'un deux a élaboré une méthode qui, d'après lui, fournit un certain nombre de prévisions quant au climat futur en un point donné de la Terre. Chaque prévision est de la forme : « entre le jour X et le jour Y, la température à cet endroit atteindra un maximum de K degrés », où les paramètres X, Y et K prennent des valeurs entières.

Le scientifique est convaincu que ses prévisions sont extrêmement précises, tandis que vous n'êtes pas aussi sûr que lui. Vous voulez écrire un programme qui indique si, pour un point donné de la Terre, les prévisions du scientifique sont cohérentes. Par *cohérent*, on entend qu'il est possible de trouver une séquence de températures $M_1, M_2, \ldots M_N$, M_i étant la température maximale au jour i, pour lesquelles toutes les prévisions sont satisfaites. Une prévision (X, Y, K) est *satisfaite* si, entre le jour X et le jour Y (inclus), la température ne dépasse jamais K, et il existe au moins un jour dont la température maximale atteint exactement K.

Contraintes

- $1 \leq T \leq 5$, où T est le nombre de points sur Terre où vous souhaitez tester la méthode du scientifique ;
- $1 \leq N \leq 100\,000$, où N est le nombre de jours étudiés pour un emplacement ;
- $1 \leq P \leq 50\,000$, où P est le nombre de prévisions faites pour un emplacement ;
- $1 \leq X \leq Y \leq N$, où X et Y sont les jours impliqués dans la prévision (où 1 est le premier jour de la période d'étude et N le dernier) ;
- $1 \leq K \leq 100\,000$, où K est une température maximale prévue (mesurée en centièmes de degrés Kelvin).

De plus, dans 30 % des cas, toutes les périodes d'étude satisferont $P \leq 100$.

Entrée

Votre programme doit lire sur l'entrée standard. La première ligne contiendra un unique entier T, le nombre de points où vous souhaitez tester la méthode du scientifique. Les ligne suivantes contiendront la description des T tests, l'un après l'autre.

Chaque description commence par une ligne contenant les deux entiers N et P, séparés par un espace. Les P lignes suivantes contiennent chacune une prévision. Chaque prévision est décrite par trois entiers X, Y et K, séparés par des espaces.

Sortie

Votre programme doit écrire exactement T lignes sur la sortie standard. La i^{e} de ces lignes doit contenir un unique entier, correspondant à la description du i^{e} test. Pour chaque test, cet entier doit être '1' si les prévisions correspondantes sont cohérentes, ou '0' s'il est impossible de satisfaire toutes les contraintes à cet endroit.

Exemple d'entrée

```
2
5 3
1 2 2
4 5 1
2 4 3
4 4
3 3 4
4 4 3
1 4 2
1 2 1
```

Exemple de sortie

```
1
0
```

Explication

Les données d'exemple contiennent deux descriptions de test. Le premier test cherche à vérifier trois prévisions sur une durée de cinq jours. Ces prévisions sont les suivantes :

- parmi les jours 1 et 2 : au moins un jour aura une température maximale de 2, et aucun jour ne dépassera la température 2 ;
- parmi les jours 4 et 5 : au moins un jour aura une température maximale de 1, et aucun jour ne dépassera la température 1 ;
- parmi les jours 2, 3 et 4 : au moins un jour aura une température maximale de 3, et aucun jour ne dépassera la température 3.

La séquence de températures 1, 2, 3, 1, 1 satisfait ces prévisions. Par conséquent, cet ensemble de prévisions est cohérent.

Le second test cherche à vérifier quatre prévisions sur une durée de quatre jours. Ces prévisions sont les suivantes :

- le jour 3 doit avoir une température maximale de 4 ;
- le jour 4 doit avoir une température maximale de 3 ;
- parmi les jours 1, 2, 3 et 4 : au moins un jour aura une température maximale de 2, et aucun jour ne dépassera la température 2 ;
- parmi les jours 1 et 2 : au moins un jour aura une température maximale de 1, et aucun jour ne dépassera la température 1.

Clairement, la troisième prévision entre en conflit avec les deux premières, donc cet ensemble de prévisions n'est pas cohérent.

Score

Le score pour chaque test d'entrée sera de 100 % si la bonne séquence de réponses est fournie sur la sortie standard, ou de 0 % sinon.