Olympiades Régionales d'Informatique Franco-Australiennes

16 mars 2008

Durée : 4 heures

4 problèmes

Problème 1
Graffiti

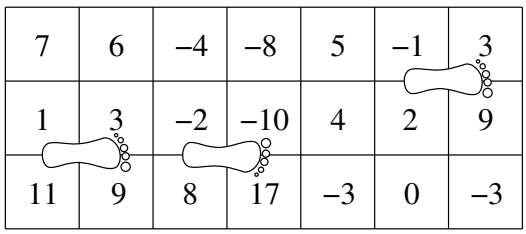
Limites de temps et de mémoire : 1 seconde, 32 Mo

Sur le chemin que vous empruntez quotidiennement pour aller à l'école, il y a un long trottoir pavé par des blocs de béton. Le trottoir fait trois blocs de large et est assez long (il semble toujours plus long quand vous rentrez à la maison). Un jour, vous découvrez que de jeunes irresponsables ont tracé des graffitis sur tous ces blocs. En regardant de plus près, vous réalisez qu'il s'agit de graffitis mathématiques : chaque bloc de béton a été marqué avec un entier.

Cette fois, la curiosité ne vous emporte pas. Vous décidez plutôt de jouer à un jeu avec vos amis sur le chemin de l'école. Le jeu se joue comme suit :

- Vous commencez par vous placer juste avant le début du trottoir contenant les graffitis.
- Chaque pas le long du trottoir doit être exactement sur une position à cheval sur les coins de quatre blocs. En marchant sur un coin d'un bloc, vous gagnez des points. Le nombre de points que vous gagnez pour un pas donné est la somme des nombres se trouvant sur les quatre blocs sur lesquels vous avez marché.
- Vous ne pouvez pas marcher sur le coin d'un bloc sur lequel vous avez déjà mis le pied. Autrement dit, une fois que vous avez mis le pied sur un coin d'un bloc, vous ne pouvez marcher sur aucun autre coin de ce bloc.
- Le nombre total de points que vous gagnez est additionné et devient votre score final.

Par exemple, le trottoir ci-dessous fait trois blocs de large et sept blocs de long. Le meilleur score possible est 50, obtenu en marchant sur les coins indiqués.



$$(1+3+11+9)+(-2-10+8+17)+(-1+3+2+9)=24+13+13=50$$

Notez que vous pouvez laisser un nombre quelconque de blocs entre vos pas et ainsi, si tous les blocs contiennent un nombre négatif de points, vous pouvez simplement choisir de ne marcher sur aucun bloc et garder un score de 0.

Après plusieurs jours à jouer à ce jeu avec vos amis et à perdre systématiquement, vous décidez d'utiliser votre ordinateur pour vous assurer le meilleur score. Votre objectif est de déterminer le meilleur score qui peut être obtenu sur un trottoir couvert de graffitis, étant donné la longueur du trottoir et le nombre indiqué sur chaque bloc.

Contraintes

- $1 \leq L \leq 500\,000$, où L est la longueur du trottoir.

Entrée

Votre programme doit lire sur l'entrée standard. La première ligne de l'entrée doit contenir un simple entier L, la longueur du trottoir.

Les L lignes suivantes ont chacune la forme x y z, décrivant les nombres écrits sur chacun des trois blocs de la rangée décrite. Ces trois nombres sont donnés dans l'ordre de la gauche à la droite du trottoir (tel que vu par une personne qui y marche), et seront toujours des entiers entre $-100\,000$ et $100\,000$, inclus.

Sortie

Votre programme doit écrire une ligne sur la sortie standard. Cette ligne doit contenir un simple entier, donnant le plus grand score que vous pouvez atteindre en marchant le long du trottoir. On vous garantit que la réponse ne dépassera jamais $2\,000\,000\,000$.

Exemple d'entrée

```
7
7 1 11
6 3 9
-4 -2 8
-8 -10 17
5 4 -3
-1 2 0
3 9 -3
```

Exemple de sortie

```
50
```

Score

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

Problème 2
Piste de luge

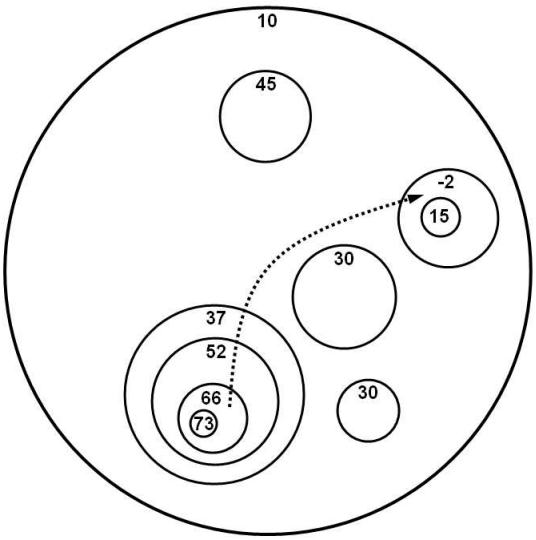
Limites de temps et de mémoire : 1 seconde, 40 Mo

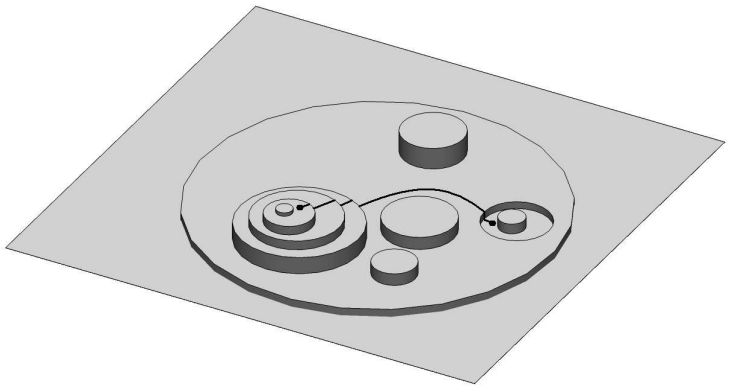
Pour la première fois depuis longtemps, sans doute à cause du changement climatique, il a beaucoup neigé dans votre région. Vous décidez d'en profiter pour organiser une compétition de luge dans les collines se trouvant derrière votre maison. Vous cherchez un endroit adapté pour y créer une piste, et souhaitez trouver une zone qui permette aux luges d'atteindre des vitesses élevées, bien qu'il n'y ait pas beaucoup de pentes raides dans les environs.

Vous avez obtenu une carte topographique de la zone, sur laquelle des *contours* sont représentés. Ces contours sont des cercles parfaits, et la carte indique l'altitude de la zone qui touche le bord intérieur de chaque cercle. La zone se trouvant en dehors de tous les cercles a une altitude de zéro. Comme ces contours sont la seule information dont vous disposez sur la forme des collines, vous pouvez considérer que les zones entre les contours sont plates. Deux contours ne se croisent jamais et ne se touchent jamais.

En utilisant cette carte, votre but est de trouver une piste qui ne traverse pas plus de K cercles, pour un nombre donné K (vous ne voulez pas que la piste soit trop bosselée), et telle que la différence totale d'altitude entre le point de départ et l'arrivée soit aussi grande que possible. Votre piste peut contenir des parties en montée mais ne doit jamais atteindre une altitude plus élevée que le point de départ.

Par exemple, le diagramme ci-dessous montre une telle carte. Si le nombre maximum de cercles que vous pouvez traverser est $K = 4$, alors la piste ayant la plus grande différence d'altitude est celle représentée par la ligne pointillée, qui a une différence d'altitude de $66 - (-2) = 68$. Le deuxième diagramme donne une vue plus claire du paysage décrit par cette carte (pas à l'échelle).





Contraintes

Pour 95% des points disponibles :

- $1 \leq C \leq 2000$, où C est le nombre de cercles sur la carte ;
- $1 \leq K \leq 200$, où K est le nombre maximal de cercles que votre piste peut croiser ;
- $-1000 \leq X, Y \leq 1000$, où X et Y sont les coordonnées du centre d'un cercle ;
- $1 \leq R \leq 2000$, où R est le rayon d'un cercle ;
- $-1000 \leq A \leq 1000$, où A est l'altitude d'une zone.

Pour 30% de ces points (qui sont une partie des 95% mentionnés ci-dessus) :

- $1 \leq C \leq 100$, où C est le nombre de cercles de la carte ;

De plus, pour les 5% de points restants :

- $1 \leq C \leq 40\,000$, où C est le nombre de cercles de la carte.

Entrée

Votre programme doit lire sur l'entrée standard. La première ligne de l'entrée doit contenir les entiers C et K, où C est le nombre de cercles de la carte, et K est le nombre maximal de cercles que votre piste peut traverser.

Les C lignes suivantes décrivent les cercles par ordre croissant de leur rayon. Chacune de ces lignes décrit un cercle sous la forme de quatre entiers séparés par des espaces : X_i, Y_i, R_i et A_i, où (X_i,Y_i) sont les coordonnées du centre du cercle, R_i est le rayon et A_i est l'altitude des zones qui touchent le bord interne du cercle..

Sortie

Votre programme doit écrire une ligne sur la sortie standard. Cette ligne doit contenir un simple entier, qui donne la différence maximale d'altitude que vous pouvez obtenir entre le point de départ et l'arrivée d'une piste en respectant les contraintes décrites ci-dessus.

Exemple d'entrée

```
10 4
38 61 2 73
69 34 3 15
61 59 4 30
40 60 5 66
58 44 6 30
71 34 6 -2
47 21 6 45
41 58 8 52
41 57 11 37
48 40 33 10
```

Exemple de sortie

```
68
```

Score

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

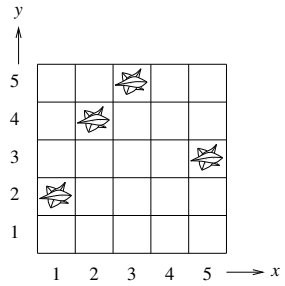
Problème 3
Souffleur de feuilles

Fichier d'entrée : *leaf*.k.*in*
Fichier de sortie : *leaf*.k.*out*

Tâche à sortie uniquement

L'automne est arrivé, et les feuilles des arbres tombent dans votre cour pavée. Bien que les feuilles soient jolies, votre cour est plus jolie, donc vous décidez de la débarasser de ces feuilles.

Votre cour est constituée d'une grille de cases, comme illustré ci-dessous ; ces cases peuvent être décrites par des coordonnées (x, y). Les feuilles sont tombées pour former n piles, chacune sur une case différente.

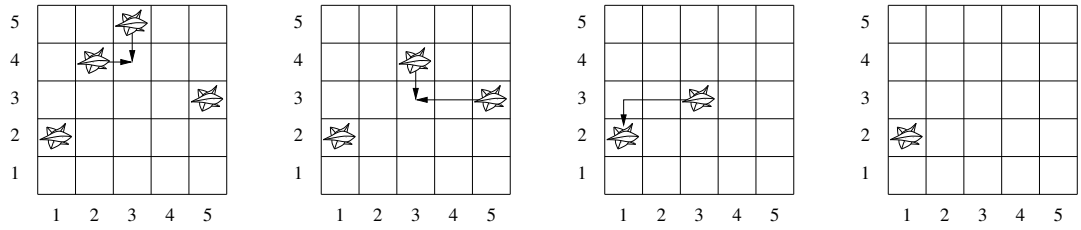


Vous disposez d'un appareil à souffler les feuilles, qui vous permet de déplacer ces piles. Le souffleur peut déplacer une pile de feuilles horizontalement ou verticalement d'une case à l'autre. Si deux piles se rencontrent, elles se rejoignent pour former une unique pile (plus grande), que vous pouvez continuer à déplacer comme précédemment.

Vous souhaitez réunir les feuilles en une seule pile, en utilisant aussi peu de *mouvements de piles* que possible. Un mouvement de pile consiste à souffler une pile de feuilles (d'une taille quelconque) vers une case adjacente.

Par exemple, considérez l'illustration ci-dessous. Les feuilles commencent en quatre piles, situées sur les cases $(1, 2)$, $(2, 4)$, $(3, 5)$ et $(5, 3)$, comme montré dans le premier diagramme.

- Pour commencer, vous pouvez souffler les deux piles en $(2, 4)$ et $(3, 5)$ vers la case $(3, 4)$. Ceci nécessite deux mouvements de pile. Le résultat est que les deux piles se rejoignent en une unique pile, comme indiqué sur le deuxième diagramme.
- Vous pouvez ensuite souffler la nouvelle pile en $(3, 4)$ vers $(3, 3)$, ce qui nécessite un mouvement de pile, puis souffler la pile tout à droite en $(5, 3)$ vers $(3, 3)$, utilisant deux mouvements de pile supplémentaires. Le résultat est que les deux piles fusionnent en une seule pile en $(3, 3)$, comme indiqué dans le troisième diagramme.
- Pour finir, vous pouvez souffler la nouvelle pile en $(3, 3)$ deux cases vers la gauche, et une case vers le bas, pour lui faire rejoindre la pile en $(1, 2)$. Ceci nécessite trois autres mouvements. A ce moment, toutes les feuilles ont été réunies en une seule pile, comme indiqué dans le quatrième diagrame, donc c'est terminé.



En suivant ces étapes, un total de $2 + 3 + 3 = 8$ mouvements de pile est utilisé, ce qui est le plus petit nombre possible.

Vous n'êtes pas obligé de trouver la meilleure solution possible. Vous devez simplement utiliser aussi peu de mouvements de pile que vous le pouvez. Votre solution sera comparée avec les autres solutions, et les meilleures obtiendront plus de points. Voyez la section *Score* pour plus de détails.

Entrée

On vous donne dix fichiers d'entrée `leaf.`k`.in` ($1 \leq k \leq 10$).

La première ligne de chaque fichier contient un simple entier n, donnant le nombre de piles ($2 \leq n \leq 500$). Chacune des n lignes qui suivent est de la forme x y, et donne les coordonnées d'une pile de feuilles. Toutes les coordonnées x et y sont des entiers dans l'intervalle $1 \leq x, y \leq 1\,000$.

Sortie

Pour chaque fichier d'entrée `leaf.`k`.in`, vous devez produire un fichier de sortie `leaf.`k`.out` qui contient votre solution.

Chaque fichier solution doit contenir une ligne pour chaque mouvement de pile, du premier au dernier mouvement effectué. Chaque ligne doit être de la forme x y p q, ce qui indique que la pile se trouvant sur la case (x, y) est soufflée vers la case (p, q). Ces deux cases doivent être adjacentes horizontalement ou verticalement, et toutes les coordonnées de cases doivent rester dans l'intervalle $1 \leq x, y, p, q \leq 1\,000$.

Exemple d'entrée

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

Exemple de sortie

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

Score

Il n'y a pas de "meilleure solution" que vous devez absolument atteindre. Votre score pour chaque scénario d'entrée sera en effet déterminé relativement aux autres candidats contre lesquels vous concourrez (ainsi que l'une des solutions des juges).

Pour chaque scénario d'entrée, le candidat qui trouve une solution valide en utilisant le moins de mouvements sera identifié. Supposez que ce candidat utilise p mouvements de piles au total. En supposant que votre sortie contienne également une solution valide, son score sera calculé comme suit :

- Si votre solution utilise entre p et $1.1 \times p$ mouvements de pile, son score sera calculé selon une échelle linéaire entre 100% et 50%, où 100% correspond à p mouvements, et 50% correspond à $1.1 \times p$ mouvements.
- Si votre solution utilise entre $1.1 \times p$ et $2 \times p$ mouvements de pile, son score sera calculé selon une échelle linéaire entre 50% et 10%, où 50% correspond à $1.1 \times p$ mouvements, et 10% correspond à $2 \times p$ mouvements.
- Si votre solution utilise plus de $2 \times p$ mouvements de pile, elle obtiendra précisément 10% des points.

Par exemple, considérez un scénario d'entrée pour lequel la meilleure solution trouvée par un candidat utilise 100 mouvements de pile. Le score pour une solution valide sera calculé comme suit :

Mouvements	100	102	104	106	108	110	140	170	200	300	900
Score	100%	90%	80%	70%	60%	50%	37%	23%	10%	10%	10%

Une solution qui effectue un mouvement invalide (par exemple, un déplacement vers une case non adjacente, ou se déplacer en dehors de l'intervalle de coordonnées 1..1000) obtiendra un score de zéro. De même, une solution qui ne réunit pas toutes les feuilles en une unique pile obtiendra un score de zéro.

Si une solution effectue un mouvement valide, mais partant d'une case vide (i.e., il n'y a pas de feuilles à souffler à partir de cette case), ce mouvement inutile sera simplement ignoré (mais sera ajouté à votre nombre total de mouvements de pile).

Soumission pour une tâche à sortie uniquement

Pour soumettre vos fichiers de sortie, vous devez faire ce qui suit :

- Créez un fichier zip contenant tous vos fichiers de sortie. Vous pouvez inclure autant (ou aussi peu) de fichiers de sortie que vous le souhaitez (i.e., vous n'avez pas besoin de fournir une solution pour tous les scénarios d'entrée). Sur un système GNU/Linux, vous pouvez utiliser une commande comme la suivante :

  ```
  zip messolutions.zip leaf.*.out
  ```

  Sur un système windows, vous pouvez créer un fichier zip en s ;electionnant *Fichier* → *Nouveau* → *Dossier compressé (zippé)* à partir de l'explorateur windows, puis copier vos fichiers de sortie dans ce nouveau fichier zip.
- Soumettez ce fichier zip sur le site du FARIO, de la même manière que vous soumettriez une solution ordinaire.

Le site web va regarder à l'intérieur du fichier zip, et assurer que les fichiers de sortie sont nommés correctement, bien qu'il ne vérifiera pas le formattage ou la structure de ces fichiers.

Si vous resoumettez un fichier zip différent, **l'ancien fichier zip sera supprimé**. Par exemple, supposez que vous soumettez un fichier zip avec les solutions des cinq premiers scénarios, et que vous trouvez ensuite une solution au sixième. Votre nouvelle soumission doit contenir **les six fichiers de sortie**, car lorsque vous soumettez vos nouvelles solutions, les anciennes seront supprimées.

Veuillez contacter les organisateurs via `fario@fario.org` si vous avez des questions au sujet de cette procédure.

Problème 4
Escalade

Limites de temps et de mémoire : 1 seconde, 64 Mo

Pour des raisons encore inconnues, l'escalade en salle est en train de devenir un passe-temps populaire parmi de nombreux ex-philatélistes, philantropes et philosophes. Vous avez vous-mêmes decidé de suivre la tendance, et commencez à vous préparer à devenir le champion du monde d'escalade. L'escalade en salle consiste à aller du bas d'un mur au sommet en n'utilisant que votre force pour monter, et en faisant tout votre possible pour ne pas tomber. Près du sommet se trouve une cloche que vous sonnez pour faire part au monde de votre exploit.

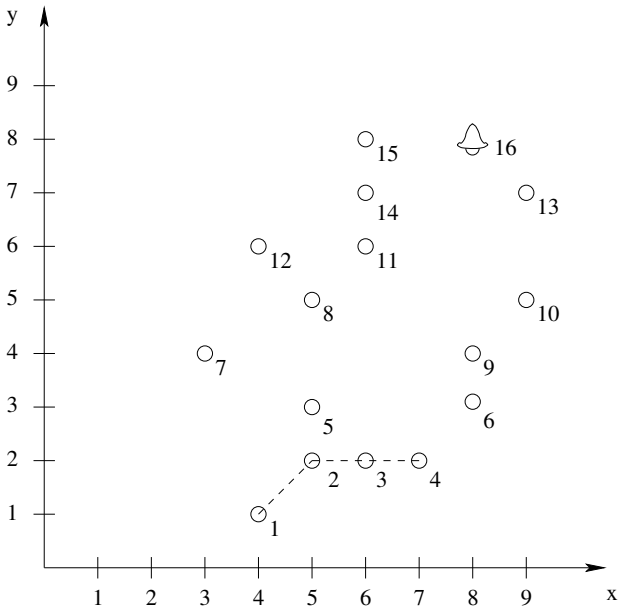
Le mur est une surface plane à deux dimensions, contenant des prises en divers points. Pour s'accrocher au mur, vous devez être aggripé à trois prises distinctes quand vous êtes stables, ou à deux prises distinctes quand vous vous déplacez. Si vous en avez moins, vous tombez bêtement !

Pour vous déplacer sur le mur, vous pouvez lâcher l'une de ces trois prises, puis attraper une nouvelle prise. Le temps nécessaire pour effectuer ce déplacement est égal à la somme des distances horizontales et verticales entre la prise que vous avez lâchée, et celle que vous attrapez.

Comme vous ne pouvez atteindre qu'une certaine distance, chacune des trois prises auxquelles vous êtes agrippé doit être à au plus R mètres d'une autre prise que vous tenez, ceci mesuré par la somme des distances horizontales et verticales entre les deux points. Notez qu'elles n'ont pas à être *toutes* à R mètres les unes des autres. Par exemple, la configuration de départ du diagramme ci-dessous représentée par les lignes pointillées) a une distance de 2 entre les points 1 et 2, une distance de 2 entre les points 2 et 4, mais une distance de 4 entre les points 1 est 4. Ceci est une configuration valide même si $R = 2$.

Pour devenir le grimpeur le plus rapide au monde, vous devez déterminer un ensemble de déplacements qui vous mènera à la cloche aussi vite que possible. La cloche est située sur l'une des prises (mais n'est pas nécessairement la plus haute prise). On vous indique sur quelles prises vous commencez à grimper, et on vous garantit que celles-ci correspondent à une situation valide.

Etant donné (i) l'ensemble des prises d'un mur décrites par des coordonnées (x, y) sur un plan et (ii) vos prises de départ, votre objectif est de trouver le temps le plus petit nécessaire pour aller de la position de départ à la cloche.



Par exemple, considérez le diagramme ci-dessus avec $R = 2$. Chaque cercle représente une prise. Il y a 16 prises (en comptant celle avec la cloche). Supposez que l'on vous demande de commencer agrippé aux prises 1, 2 et 4 comme indiqué. Une manière possible d'atteindre la cloche serait :

- lâcher la prise 4 et attrapper la prise 3 (1 seconde) ;
- lâcher la prise 1 et attrapper la prise 5 (3 secondes) ;
- lâcher la prise 2 et attrapper la prise 8 (3 secondes) ;
- lâcher la prise 3 et attrapper la prise 11 (4 secondes) ;
- lâcher la prise 5 et attrapper la prise 15 (6 secondes) ;
- lâcher la prise 8 et attrapper la prise 16 (6 secondes) ;

Ceci demande un total de $1+3+3+4+6+6 = 23$ secondes, et est en fait le moyen le plus rapide d'atteindre la cloche.

Contraintes

- $1 \leq N \leq 50$, où N est le nombre de prises sur le mur ;
- $1 \leq R \leq 6$, où R est la distance que vous pouvez atteindre (i.e. la somme maximale de la distance horizontale et verticale entre chaque point que vous tenez, et le point le plus proche que vous tenez) ;
- $-1000 \leq x, y \leq 1000$, où x et y sont les coordonnées d'une prise.

Pour 50% des points, $N \leq 20$.

Entrée

Votre programme doit lire sur l'entrée standard. La première ligne de l'entrée contiendra les entiers N et R, séparés par un espace. Les N prises seront numérotées $1 \ldots N$. La deuxième ligne contiendra les trois entiers distincts A, B et C qui indiquent les trois prises à partir desquelles vous commencez ($1 \leq A, B, C \leq N$).

Les N lignes suivantes décrivent les positions des prises. La $i$ème de ces lignes décrit la position de la $i$ème prise, par deux entiers x_i et y_i. Deux prises n'ont jamais la même position. La cloche est toujours attachée à la Nième prise.

Sortie

Votre programme doit écrire une ligne sur la sortie standard.

Cette ligne doit contenir un simple entier, qui donne le temps le plus petit nécessaire pour aller de la position de départ à la cloche. On vous garantit qu'il sera toujours possible d'atteindre la cloche.

Exemple d'entrée

```
16 2
1 2 4
4 1
5 2
6 2
7 2
5 3
8 3
3 4
5 5
8 4
9 5
6 6
4 6
9 7
6 7
6 8
8 8
```

Exemple de sortie

```
23
```

Score

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