

CEOI 2026



Jour 2
(français)

Day 2
(French)

Découpe de fleurs

Limite de temps: 4 s Limite de mémoire: 256 MiB

Dans le *BEACH* communautaire du CEOI, nous cultivons une collection de fleurs spéciales qui entrelacent étroitement leurs racines. Si nous les coupons, elles repoussent, pour autant que nous ne coupons pas de manière trop agressive et ne causions pas de dégâts irréparables.

Si une paire de fleurs, a et b , entrelacent leurs racines, nous disons qu'elles sont « connectées ». Sinon, elles sont « déconnectées ». Les racines poussent selon la règle suivante : soient a et b deux fleurs déconnectées. S'il existe au moins 2 autres fleurs c et d telles que chacune des fleurs a et b est connectée à la fois à c et à d , alors des racines pousseront entre a et b , et elles deviendront connectées.

Ces fleurs poussent maintenant depuis un certain temps, et toutes les racines qui pouvaient pousser en suivant la règle ci-dessus ont poussé. Autrement dit, si deux fleurs a et b sont toutes deux connectées à certaines fleurs c et d , alors a et b sont assurément connectées entre elles.

Nous devons déraciner ce *BEACH* et le déplacer vers le prochain site du CEOI. Afin de simplifier la migration, nous aimerions couper autant de racines que possible. Cependant, nous voulons que les fleurs repoussent pour retrouver leur état actuel. Quel est le nombre maximal de paires de fleurs connectées que nous pouvons couper pour qu'elles repoussent tout de même jusqu'à retrouver leur état actuel ? Le nombre d'itérations de croissance nécessaires n'a pas d'importance.

Entrée

La première ligne de l'entrée contient deux entiers séparés par une espace, n et m , soit le nombre de fleurs et le nombre de connexions existantes entre elles. Suivent m lignes, chacune contenant une paire d'entiers a_i et b_i , indiquant que les fleurs a_i et b_i sont connectées. Les fleurs sont numérotées par les entiers $1 \dots n$. Il est garanti que l'entrée respecte la règle décrite dans l'énoncé de la tâche.

Contraintes

- $1 \leq n \leq 1000$
- $1 \leq m \leq 10^5$

Sous-tâches

- Sous-tâche 1 (20 points) : $n \leq 10$ et $m \leq 20$
- Sous-tâche 2 (14 points) : $m = \frac{n \cdot (n-1)}{2}$
- Sous-tâche 3 (15 points) : nous garantissons que chaque fleur est connectée à au plus 7 autres fleurs.
- Sous-tâche 4 (15 points) : $n \leq 50$ et $m \leq 1000$
- Sous-tâche 5 (36 points) : aucune contrainte supplémentaire.

Sortie

Affichez un seul entier — le nombre maximal de connexions que nous pouvons couper.

Exemple

Entrée

9 14

1 2

1 4

1 5

2 4

2 5

3 4

4 5

3 6

4 6

6 7

6 9

7 9

8 9

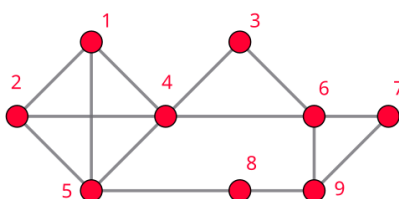
5 8

Sortie

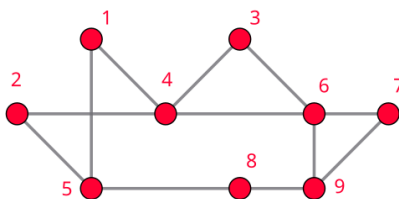
2

Commentaire

Les fleurs de l'exemple avant toute découpe. On peut vérifier qu'aucune connexion supplémentaire ne peut se former selon la procédure de croissance décrite.



Les fleurs de l'exemple après la découpe ont 2 connexions de moins. La connexion entre 1 et 2 peut repousser parce que les fleurs 1 et 2 sont toutes deux connectées aux fleurs 4 et 5. De même, la connexion entre 4 et 5 peut repousser parce que les fleurs 4 et 5 sont toutes deux connectées aux fleurs 1 et 2.



Tours

Limite de temps: 1 s Limite de mémoire: 256 MiB

Vous disposez de n ordinateurs et de m tours, tous situés à des positions distinctes le long d'une droite. Vous devez apparier les ordinateurs à l'aide de câbles de telle sorte que chaque câble parte d'un ordinateur, visite certaines tours et se termine à un autre ordinateur. Le câble peut visiter n'importe laquelle des tours (pas seulement celles situées entre les ordinateurs) dans un ordre arbitraire. Il peut ignorer des tours en passant à côté d'elles sans les visiter. Il peut aussi ne visiter aucune tour et relier directement les deux ordinateurs, mais il ne peut pas relier un ordinateur à lui-même. Le nombre d'ordinateurs est pair.

Soient a et b les positions de deux ordinateurs et soient $x_1, \dots, x_k$ les positions des tours visitées par le câble. La longueur du câble est $|a - x_1| + |x_1 - x_2| + \dots + |x_{k-1} - x_k| + |x_k - b|$. Pour un câble donné, on définit son score comme $f \cdot u - l$, où l est sa longueur, f est une constante fixée et u est le nombre de tours distinctes que le câble visite parmi $x_1, \dots, x_k$. Plusieurs câbles peuvent visiter la même tour, et cette tour contribue alors au score de chacun de ces câbles.

Vous devez calculer la somme maximale possible des scores des câbles permettant d'apparier tous les ordinateurs (c'est-à-dire que chaque ordinateur doit appartenir à exactement une paire, ou de manière équivalente, être relié à exactement un câble).

Entrée

La première ligne contient le nombre de cas de test, T . Les cas de test se suivent les uns après les autres. Chaque cas de test est constitué de trois lignes. La première ligne contient trois entiers n , m et f — le nombre d'ordinateurs, le nombre de tours et la constante f . La deuxième ligne contient n entiers $a_1, a_2, \dots, a_n$ — les positions des ordinateurs. La troisième ligne contient m entiers $b_1, b_2, \dots, b_m$ — les positions des tours.

Contraintes

Soient N et M respectivement la somme des n et la somme des m sur l'ensemble des cas de test.

- $1 \leq T \leq 10^4$
- $1 \leq N, M \leq 2 \cdot 10^5$
- $0 \leq f \leq 10^9$
- n est pair
- $1 \leq a_i, b_i \leq 10^9$
- Toutes les positions des ordinateurs et des tours sont distinctes (au sein d'un même cas de test).

Sous-tâches

- Sous-tâche 1 (5 points) : $N \leq 5000$, $m = 1$

- Sous-tâche 2 (10 points) : $T \leq 20$, $n \leq 10$, $m \leq 100$
- Sous-tâche 3 (27 points) : $N, M \leq 5000$
- Sous-tâche 4 (21 points) : $N \leq 5000$
- Sous-tâche 5 (37 points) : Aucune contrainte supplémentaire.

Sortie

Affichez T entiers, chacun sur sa propre ligne — la somme maximale possible des scores des câbles pour chaque cas de test.

Exemple

Entrée	Sortie
4	89
2 1 100	-4
1 10	12
11	51
4 1 10	
2 4 6 8	
20	
4 1 10	
2 4 6 8	
5	
6 3 10	
2 13 4 8 6 10	
5 1 9	

Vim

Limite de temps: 1 s Limite de mémoire: 256 MiB

Il serait difficile de trouver quelque chose de plus *geek* que le bon vieux Vim (à l'exception, bien sûr, de l'indémodable Emacs) — un éditeur dans lequel on peut faire des choses dont on ne pourrait que rêver dans des éditeurs « classiques ». Enfin, à condition d'accepter d'investir une semaine d'apprentissage et un mois de pratique afin que les commandes inhabituelles mais puissantes qu'il propose deviennent notre « mémoire musculaire ». Eh bien, tout comme dans les autres éditeurs, Vim dispose d'un *curseur*, qui se trouve toujours sur l'un des caractères. Au départ, il se trouve sur le premier caractère du texte, et tout comme dans les autres éditeurs, Vim dispose d'un *presse-papier*, destiné à stocker (et, indirectement, à copier et à déplacer) des morceaux de texte. Le presse-papier est initialement vide.

Dans cette tâche, nous supposons qu'il y a exactement un caractère '-' (moins) dans l'éditeur au départ, et notre objectif est de produire exactement n signes moins consécutifs. Nous utiliserons quatre commandes pour nous y aider :

- **h** : Si le curseur se trouve sur le premier caractère, alors cette commande ne fait rien ; sinon, elle déplace le curseur sur le caractère situé à gauche.
- **l** : Si le curseur se trouve sur le dernier caractère, alors cette commande ne fait rien ; sinon, elle déplace le curseur sur le caractère situé à droite.
- **Y** : La suite de caractères s'étendant de la position du curseur jusqu'à la fin du texte est copiée dans le presse-papier, « écrasant » tout contenu précédent du presse-papier. (Si le curseur se trouve sur le premier caractère du texte, le texte entier sera copié dans le presse-papier.)
- **P** : Une copie du texte stocké dans le presse-papier est insérée avant le caractère sur lequel se trouve le curseur, et le curseur est déplacé sur le dernier caractère inséré. Le contenu du presse-papier n'est pas modifié. Si le presse-papier est vide, il ne se passe rien.

Écrivez un programme qui lit le nombre n et affiche le nombre minimal de commandes nécessaires pour terminer avec exactement n caractères '-' consécutifs dans Vim, dans les conditions décrites. Le programme doit également afficher un exemple de suite comportant le nombre minimal de commandes.

Entrée

La première ligne contient le nombre de cas de test t . Les t lignes suivantes contiennent les cas de test avec la longueur de chaîne souhaitée n .

Contraintes

- $1 \leq t \leq 100$
- $1 \leq n \leq 10^7$

Sous-tâches

Si vous affichez uniquement le nombre correct de commandes, mais aucun exemple ou un exemple incorrect de suite de commandes, vous recevrez la moitié des points pour cette sous-tâche.

- Sous-tâche 1 (20 points) : $n \leq 100$
- Sous-tâche 2 (8 points) : $n \leq 1000$
- Sous-tâche 3 (18 points) : $n \leq 10^4$
- Sous-tâche 4 (18 points) : $n \leq 10^5$
- Sous-tâche 5 (18 points) : $n \leq 10^6$
- Sous-tâche 6 (18 points) : Aucune contrainte supplémentaire.

Sortie

Pour chaque cas de test, affichez le nombre minimal de commandes requis ainsi qu'un exemple d'une telle suite de commandes, chacun sur sa propre ligne.

Exemple

Entrée

2
21
2

Sortie

10 YPYPhPYPPP
2 YP

Commentaire

Comme nous pouvons le voir dans le tableau suivant, dans le premier cas, la suite YPYPhPYPPP donne le résultat correct. Le caractère '=' dans la colonne Écran représente le caractère '-' sur lequel se trouve le curseur.

Étape	Commande	Écran	Presse-papier
0		=	(vide)
1	Y	=	-
2	P	=-	-
3	Y	=--	--
4	P	=---	--
5	h	=----	--
6	P	=-----	--
7	Y	=-----	-----
8	P	-----=-----	-----
9	P	-----=-----	-----
10	P	-----=-----	-----