# Problem S1: Voronoi Villages

### Problem Description

In the country of Voronoi, there are $N$ villages, located at distinct points on a straight road. Each of these villages will be represented by an integer position along this road.

Each village defines its *neighbourhood* as all points along the road which are closer to it than to any other village. A point which is equally close to two distinct villages $A$ and $B$ is in the neighbourhood of $A$ and also in the neighbourhood of $B$.

Each neighbourhood has a *size* which is the difference between the minimum (leftmost) point in its neighbourhood and the maximum (rightmost) point in its neighbourhood.

The neighbourhoods of the leftmost and rightmost villages are defined to be of infinite size, while all other neighbourhoods are finite in size.

Determine the smallest size of any of the neighbourhoods (with exactly 1 digit after the decimal point).

### Input Specification

The first line will contain the number $N$ ($3 \leq N \leq 100$), the number of villages. On the next $N$ lines there will be one integer per line, where the $i$th line contains the integer $V_i$, the position of the $i$th village ($-1\,000\,000\,000 \leq V_i \leq 1\,000\,000\,000$). All villages are at distinct positions.

### Output Specification

Output the smallest neighbourhood size with exactly one digit after the decimal point.

### Sample Input

```
5
16
0
10
4
15
```

### Output for Sample Input

```
3.0
```

### Explanation for Output for Sample Input

The neighbourhoods around $0$ and $16$ are infinite. The neighbourhood around $4$ is $5$ units ($2$ to the left, and $3$ to the right). The neighbourhood around $10$ is $5.5$ units ($3$ to the left and $2.5$ to the right). The neighbourhood around $15$ is $3.0$ units ($2.5$ to the left and $0.5$ to the right).


Version française sont après la version anglaise

# Problème S1 : Villages de Voronoï

**Description du problème**
Dans le pays de Voronoï, il y a $N$ villages, situés à des points distincts sur une route droite. Chacun de ces villages sera représenté par la position d'un entier sur cette droite.

Chaque village a un *voisinage* composé de tous les points sur la droite qui sont plus près de lui que de tout autre village. Un point qui est à la même distance des villages distincts $A$ et $B$ se trouve dans le voisinage de $A$ et dans le voisinage de $B$.

Chaque voisinage a une *grandeur* qui est égale à la différence entre le point minimum (le plus à gauche) de son voisinage et le point maximum (le plus à droite) de son voisinage.

Les villages le plus à gauche et le plus à droite ont des voisinages de grandeur infinie, tandis que tous les autres villages ont un voisinage de grandeur finie.

Déterminer la grandeur de voisinage la plus petite (en écrivant exactement un chiffre après le point décimal).

**Précisions par rapport aux entrées**
La première ligne contiendra un nombre $N$ ($3 \leq N \leq 100$), soit le nombre de villages. Chacune des $N$ lignes suivantes contiendra un entier, la $i^{\text{ième}}$ ligne contenant $V_i$, la position du $i^{\text{ième}}$ village, ($-1\,000\,000\,000 \leq V_i \leq 1\,000\,000\,000$). Les villages ont tous des positions distinctes.

**Précisions par rapport aux sorties**
La sortie sera la plus petite grandeur de voisinage, avec exactement un chiffre après le point décimal.

**Exemple d'entrée**
```
5
16
0
10
4
15
```
**Sortie pour l'exemple d'entrée**
```
3.0
```

**Explication de la sortie pour l'exemple d'entrée**
Les voisinages autour de $0$ et de $16$ ont une grandeur infinie. Le voisinage autour de $4$ a une grandeur de $5$ ($2$ vers la gauche et $3$ vers la droite). Le voisinage autour de $10$ a une grandeur de $5.5$ ($3$ vers la gauche et $2.5$ vers la droite). Le voisinage autour de $15$ a une grandeur de $3.0$ ($2.5$ vers la gauche et $0.5$ vers la droite).


English version appears before the French version

# Problem J4/S2: Sunflowers

**Problem Description**

Barbara plants $N$ different sunflowers, each with a unique height, ordered from smallest to largest, and records their heights for $N$ consecutive days. Each day, all of her flowers grow taller than they were the day before.

She records each of these measurements in a table, with one row for each plant, with the first row recording the shortest sunflower's growth and the last row recording the tallest sunflower's growth. The leftmost column is the first measurement for each sunflower, and the rightmost column is the last measurement for each sunflower.

If a sunflower was smaller than another when initially planted, it remains smaller for every measurement.

Unfortunately, her children may have altered her measurements by rotating her table by a multiple of 90 degrees.

Your job is to help Barbara determine her original data.

**Input Specification**

The first line of input contains the number $N$ ($2 \leq N \leq 100$). The next $N$ lines each contain $N$ positive integers, each of which is at most $10^9$. It is guaranteed that the input grid represents a rotated version of Barbara's grid.

**Output Specification**

Output Barbara's original data, consisting of $N$ lines, each of which contain $N$ positive integers.

**Sample Input 1**

```
2
1 3
2 9
```

**Output for Sample Input 1**

```
1 3
2 9
```

**Explanation of Output for Sample Input 1**

The data has been rotated a multiple of 360 degrees, meaning that the input arrangement is the original arrangement.

Version française sont après la version anglaise

**Sample Input 2**

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

**Output for Sample Input 2**

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

**Explanation of Output for Sample Input 2**

The original data was rotated 90 degrees to the right/clockwise.

**Sample Input 3**

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

**Output for Sample Input 3**

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

**Explanation of Output for Sample Input 3**

The original data was rotated 90 degrees to the left/counter-clockwise.

Version française sont après la version anglaise

# Problème J4/S2 : Tournesols

**Description du problème**
Barbara plante $N$ tournesols, du plus petit au plus grand, les tailles des tournesols étant toutes différentes les unes des autres. Chaque jour, pendant $N$ jours consécutifs, elle note la taille de chaque tournesol. Chaque jour, la taille de chaque tournesol augmente.

Elle inscrit les tailles dans un tableau, une rangée par tournesol. La première rangée contient les tailles du plus petit tournesol et la dernière rangée contient celles du plus grand tournesol. La colonne la plus à gauche contient la plus petite taille de chaque tournesol et la colonne la plus à droite contient la plus grande taille de chaque tournesol.

Si un tournesol était plus petit qu'un autre au départ, il demeure plus petit chaque jour.

Malheureusement, il se peut que les enfants de Barbara aient changé les résultats en faisant subir au tableau une rotation d'un multiple de 90 degrés.

Vous avez pour mission d'aider Barbara à récupérer ses données initiales.

**Précisions par rapport aux entrées**
La première ligne d'entrées contiendra le nombre $N$ ($2 \leq N \leq 100$). Les $N$ lignes d'entrées suivantes contiendront chacune $N$ entiers strictement positifs, chacun ne dépassant pas $10^9$. Il est garanti que le tableau d'entrées représente une version du tableau initial de Barbara après une rotation.

**Précisions par rapport aux sorties**
La sortie représentera le tableau initial de Barbara sous la forme de $N$ lignes contenant chacune $N$ entiers strictement positifs.

**Exemple d'entrée 1**
```
2
1 3
2 9
```

**Sortie pour l'exemple d'entrée 1**
```
1 3
2 9
```

**Explication de la sortie pour l'exemple d'entrée 1**
Le tableau de Barbara a subi une rotation d'un multiple de 360 degrés et le tableau d'entrée est donc le tableau initial de Barbara.

English version appears before the French version

**Exemple d'entrée 2**

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

**Sortie pour l'exemple d'entrée 2**

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

**Explication de la sortie pour l'exemple d'entrée 2**

Le tableau de Barbara a subi une rotation de 90 degrés vers la droite (dans le sens des aiguilles d'une montre).

**Exemple d'entrée 3**

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

**Sortie pour l'exemple d'entrée 3**

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

**Explication de la sortie pour l'exemple d'entrée 3**

Le tableau de Barbara a subi une rotation de 90 degrés vers la gauche (dans le sens contraire des aiguilles d'une montre).

English version appears before the French version

# Problem S3: RoboThieves

**Problem Description**

A robot has stolen treasure from a factory and needs to escape without getting caught. The factory can be modelled by an $N$ by $M$ grid, where the robot can move up, down, left, or right.

Each cell of the grid is either empty, a wall, a camera, a conveyor, or the robot's initial position. The robot can only walk on empty cells (denoted by `.`) or conveyors. The first row, last row, first column and last column of the grid consists of walls (denoted by `W`), and there may be walls in other cells.

Conveyors cause the robot to move in a specific direction, denoted by `L`, `R`, `U`, `D` for left, right, up, down respectively. The robot is unable to move on its own while on a conveyor. It is possible that the robot can become stuck forever on conveyors.

Cameras (denoted by `C`) can see in all four directions up, down, left, and right, but cannot see through walls. The robot will be caught if it is in the same cell as a camera or is seen by a camera while on an empty cell. Conveyors are slightly elevated, so the robot cannot be caught while on a conveyor, but cameras can see empty cells on the other side of conveyors.

The robot is initially at the cell denoted by `S`. The exit could be at any of the empty cells. For each empty cell, determine the minimum number of steps needed for the robot to move there without being caught, or determine that it is impossible to move there. A step consists of moving once up, down, left or right. Being moved by a conveyor does not count as a step.

**Input Specification**

The first line of input contains two integers $N$ and $M$ ($4 \leq N, M \leq 100$). The next $N$ lines of input will each contain $M$ characters, each of which is one of the eight characters `W`, `.`, `C`, `S`, `L`, `R`, `U`, or `D`.

There will be exactly one `S` character and at least one `.` character. The first and last character of every row and column will be `W`.

For 5 of the 15 marks available, there are no cameras or conveyors.

For an additional 5 of the 15 marks available, there are no conveyors.

**Output Specification**

For each empty cell, print one line with one integer, the minimum number of steps for the robot to move to this empty cell without being caught or $-1$ if it is impossible to move to this empty cell.

The output should be in row major order; the order of empty cells seen if the input is scanned line by line top-to-bottom and then left-to-right on each line. See the sample outputs for examples of row major order output.

Version française sont après la version anglaise

**Sample Input 1**

```
4 5
WWWWW
W.W.W
WWS.W
WWWWW
```

**Output for Sample Input 1**

```
-1
2
1
```

**Explanation of Output for Sample Input 1**

The robot cannot move to the top left empty cell because it is blocked by walls.

The top right empty cell can be reached in 2 steps and the bottom right empty cell can be reached in 1 step.

**Sample Input 2**

```
5 7
WWWWWWW
WD.L.RW
W.WCU.W
WWW.S.W
WWWWWWW
```

**Output for Sample Input 2**

```
2
1
3
-1
-1
1
```

**Explanation of Output for Sample Input 2**

The empty cell to immediate left of the robot is seen by the camera so the robot cannot move there.

The empty cell right below the `R` conveyor is also seen by the camera as conveyors do not block the the sight of cameras.

Note that the robot can use the `U` and `L` conveyors to avoid the getting caught by the camera.

If the robot moves to the `R` conveyor, it will become stuck forever there.

Version française sont après la version anglaise

# Problème S3 : Robot voleur

**Description du problème**
Un robot a volé un trésor dans une usine et veut s'enfuir sans se faire prendre. L'usine peut être modélisée par un quadrillage $N$ sur $M$, sur lequel le robot peut se déplacer vers la gauche, la droite, le haut ou le bas.

Chaque cellule du quadrillage peut être vide ou être occupée par un mur, une caméra ou un convoyeur. Elle peut aussi représenter la position initiale du robot. Le robot peut seulement se déplacer vers une cellule vide (représentée par `.`) ou une cellule occupée par un convoyeur. La première et la dernière rangée, ainsi que la première et la dernière colonne du quadrillage sont remplies de murs (représentés par `W`) et il peut y avoir des murs dans les autres cellules.

Les convoyeurs déplacent le robot dans une de quatre directions particulières, soit vers la gauche (`L`), la droite (`R`), le haut (`U`) ou le bas (`D`). Le robot ne peut se déplacer de lui-même lorsqu'il se trouve sur un convoyeur. Il est possible que le robot se trouve pris pour toujours sur un convoyeur.

Des caméras (représentées par `C`) peuvent voir dans chacune des quatre directions (la gauche, la droite, le haut, le bas), mais elles ne peuvent pas voir à travers un mur. Le robot sera capturé s'il est situé dans une même cellule qu'une caméra ou s'il peut être vu par une caméra lorsqu'il est situé dans une cellule vide. Les convoyeurs sont situés sur une surface élevée, ce qui fait que le robot ne peut être capturé lorsqu'il est sur un convoyeur, mais les caméras peuvent voir les cellules vides de l'autre côté des convoyeurs.

Au départ, le robot sera situé dans une cellule représentée par `S`. La sortie du robot peut se faire dans n'importe quelle cellule vide. Pour chaque cellule vide, déterminer le nombre minimal de pas qu'il faut au robot pour s'y rendre dans être capturé, ou déterminer qu'il est impossible de s'y rendre. Un pas est un mouvement d'une cellule vers la droite, la gauche, le haut ou le bas. Un déplacement sur un convoyeur n'est pas considéré comme un pas.

**Précisions par rapport aux entrées**
La première ligne d'entrée contiendra deux entiers $N$ et $M$ ($4 \leq N, M \leq 100$). Les $N$ lignes suivantes contiendront chacune $M$ caractères, parmi les huit caractères `W`, `.`, `C`, `S`, `L`, `R`, `U`, ou `D`.

Il y aura exactement un caractère `S` et au moins un caractère `.` (cellule vide). Le premier et le dernier caractère de chaque ligne et de chaque colonne seront des `W`.

Pour 5 des 15 points disponibles, il n'y aura aucune caméra et aucun convoyeur.

Pour 5 autres des 15 points disponibles, il n'y aura aucun convoyeur.

**Précisions par rapport aux sorties**
Pour chaque cellule vide, imprimer une ligne contenant un entier, soit le nombre minimal de pas qu'il faut au robot pour s'y rendre sans être capturé ou $-1$ s'il est impossible pour le robot de se rendre dans cette cellule.

English version appears before the French version

La sortie devrait se faire en suite de lignes (row-major order) ; l'ordre des cellules vides qui sont vues lorsque l'entrée est lue ligne par ligne, du haut vers le bas, puis de gauche à droite pour chaque ligne. Voir les exemples de sorties pour des exemples de suites de lignes.

**Exemple d'entrée 1**

```
4 5
WWWWW
W.W.W
WWS.W
WWWWW
```

**Sortie pour l'exemple d'entrée 1**

```
-1
2
1
```

**Explication de la sortie pour l'exemple d'entrée 1**

Le robot ne peut pas se rendre dans la cellule vide en haut à gauche, puisque qu'elle est entourée de murs.

La cellule en haut à droite peut être atteinte en 2 pas et la cellule en bas à droite peut être atteinte en 1 pas.

**Exemple d'entrée 2**

```
5 7
WWWWWWW
WD.L.RW
W.WCU.W
WWW.S.W
WWWWWWW
```

**Sortie pour l'exemple d'entrée 2**

```
2
1
3
-1
-1
1
```

**Explication de la sortie pour l'exemple d'entrée 2**

La cellule vide immédiatement à la gauche du robot ne peut être atteinte, puisqu'elle est vue par la caméra.

English version appears before the French version

La cellule vide immédiatement au-dessous du convoyeur `R` peut être vue par la caméra, puisque le convoyeur ne bloque pas sa vue. Cette cellule ne peut donc pas être atteinte.

On remarque que le robot peut utiliser les convoyeurs `U` et `L` pour éviter d'être vu par la caméra.

Si le robot utilise le convoyeur `R`, il ne pourra plus bouger.

English version appears before the French version

# Problem S4: Balanced Trees

**Problem Description**
Trees have many fascinating properties. While this is primarily true for trees in nature, the concept of trees in math and computer science is also interesting. A particular kind of tree, a *perfectly balanced tree*, is defined as follows.

Every perfectly balanced tree has a positive integer *weight*. A perfectly balanced tree of weight $1$ always consists of a single node. Otherwise, if the weight of a perfectly balanced tree is $w$ and $w \geq 2$, then the tree consists of a root node with branches to $k$ subtrees, such that $2 \leq k \leq w$. In this case, all $k$ subtrees must be completely identical, and be perfectly balanced themselves.

In particular, all $k$ subtrees must have the same weight. This common weight must be the maximum integer value such that the sum of the weights of all $k$ subtrees does not exceed $w$, the weight of the overall tree. For example, if a perfectly balanced tree of weight $8$ has $3$ subtrees, then each subtree would have weight 2, since $2 + 2 + 2 = 6 \leq 8$.

Given $N$, find the number of perfectly balanced trees with weight $N$.

**Input Specification**
The input will be a single line containing the integer $N$ ($1 \leq N \leq 10^9$).

For $5$ of the $15$ marks available, $N \leq 1000$.

For an additional $2$ of the $15$ marks available, $N \leq 50000$.

For an additional $2$ of the $15$ marks available, $N \leq 10^6$.

**Output Specification**
Output a single integer, the number of perfectly balanced trees with weight $N$.

**Sample Input 1**
```
4
```

**Output for Sample Input 1**
```
3
```

**Explanation for Output for Sample Input 1**
One tree has a root with four subtrees of weight 1; a second tree has a root with two subtrees of weight 2; the third tree has a root with three subtrees of weight 1.

**Sample Input 2**
```
10
```

**Output for Sample Input 2**
```
13
```

Version française sont après la version anglaise

# Problème S4 : Arbres équilibrés

**Description du problème**
Les arbres ont de nombreuses propriétés fascinantes. Bien que ce soit surtout le cas des arbres dans la nature, le concept d'arbre en mathématiques et en informatique est aussi intéressant. On définit un arbre particulier, soit un *arbre parfaitement équilibré*, comme suit.

Chaque arbre parfaitement équilibré a un *poids* qui est un entier strictement positif. Un arbre parfaitement équilibré, avec un poids de $1$, est un arbre constitué d'exactement un noeud. Autrement, si un arbre parfaitement équilibré a une poids égal à $p$, avec $p \geq 2$, alors l'arbre est constitué d'une racine avec des branches vers $k$ sous-arbres, de manière que $2 \leq k \leq p$. Dans ce cas, tous les $k$ sous-arbres doivent être parfaitement identiques et parfaitement équilibrés eux-mêmes.

En particulier, tous les $k$ sous-arbres doivent avoir le même poids. Ce poids commun doit être la valeur entière maximale telle que la somme des poids des $k$ sous-arbres ne dépasse pas $p$, le poids de l'arbre global. Par exemple, si un arbre parfaitement équilibré a un poids de $8$ et $3$ sous-arbres, alors chaque sous-arbre aura un poids de $2$, puisque $2 + 2 + 2 = 6 \leq 8$.

Étant donné $N$, déterminer le nombre d'arbres parfaitement équilibrés qui ont un poids de $N$.

**Précisions par rapport aux entrées**
L'entrée sera constituée d'une ligne contenant l'entier $N$ ($1 \leq N \leq 10^9$).

Pour $5$ des $15$ points disponibles, on aura $N \leq 1000$.

Pour $2$ autres des $15$ points disponibles, on aura $N \leq 50000$.

Pour $2$ autres des $15$ points disponibles, on aura $N \leq 10^6$.

**Précisions par rapport aux sorties**
La sortie sera un entier qui représente le nombre d'arbres parfaitement équilibrés qui ont un poids de $N$.

**Exemple d'entrée 1**
```
4
```

**Sortie pour l'exemple d'entrée 1**
```
3
```

**Explication de la sortie pour l'exemple d'entrée 1**
Un arbre a une racine avec quatre sous-arbres qui ont un poids de 1 ; un deuxième arbre a une racine avec deux sous-arbres qui ont un poids de 2 ; le troisième arbre a une racine avec trois sous-arbres qui ont un poids de 1.

**Exemple d'entrée 2**
```
10
```

**Sortie pour l'exemple d'entrée 2**
```
13
```

English version appears before the French version

# Problem S5: Maximum Strategic Savings

**Problem Description**
A long time ago in a galaxy far, far away, there are $N$ planets numbered from 1 to $N$. Each planet has $M$ cities numbered from 1 to $M$. Let city $f$ of planet $e$ be denoted as $(e, f)$.

There are $N \times P$ two-way flights in the galaxy. For every planet $e$ ($1 \leq e \leq N$), there are $P$ flights numbered from 1 to $P$. Flight $i$ connects cities $(e, a_i)$ and $(e, b_i)$ and costs $c_i$ energy daily to maintain.

There are $M \times Q$ two-way portals in the galaxy. For all cities with number $f$ ($1 \leq f \leq M$), there are $Q$ portals numbered from 1 to $Q$. Portal $j$ connects cities $(x_j, f)$ and $(y_j, f)$ and costs $z_j$ energy daily to maintain.

It is possible to travel between any two cities in the galaxy using only flights and/or portals.

Hard times have fallen on the galaxy. It was decided that some flights and/or portals should be shut down to save as much energy as possible, but it should remain possible to travel between any two cities afterwards.

What is the maximum sum of energy that can be saved daily?

**Input Specification**
The first line contains four space-separated integers $N$, $M$, $P$, $Q$ ($1 \leq N, M, P, Q \leq 10^5$).

Then $P$ lines follow; the $i$-th one contains three space-separated integers $a_i$, $b_i$, $c_i$ ($1 \leq a_i, b_i \leq M, 1 \leq c_i \leq 10^8$).

Then $Q$ lines follow; the $j$-th one contains three space-separated integers $x_j$, $y_j$, $z_j$ ($1 \leq x_j, y_j \leq N, 1 \leq z_j \leq 10^8$).

It is guaranteed that it will be possible to travel between any two cities using flights and/or portals. There may be multiple flights/portals between the same pair of cities or a flight/portal between a city and itself.

For 2 of the 15 available marks, $P, Q \leq 100$ and $c_i = 1$ for all $1 \leq i \leq P$, and $z_j = 1$ for all $1 \leq j \leq Q$.

For an additional 2 of the 15 available marks, $P, Q \leq 200$.

For an additional 5 of the 15 available marks, $N, M \leq 200$.

**Output Specification**
Output a single integer, the maximum sum of energy that can be saved daily.

Version française sont après la version anglaise

**Sample Input 1**

```
2 2 1 2
1 2 1
2 1 1
2 1 1
```

**Output for Sample Input 1**

```
3
```

**Sample Input 2**

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

**Output for Sample Input 2**

```
41
```

**Explanation for Output for Sample Input 2**

One possible way is to shut down the flights between cities $(1, 1)$ and $(1, 1)$, $(2, 1)$ and $(2, 1)$, $(1, 1)$ and $(1, 2)$, $(1, 3)$ and $(1, 2)$, $(2, 3)$ and $(2, 2)$, and shut down the portal between cities $(2, 3)$ and $(1, 3)$ for total energy savings of $8 + 8 + 6 + 7 + 7 + 5 = 41$.

Version française sont après la version anglaise

# Problème S5 : Épargne maximale stratégique

**Description du problème**
Il y a très longtemps, dans une galaxie très très lointaine, il y avait $N$ planètes numérotées de 1 à $N$. Sur chaque planète il y avait $M$ villes numérotées de 1 à $M$. La ville $f$ de la planète $e$ sera notée $(e, f)$.

Il y a $N \times P$ vols aller-retour dans la galaxie. Pour chaque planète $e$ $(1 \leq e \leq N)$, il y a $P$ vols numérotés de 1 à $P$. Le vol $i$ relie les villes $(e, a_i)$ et $(e, b_i)$ et coute $c_i$ unités d'énergie par jour.

Il y a $M \times Q$ portails à deux sens dans la galaxie. Pour chaque ville numéro $f$ $(1 \leq f \leq M)$, il y a $Q$ portails numérotés de 1 à $Q$. Le portail $j$ relie les villes $(x_j, f)$ et $(y_j, f)$ et coute $z_j$ unités d'énergie par jour.

Il est possible de voyager entre n'importe quelles deux villes dans la galaxie en n'utilisant que des vols et/ou des portails.

La galaxie connait présentement des moments difficiles. Il a été décidé d'abolir certains vols et/ou certains portails pour épargner autant d'énergie que possible, tout en permettant les voyages entre n'importe quelles deux villes après ces mesures.

Quelle est la quantité maximale d'énergie qui peut être épargnée par jour ?

**Précisions par rapport aux entrées**
La première ligne d'entrées contiendra quatre entiers, $N$, $M$, $P$ et $Q$ ($1 \leq N, M, P, Q \leq 10^5$) séparés d'une espace.

Suivront $P$ lignes ; la $i^{\text{ième}}$ ligne contiendra trois entiers, $a_i$, $b_i$ et $c_i$ ($1 \leq a_i, b_i \leq M, 1 \leq c_i \leq 10^8$) séparés d'une espace.

Suivront $Q$ lignes ; la $j^{\text{ième}}$ ligne contiendra trois entiers, $x_j$, $y_j$, $z_j$ ($1 \leq x_j, y_j \leq N, 1 \leq z_j \leq 10^8$) séparés d'une espace.

Il est assuré qu'il sera possible de voyager entre n'importe quelles deux villes en utilisant des vols et/ou des portails. Il peut y avoir plusieurs vols ou portails entre deux mêmes villes, ou un vol ou un portail entre une ville et elle-même.

Pour 2 des 15 points disponibles, on aura $P, Q \leq 100$ et $c_i = 1$ pour tous $1 \leq i \leq P$ et $z_j = 1$ pour tous $1 \leq j \leq Q$.

Pour 2 autres des 15 points disponibles, on aura $P, Q \leq 200$.

Pour 5 autres des 15 points disponibles, on aura $N, M \leq 200$.

**Précisions par rapport aux sorties**
La sortie sera un entier qui représente la quantité maximale d'énergie qui peut être épargnée par jour.

English version appears before the French version

**Exemple d'entrée 1**

```
2 2 1 2
1 2 1
2 1 1
2 1 1
```

**Sortie pour l'exemple d'entrée 1**

```
3
```

**Exemple d'entrée 2**

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

**Sortie pour l'exemple d'entrée 2**

```
41
```

**Explication de la sortie pour l'exemple d'entrée 2**

Une façon est de fermer les vols entre les villes $(1, 1)$ et $(1, 1)$, $(2, 1)$ et $(2, 1)$, $(1, 1)$ et $(1, 2)$, $(1, 3)$ et $(1, 2)$, $(2, 3)$ et $(2, 2)$ et de fermer les portails entre les villes $(2, 3)$ et $(1, 3)$, ce qui épargnera $(8 + 8 + 6 + 7 + 7 + 5)$ unités d'énergie, ou 41 unités d'énergie.

English version appears before the French version