LINMA1691 : Théorie des graphes
Devoir 2 : Plus courts chemins

Trouver le plus court chemin entre deux noeuds est un sujet classique en théorie des graphes débouchant sur de nombreuses applications. Dans ce devoir, nous en explorons et implémentons deux en **Python**. Plus précisément, il vous est demandé d'implémenter efficacement les fonctions *shortest_path_1* et *shortest_path_2* du fichier *template.py*. Vous ne pouvez pas importer d'autres librairies que celles déjà présentes.

1 Trouver la sortie du labyrinthe

Contexte

Vous avez trouvé la carte d'un labyrinthe présentée sous la forme d'une grille, où chaque case représente un carré d'un mètre sur un mètre. Le labyrinthe est consitué de murs ('#') et d'espaces libres ('.') ainsi que d'une entrée ('E') et une sortie ('S'). Vous vous demandez quelle est la distance minimale nécessaire pour traverser le labyrinthe de l'entrée à la sortie.

Input

L'input vous est donné sous la forme d'arguments d'une fonction à compléter : *shorthest_path_1(maze)*.

maze est un tableau à deux dimensions (liste de listes en Python) de dimension $N \times M$ décrivant le labyrinthe. Chaque élément du tableau est un des symbole suivant : '.' signifie un espace libre, '#' un mur, 'E' l'entrée du labyrinthe et 'S' la sortie du labyrinthe. Il est garanti que $4 \leq N, M \leq 10^3$.

Exemple de labyrinthe:

```
# # # # #
# E # S #
# . # . #
# . . . #
# # # # #
```

Réponse : 6

Remarque : Tous les bords du labyrinthe sont constitués de murs.

Output

Il vous est demandé de compléter la fonction et de retourner la distance (en nombre de cases) nécessaire pour sortir du labyrinthe (symbole S) à partir de l'entrée (symbole E) si c'est possible, ou -1 si c'est impossible.

2 Epreuve sportive

Contexte

Vous participez à un jeu sportif consistué de différents stands organisant une épreuve. Chaque stand est relié à certains autres stands par un réseau de sentiers et il vous est demandé d'arriver au stand d'arrivée à partir du stand de départ. Lorsque vous arrivez à un stand, vous devez réussir l'épreuve avant de pouvoir continuer. Vous voulez faire la course avec vos amis; sachant le temps qu'il vous faut pour parcourir chaque sentier et réussir chaque épreuve, déterminez le temps minimal nécessaire pour remporter le jeu.

Input

L'input vous est donné sous la forme d'arguments d'une fonction à compléter : *shortest_path_2(tasks, edges)*.

tasks est un tableau de N entiers [t_1, ..., t_N] décrivant le temps nécessaire pour réussir chaque épreuve. Chaque stand est indexé par un entier compris entre 1 et *N*. Il est garanti que $N \geq 3$.

paths est un tableau de M tuples (u, v, t) décrivant un sentier entre les stands u et v parcourable en t minutes.

Il est garanti que la durée de chaque tâche ainsi que le temps de parcours de chaque sentier est plus petit ou égal à 1000 secondes.

Vous devez vous rendre du stand 1 au stand N.

Remarque : Il peut y avoir une épreuve au stand de départ et d'arrivée

Exemple d'input :

([1, 2, 5], [(1,2,1),(1,3,10), (2,3,1)])

Réponse : 10

Output

Il vous est demandé de compléter la fonction et de retourner le temps minimal pour remporter la course.

Il est toujours possible de finir l'épreuve.

Consignes

Vous devez soumettre chaque méthode sur l'activité Inginious[1] du même nom.

Le langage de programmation est **Python 3** (version 3.5).

[1] https://inginious.info.ucl.ac.be