LINMA1691 : Théorie des graphes
Devoir 4 : flots et couplages

Comme vu en cours, les problèmes de flot et couplage ont de nombreuses applications. Dans ce devoir, nous en explorons et implémentons une en **Python**. Plus précisément, il vous est demandé d'implémenter efficacement la fonction *matching* du fichier *template.py*. Vous ne pouvez pas importer d'autres librairies que celles déjà présentes.

1 Cache-cache

Contexte

Entre deux devoirs de théorie des graphes, vous jouez à cache-cache dans les bois avec vos amis pour vous détendre. Vous connaissez toutes les cachettes et estimez qu'on peut parcourir n'importe quelle distance à vol d'oiseau. Le maitre du jeu compte les yeux fermés pendant T secondes puis élimine directement tous ceux qui ne sont pas cachés. Puisque une cachette ne peut accueillir qu'une personne à la fois, vous devez donc vous arranger pour les répartir au mieux entre vous. Sachant la position et la vitesse de chacun de vos camarades, pouvez vous déterminer le nombre maximal de vos amis qui seront cachés quand le temps sera écoulé ?

Input

L'input vous est donné sous la forme d'arguments d'une fonction à compléter : *matching(T, friends, hiding_places)*

T est le nombre de secondes que compte le maitre du jeu avant d'ouvrir les yeux ($T \geq 1$). *friends* est une liste de dimension N décrivant la position et la vitesse de vos amis. Chaque élément de la liste est un tuple (x, y, v) représentant un ami situé à la position (x, y) et courant à une vitesse v. *hiding_places* est une liste de dimension M décrivant la position des cachettes. Chaque élément du tableau est un tuple (x, y) représentant une cachette située à la position (x, y). Il est garanti que $1 \leq N, M \leq 100$.

Exemple d'input:

(3, [(0, 4, 1), (2, 3, 0.5), (10, 10, 2)], [(0, 1), (1, 4)])

Réponse : 2

Output

Il vous est demandé de compléter la fonction et de retourner le plus grand nombre de personnes cachées si les cachettes sont réparties optimalement.

Nous attendons un algorithme de complexité temporelle $\mathcal{O}(N^3)$.

Consignes

Vous devez soumettre votre implémentation de la méthode sur l'activité Inginious[1] du même nom.

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

Deadline : 4 décembre 2019, 10h30. La deadline est stricte : il n'est plus possible de soumettre après cette date.

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