Ce sujet est adapté du concours Hash Code 2014.

Introduction

Des services comme Google Street View ou Apple Look Around permettent de visualiser une adresse routière sans avoir à s'y déplacer, avec des vues à 360°le long de chaque route. Ils se basent sur des photographies prises à intervalles réguliers par des véhicules spécialisés, qui sillonnent les villes avec plusieurs appareils photographiques.

La réalisation des photographies d'une ville pose un problème d'optimisation : on dispose d'une flotte de véhicules limitée, et on souhaite parcourir autant de rues que possible avec un volume de carburant donné.

Étant donnée une description des rues de la ville et un nombre de véhicules disponibles aux réservoirs pleins, votre tâche est de planifier le mouvement des véhicules pour maximiser la longueur totale des rues couvertes au moins une fois.

Description du problème

La ville est représentée par un graphe. Les nœuds représentent les intersections et sont connectés par des arêtes représentant les rues. Chaque rue a trois propriétés :

- **direction** - chaque rue peut être à sens unique ou à double sens.
- **longueur** - la distance en mètres qu'un véhicule couvre en parcourant cette rue. Cette distance contribue au score, elle correspond à la longueur réelle de la rue (y compris les éventuelles courbes).
- **coût** - la quantité de carburant nécessaire pour parcourir la rue (qui varie en fonction de la vitesse sur cette portion, de sa pente, etc.).

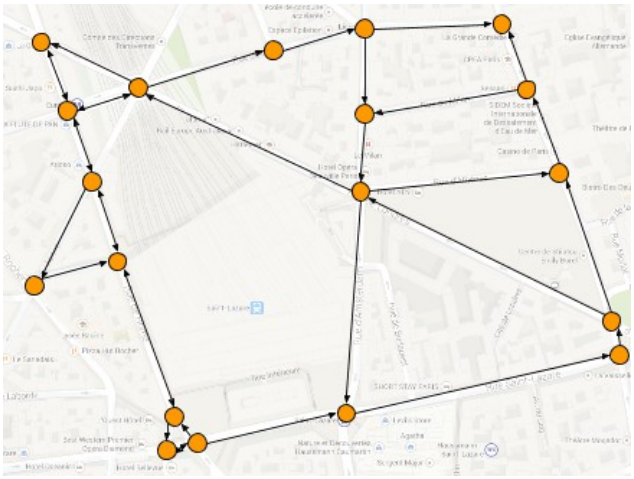


Figure 1: Une partie de la ville

Il ne peut pas y avoir plus qu'une rue connectant deux intersections, ni une rue connectant une intersection avec elle-même. Le graphe n'est pas nécessairement planaire (certaines rues peuvent se croiser à cause des ponts et des tunnels). Les intersections et les rues sont identifiées en utilisant leur indice (en commençant par l'indice 0) correspondant à l'ordre dans lequel elles apparaissent dans le fichier d'entrée (voir ci-dessous).

Vous gérez une flotte de $\boldsymbol{V}$ véhicules, tous situés à l'intersection $\boldsymbol{S}$ au début du jeu. Vous devez planifier le mouvement de chaque véhicule pour un volume d'essence initial $\boldsymbol{E}$ par véhicule (vous pouvez tout utiliser, le volume réel est plus important pour permettre le retour à une station service à la fin). Tous les déplacements planifiés dans les initéraires doivent se terminer sans dépasser V. Les déplacements suivants ne comptent pas dans le score.

Votre score est la somme des longueurs des rues qui ont été parcourues au moins une fois par une voiture de votre flotte. Parcourir une rue qui a déjà été parcourue (y compris une rue à double sens dans la direction opposée) n'augmente pas le score.

Données d'entrée

Les données d'entrée sont fournies sous la forme d'un fichier texte, composé de :

- une ligne contenant les entiers suivants séparés par des espaces :
  - $\boldsymbol{I}$ - le nombre d'intersections de la ville
  - $\boldsymbol{R}$ - le nombre de rues de la ville
  - $\boldsymbol{E}$ - le volume d'essence initial en millilitres de chaque véhicule
  - $\boldsymbol{V}$ - le nombre de véhicules de la flotte
  - $\boldsymbol{S}$ - l'indice de l'intersection à laquelle les véhicules sont initialement positionnés
- $\boldsymbol{I}$ lignes décrivant les intersections, avec pour chaque les flottants $\boldsymbol{x_i}$ et $\boldsymbol{y_i}$ $(0 \leq i < I)$ séparés par un espace, décrivant les coordonnées planaires de chaque intersection.

- $\boldsymbol{R}$ lignes décrivant les rues. La r-ième ($0 \leq r < R$) de ces lignes contient les entiers suivants séparés par des espaces :
  - $\boldsymbol{A_r}$ et $\boldsymbol{B_r}$ ($0 \leq A_r, B_r < I, A_r \neq B_r$) - les indices des intersections connectées par la rue.
  - $\boldsymbol{D_r}$ - qui vaut 1 si la rue est à sens unique de A_r à B_r, et 2 si elle est à double sens.
  - $\boldsymbol{C_r}$ - le coût en millilitres de carburant pour parcourir la rue, c'est-à-dire la quantité de carburant qu'un véhicule consommera pour aller d'un bout à l'autre.
  - $\boldsymbol{L_r}$ - la longueur de la rue en mètres. C'est le score attribué lorsque la rue est traversée pour la première fois.

Exemple de fichier d'entrée
(3 intersections, 2 rues, 3L de carburant, 2 véhicules partant de 0)

```
3 2 3000 2 0
48.8582 2.2945
50.0 3.09
51.424242 3.02
0 1 1 30 250
1 2 2 45 200
```

Résolution

Pour résoudre le problème, votre programme doit imprimer une solution textuelle, avec une ligne par véhicule. Il doit s'exécuter en moins de 10 secondes pour être validé en ligne (voir plus bas). Chaque ligne est une liste des intersections parcourues par un véhicule (y compris l'intersection de départ), dans l'ordre de leur parcours. Chaque intersection est identifiée par son indice i, qui correspond à l'ordre dans lequel elle apparaissait dans le fichier d'entrée. Les indices sont à écrire séparés par des espaces.

Exemple de données de sortie
(le premier véhicule parcourt 3 intersections, et le second reste à l'intersection de départ)

```
0 1 2
0
```

Pour être valide, une solution doit respecter les critères suivants :

- il faut autant de lignes que le nombre de véhicules $\boldsymbol{V}$ de la flotte.
- la première intersection de chaque véhicule doit être l'intersection de départ $\boldsymbol{S}$ indiquée dans le fichier d'entrée.

- pour chaque paire d'intersections consécutives dans l'initéraire d'une voiture, il doit exister une rue entre elles, et elle doit pouvoir être parcourue dans la bonne direction.
- le coût total en carburant de chaque itinéraire doit être inférieur ou égal à $\boldsymbol{E}$.

Le score d'une solution est la somme des longueurs des rues qui ont été visitées au moins une fois par un véhicule. Vous serez classés en fonction de votre meilleur score. Dans l'exemple de solution ci-dessus, les deux rues disponibles sont parcourues donc le score calculé est de 450.

Votre code pourra être envoyé en ligne sur :

http://eclope.mi90.ec-lyon.fr/problem/tc1td5

Étapes

Afin de vous aider à aborder le problème nous vous proposons de suivre les étapes suivantes. A noter que vous pouvez évaluer le temps d'exécution de votre code en utilisant le module `timeit`[1] comme suit :

```
import timeit
timeit.timeit(< votre fonction>)
```

Étape 1 – Définir des structures de données `Rue` et `Intersections`.

Étape 2 – Initialisez ces structures de données avec les données fournies. Vous pourrez ré-utiliser le code fourni ci-dessous :

```
f = open('paris_54000.txt', 'r')
I, R, E, V, S = map(int, f.readline().split())

for i in range(I):
    x, y = map(float, f.readline().split())

for i in range(R):
    a, b, d, c, l = map(int, f.readline().split())
```

Étape 3 – Un algorithme naif consiste à partir du point de départ, et ensuite prendre une rue au hasard. Commencez par planifier l'itinéraire d'une voiture seule (en laissant les autres à l'intersection de départ).

Étape 4 – Proposez une heuristique de choix de la rue meilleure que l'aléatoire. Vous calculerez le score courant de votre solution (qui vous est également renvoyé sur la plateforme pour comparer votre calcul).

Étape 5 – Planifiez à présent l'itinéraire de toutes les voitures.

Étape 6 – Une méthode plus élaborée d'optimisation consiste à partir d'une solution complète de choisir un itinéraire existant, de le supprimer et d'en générer un nouveau. N'oubliez pas que votre code sur la plateforme en ligne doit s'exécuter en moins de 10 secondes.

[1] https://docs.python.org/3/library/timeit.html

Une plateforme de test de code Python est accessible à l'adresse suivante :

http://eclope.mi90.ec-lyon.fr/

La plateforme fonctionne comme indiqué ci-dessous. Attention ! Il faudra modifier la manière dont vous lisez les données en entrée !

Étape 1 : Ouvrez la page web du site ci-dessus dans un navigateur et identifiez vous :

Login : votre nom CAS (1ere lettre du prénom + 7 premières lettres du nom de famille)
Pass : identique au nom CAS (mot de passe provisoire)

Étape 2 : Actions importantes à réaliser sur votre profil
→ https://eclope.mi90.ec-lyon.fr/edit/profile/
1) changez votre mot de passe.
2) sélectionnez votre groupe de TD dans la liste des organisations.
Sinon vous n'apparaitrez pas dans la liste des étudiants
→ exemple pour `d1a` https://eclope.mi90.ec-lyon.fr/organization/2-D1a

Étape 3 : Cliquez sur l'onglet problème pour obtenir la liste des questions de ce TD
→ https://eclope.mi90.ec-lyon.fr/problems/

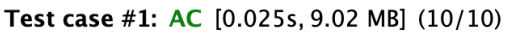
Étape 4 : Sélectionnez une question par exemple un test d'addition
→ https://eclope.mi90.ec-lyon.fr/problem/aplusb

Étape 5 : Cliquez sur `Submit solution` et copiez-collez votre code dans le champ texte
→ https://eclope.mi90.ec-lyon.fr/problem/aplusb/submit

Étape 6 : Selectionnez Python 3 et enfin cliquez sur `Submit` et le serveur exécute votre code avec différents jeux de données (via `input()`)

Étape 7 : Le serveur compare les résultats de votre code (produits par `print()`) à ceux attendus

Étape 8 : Une page affiche `Excecution Results` avec le bouton "Abort" il faut rafraichir cette page ! Votre code est correct s'il passe tous les tests et votre score est affiché.



Étape 9 : Comparez votre performance avec les autres étudiants de la promo ou de votre groupe
→ https://eclope.mi90.ec-lyon.fr/problem/aplusb/rank/