Annexe pour sujet "temps de trajet en metro"

On présente ici quelques algorithmes dont vous pourrez vous inspirer/que vous pouvez utiliser dans votre projet. Vous êtes bien entendu libres de trouver d'autres algorithmes.

Le premier algo est une exploration de graphe en profondeur. On suppose qu'il y a n stations. On suppose disposer d'un tableau de taille $n \times n$ qui code la matrice d'adjacence, autrement dit $T[i,j] = 1$ si les stations s_i et s_j sont voisines (stations consécutives d'une meme ligne), et vaut 0 sinon. On pourra tirer de cette matrice T, un tableau de listes, qui donne pour chaque station, la liste de ses voisins (liste toujours non vide).

A partir de T, ou du tableau (longueur n) de listes , on pourra écrire un algorithme qui étant donnée une station A (disons que A est la j-eme station), renvoie un tableau *dist* de longueur n, qui indique la distance à la station A, sur le graphe.

Sans faire appel à l'algo ci-dessus (qui prend du temps puisqu'il explore tout le graphe), on pourra écrire un algo prenant en entrée deux stations A et B, un tableau indiquant pour chaque station la liste de ses voisins, et qui renvoie la distance de A à B, sur le graphe (voire qui renvoie un plus court chemin de A à B sur le graphe). Pour cette seconde question, on pourra utiliser une exploration en profondeur, ou en largeur.

Les algorithmes esquissés ci-dessus peuvent etre utilisés pour répondre à la question : peut-on se rendre de A à B en effectuant au plus k changements ? pour ce faire, utiliser le graphe G dont les sommets sont les différentes lignes de metro (14 sommets pour nous), et dont les aretes sont les changements possibles. Il n'est pas forcément utile de mettre des multi-aretes : par exemple Madeleine et Saint-Lazare sont deux aretes possibles qui lient $L12$ à $L14$, mais pour la question d'au plus k changements, toutes les aretes liant $L12$ à $L14$ peuvent etre confondues.

Si A est sur la ligne 1 et sur la 1 uniquement, et si B est sur la ligne 11 et la 11 uniquement, alors il existe un trajet de A à B en au plus k changements, ssi il existe un chemin de longueur au plus k allant de $L1$ à $L11$, sur le *graphe des lignes.* (Lorsque A ou B est sur plusieurs lignes à la fois, c'est légèrement plus compliqué.)

Le second algo est l'algorithme de Djiskra[1]. On pourra l'implémenter sur le graphe de son choix : ou bien un graphe G où figurent toutes les stations de metro, et où chaque correspondance figure autant de fois qu'elle n'a de lignes (exemple : il y a trois sommets *Charles de Gaulle Etoile* : CDG2, CDG1 et CDG6). Une arete liant deux stations consécutives d'une meme ligne est étiquetée par $1,5$ (temps moyen en minutes entre deux stations), tandis qu'une arete entre deux stations du meme nom, indique le temps moyen pour aller d'un quai à l'autre (exemple CDG6 vers CDG2 : 3 minutes).

Ou bien on pourra implémenter une variante[2] de Djiskra sur un graphe *ad hoc*, par exemple un graphe[3] dont les sommets soient les stations qui sont à l'intersection d'au moins deux lignes[4]. Puis utiliser ce premier graphe pour répondre à la question posée (plus court chemin en temps) sur le graphe initial (avec toutes les stations).

[1] autrefois au programme du lycée. https://www.youtube.com/watch?v=MybdP4kice4

[2] attention votre variante doit tenir compte des temps de changements

[3] sur ce graphe, les stations *Motte-Picquet* et *Pasteur* sont voisines, et leur arete pèse $4,5$. De meme pour *Motte-Picquet* et *Invalides.* Cependant lorsque vous implémentez Djiskra sur ce graphe, attention que le temps de Pasteur à Invalides est plus que 9 ...

[4] autrement dit : toutes les stations de correpsondance

Enfin, on pourra envisager un peu de programmation dynamique, en se limitant par exemple aux trajets avec au plus 3 changements. Pour cet algorithme, on pourra se contenter du modèle simplifié fourni, où les lignes sont identiques qu'on les prenne dans un sens ou dans l'autre. On aura alors quatre tableaux tab_0, tab_1, tab_2, tab_3, le premier donnant les trajets directs (sans correspondance). Plus précisément, $tab_0[i,j]$ indique le temps entre la station i et la station j. On pourra mettre la valeur -1 (ou la valeur ∞) pour les couples (i,j) qui ne sont pas sur une meme ligne.

En vue de remplir tab_1, on pourra créer un tableau annexe, de taille 14×14 qui indique pour chaque couple de lignes, l'ensemble des stations de correspondance entre ces deux lignes. Pour trouver $tab_1[i,j]$, on commencera par chercher s'il existe α, β et k, tels que i soit sur la ligne α, j sur la ligne β, et k soit une correspondance de $L\alpha$ vers $L\beta$. Parmi tous ces triplets, on retiendra une station k telle que $tab_0[i,k] + tab_0[k,j] + \Delta_k(\alpha,\beta)$ soit minimal, où Δ_k dénote le temps de correspondance de $L\alpha$ vers $L\beta$, en k. On prendra alors pour $tab_1[i,j]$ la valeur minimale entre $tab_0[i,j]$ (si celle-ci est définie), et la valeur trouvée.

De meme pour remplir tab_2, on cherchera à minimiser $tab_1[i,k] + tab_0[k,j] + \Delta_k(\alpha,\beta)$, où α désigne cette fois *la ligne par laquelle on arrive en* k si on part de i et qu'on réalise ce minimum $tab_1[i,k]$. Comme pour Djiskra, il vous faut en parallele noter la provenance, au fur et à mesure : on aura des tableaux D_0, D_1, D_2 ... Par exemple ici : $D_1[i,k]$ vous fournit le numéro de ligne α.

Attention, en vue de trouver *le* trajet le plus court, il sera nécessaire d'avoir, en plus de tab_0, un tableau donnant pour chaque paire (i,j) tous les trajets directs (et donc aussi : stocker toutes les lignes qui comportent ces deux stations), et pas seulement celui de durée minimale. Par exemple si s_1 désigne la station *République* et s_2 la stations *Arts et métiers*, alors $tab_0[1,2]$ vaut $1,5$ et $D_0[1,2] = 11$ (car stations consécutives sur la ligne 11), mais vous aurez besoin d'un autre tableau que tab_0, appelons-le Dir, qui stocke les différentes possibilités (ici : $Dir[1,2] = (1,5;3)$ et $D_0[1,2] = (11,3)$), pour pouvoir remplir correctement vos tableaux.