Project : MAN

Make a network

IUT INFO 3ème année Valence. Année 2023-2024

Jean-Baptiste Caignaert ©

Image issue du jeu Mini-Métro

Ce projet a pour objectif de vous faire développer en langage Pyhton un programme qui va optimiser des déplacements de bus pour transporter des personnes.

L'univers est décrit par des routes reliant des arrêts, où circulent des bus transportant des personnes. L'univers commence au temps 0 et se termine au temps 100000 (unité de temps S) ou quand tous les trajets de toutes les personnes sont finis (le premier des deux arrivant en premier).

Les programmes sont considérés comme déterministes, les personnes prendront individuellement les bus optimisant leur propre temps de transport, on considérera qu'ils ont une connaissance globale de l'ensemble de l'univers.

A chaque arrêt, les bus se remplissent jusqu'à leur capacité maximum, selon le nombre de personnes.

Si deux personnes arrivent simultanément à un arrêt pour prendre le même bus, c'est l'ordre en terme alphanumérique de leur nom qui décide de leur montée si besoin.

Si une personne n'a pas fini son aller alors que son retour devrait déjà commencer : elle termine son aller et commencera immédiatement son retour en sortant du bus et en changeant de bus (elle ne doit pas reprendre le même bus).

Les bus sont décrits par :

- Une charge maximale = capacité maximum de personnes transportable à un instant t
- Une rapidité de chargement/déchargement en unité de temps (S) par personne (les chargements/déchargement se faisant en parallèle)
- Une vitesse de déplacement (nombre d'unité de mesure (M) en unité de temps (S))
- Un parcours : des arrêts à faire en boucle « indéfiniment »

Les arrêts sont décrits par :

- Des routes (minimum 1) reliant(s) cet arrêt
- Une file d'attente ordonnée des personnes (premier arrivé, premier choisi)

Les routes sont décrites par :

- 2 arrêts
- Une distance en unité de mesure (M)

Les personnes sont décrites par :

- Un trajet aller : heure de départ avec l'arrêt de départ, arrêt d'arrivée
- Un trajet retour : heure de départ avec l'arrêt de départ, arrêt d'arrivée
- Un nom

Premier programme à faire :

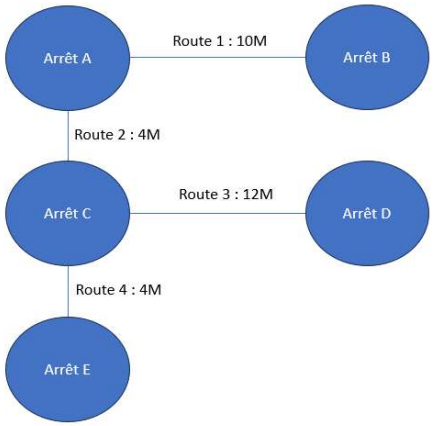
Univers à coder :

3 bus de type double :

- Capacité maximale : 10 personnes
- Chargement/déchargement : 1 personne par unité de temps (S)
- Vitesse : 1M toute les 30S
- 1 bus fait le parcours BACECA en boucle
- 1 bus fait le parcours DCEC en boucle
- 1 bus fait le parcours BED en boucle

1 Bus de type fast :

- Capacité maximale : 2 personnes
    - Chargement/déchargement : 1 personne par unité de temps (S)
- Vitesse : 1M toute les 10S
- Trajet Arrêt A à Arrêt C en boucle



Ecrivez un programme qui décrit cet univers et prendra en entrée une liste de personnes.

Par exemple personnes :

- 12 AlbertX (X valant leur numéro) qui font B->D au temps 0, puis D->A au temps 500
- 12 BobX (X valant leur numéro) qui font B->C au temps 0, puis C->B au temps 500
- 12 CharlesX (X valant leur numéro) qui font B->C au temps 0, puis C->B au temps 500
- 12 DamienX (X valant leur numéro) qui font E->A au temps 0, puis A->E au temps 600
- 45 EdouardX (X valant leur numéro) qui font A->C au temps 0, puis C->A au temps 300

Seront représentés par le fichier texte suivant :

```
12 Albert 0 BD 500 DA
12 Bob 0 BC 500 CB
12 Charles 0 BC 500 CB
```

```
12 Damien 0 EA 600 AE
45 Edouard 0 AC 300 CA
```

Votre programme prendra donc en argument un fichier décrivant les personnes de l'univers. En cas de fichier invalide/incohérent, une erreur devra être retournée, expliquant au mieux le type d'erreur retourné.

Nous vous invitons à faire de multiples tests pour vérifier que tout est cohérent.

Votre programme devra dérouler à chaque unité de temps la situation de l'univers. Il est indispensable d'avoir une représentation graphique claire pour voir à chaque unité de temps la situation globale (notamment les Arrêt Bus, Personnes, temps). Il est demandé d'avoir un traçage de l'historique et de pouvoir donc « revenir en arrière » de manière facile et fluide.

Sur vos différents algorithmes expliqués, il faudra préciser la complexité calculée.

Deuxième programme à faire (une fois le premier fini !) :

L'objectif de ce deuxième programme est de prendre en paramètre, l'ensemble de l'univers et de dérouler un premier algorithme qui dispatchera les transports des différents bus afin de minimer les temps d'attentes. En temps d'attente moyen et en temps d'attente au carré.

Ce premier résultat devra être affiché clairement à l'utilisateur sous la forme de la liste des arrêts à faire :

```
Bus1 DCEA
Bus2 AB
Bus3 EDADB
```

Il sera demandé de décrire aussi la complexité de ce premier algorithme.

Ensuite, un deuxième algorithme déroulera l'évolution de l'univers avec l'ensemble des paramètres.

Les formats des paramètres vous seront fournis après validation de votre premier programme.

Monde décrit :

Ne commencez cela QUE si vous avez finit la première partie !

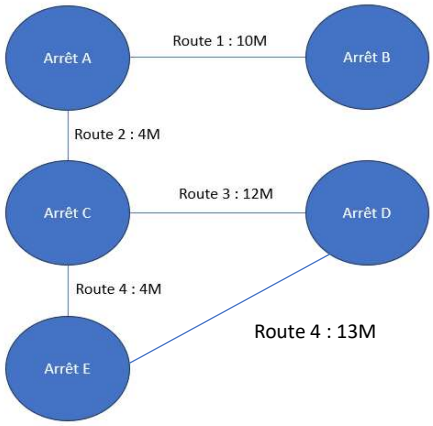
Votre programme prendra désormais 3 fichiers en paramètres :

Le premier correspond aux personnes

Le deuxième aux Arrêts/Routes

Le troisième aux Bus

Exemple :



Fichier :

```
Arrêt A B 10M
Arrêt A C 4M
Arrêt C D 12M
Arrêt E C 4M
Arrêt E D 13M
```

Exemple

Si on a 17 bus de type « double »

- Capacité maximale : 10 personnes
- Chargement/déchargement : 1 personne par unité de temps (S)
- Vitesse : 1M toute les 30S

Et on a 6 bus de type « Moto »

- Capacité maximale : 3 personnes
- Chargement/déchargement : 2 personnes par unité de temps (S)
- Vitesse : 10M toute les 20S

17 Bus double

Capacité 10

Charge 1

Vitesse 1M 30

6 Bus Moto

Capacité 3

Charge 2

Vitesse 10M 20