1 Compétition d'algorithmie présentée par

Figure 1: CFIUL

Figure 2: AEGLO

2 Introduction

Après s'être fait dépassé par Elon Musk au classement mondial, notre bon ami Bezos a décidé de step up son game et décide d'investir encore plus de ressources dans la livraison par drones, évidemment c'était avant de redevenir premier à cause qu'un investissement dans les bitcoins est aussi volatile qu'un gallon d'éther. Pour mener à terme ce projet, c'est bien beau bâtir des drones et offrir des pots de vin aux agents pour avoir les accès au ciel, il faut aussi faire un système de gestion.

3 Tâche

On vous donne une liste de commandes et une liste des entrepôts ainsi que leur produits disponibles et vous devez organiser les drones de façon a remplir le plus de commandes possible dans le temps imparti.

4 Description du problème

4.1 Carte

La simulation prendra place sur un plan cartésien **fermé** et **non-cylique**, donc les drones ne pourront par sortir des bordures de la carte et ne pourront pas non plus traverser d'une bordure à l'autre instantanément. Par contre, comme les drones peuvent voler, il sont capable de survoler n'importe quelle cellule de la carte (supposont qu'il n'y a pas d'aéroport dans la ville).

Chaque cellule sera identifiée par une paire d'entiers [*r*, *c*] (rangée, colonne; chacun limité par la taille de la carte définie au début du fichier)

4.2 Produits

Il y a un catalogue limité de produits disponibles pour la livraison et chaque type possède un poids fixe, déterminé au préalable par le challenge.

- ***N.B: Chaque produit est disponible en, au moins, un exemplaire dans la carte et le poids d'un produit sera toujours inférieur à la capacité de charge d'un drone.***

4.3 Entrepôts

Les entrepôts ainsi que leur emplacements sont déterminés au début du fichier de challenge. Il est garanti que deux entrepôts ne seront jamais sur la même cellule. Au début de la simulation, les entrepôts seront stockés avec une quantité limitée de chaque type de produit, cependant il est possible que certains types de produits ne se retrouvent pas dans un entrepôt. Malgré que les entrepôts ne sont pas remplis au fil de la simulation, il est

toujours possible de transférer des produits entre eux. Aucune limite de capacité n'est imposée.

4.4 Commandes

Chaque commande possède une liste de produits, ces derniers se trouvant en un ou plusieurs exemplaires. De plus, une commande possède une cellule à laquelle faire la livraison et elle est garantie de ne jamais tomber sur un entrepôt.

Pour compléter une livraison, il ne suffit que de livrer les produits requis à l'endroit spécifié peu importe l'ordre. Il est tout à fait possible de séparer la commande en plusieurs drones et il n'est pas obligatoire de tout livrer en même temps.

Il est garanti qu'il y aura toujours assez de produits disponibles dans l'ensemble des entrepôts pour remplir chacune des commandes.

Pour finir, il n'est pas nécessaire de remplir chacune des commandes, le but est d'en faire le plus possible. *(Voir le pointage plus bas)*

4.5 Drones

Les drones sont utilisés pour transporter les produits. Ils utilisent toujours le chemin le plus court pour arriver à leur destination et elle se calcule selon une distance euclidienne, donc la distance entre le point *a* et le point *b* se calcule avec cette formule:

$\sqrt{\mid a_x - b_x \mid^2 + \mid a_y - b_y \mid^2}$

De plus, les drones se déplacent à une vitesse de 1 unité de distance par tour et le temps total d'un vol est arrondi à la hausse. Par exemple, si la formule donne une distance de *5.192*, le temps total nécessaire pour faire le trajet sera de 6 tours.

Plusieurs drones peuvent voler à des altitudes différentes, donc ils peuvent se croiser sur la même cellule sans collision.

Au début de la simulation, tous les drones commencent au premier entrepôt (celui avec l'id **0**).

4.6 Ordres

Les drones peuvent être commandés avec les ordres de base suivant:

- **Load**: Charge une quantité d'un produits dans l'inventaire du drone. Si le drone ne se trouve pas à l'entrepôt spécifié, il s'y rendra préalablement selon la technique énoncée plus haut. L'ordre échouera si l'entrepôt ne contient pas le nombre demandé du produit voulu et si la charge du drone dépasse sa capacité.
- **Deliver**: Livre une quantité d'un produit à un client. Si le drone ne se trouve pas au client spécifié, il s'y rendra préalablement selon la technique énoncée plus haut. L'ordre échouera si le drone ne possède pas le nombre demandé du produit voulu dans son inventaire.

Les ordres suivants sont facultatifs pour résoudre le challenge, cependant ils peuvent être utiles pour optimiser les livraisons.

- **Unload**: Dépose une quantité d'un produit dans un entrepôt. Si le drone ne se trouve pas à l'entrepôt spécifié, il s'y rendra préalablement selon la technique énoncée plus haut. L'ordre échouera si le drone ne possède pas le nombre demandé du produit voulu dans son inventaire.
- **Wait**: Attends sans rien faire pendant un nombre de tours spécifié.

4.7 Simulation

La simulation se déroule en un nombre de tours spécifié dans le fichier de challenge. Les drones exécutent ses ordres dans l'ordre qu'ils sont assignés, un après l'autre. La première commande issue à un drone commence au tour 0.

La duré d'un ordre dépends de son type:

- **Load/Deliver/Unload**: Prends un tour de plus que la duré du trajet. Pour reprendre l'exemple plus haut, un trajet de *5.192* donnera un durée de d'ordre de 7 tours (6 pour le déplacement et 1 pour l'action). Si le drone se trouve déjà à l'endroit, l'ordre ne prends qu'un tour.
- **Wait**: Le nombre spécifié de tours

Si plusieurs drones font une action au même entrepôt pendant le même tour, les dépôts *(Unload)* sont traités avant les chargements *(Load)*.

5 Fichier d'entrée

Les données d'entrée sont fournies à l'aide d'un fichier texte contenant que des caractères ASCII. Les lignes sont terminée par un *\n* (Style UNIX).

Les types de produits, les entrepôts ainsi que les commandes sont numérotés à l'aides d'IDs entiers, on commence à compter à partir de 0 parce qu'on n'est pas des monstres.

5.1 Format du fichier

La première section du fichier spécifie les paramètres de la simulation. Cette section contiens une seule ligne avec les nombres entiers suivants:

- nombre de lignes (entre 1 et 10 000 inclus)
- nombre de colonnes (entre 1 et 10 000 inclus)
- nombre de drones (entre 1 et 1000 inclus)
- durée maximale de la simulation (entre 1 et 1 000 000 inclus)
- charge maximale d'un drone (entre 1 et 10 000 inclus)

La prochaine section spécifie les poids des différents produits. Elle est formée de la façon suivante:

- Une ligne contenant le nombre de produits différents (entre 1 et 10 000 inclus)
- Une ligne contenant les poids de chaque type de produit, séparés par un espace entre chaque (entre 1 et la charge maximale d'un drone inclus)

La section suivante spécifie les entrepôts ainsi que leur stocks. Elle est formée de la façon suivante:

- Une ligne contenant le nombre d'entrepôts (entre 1 et 10 000 inclus)
- Deux lignes par entrepôt avec les informations suivantes

    1. Deux nombres entiers représentant la position de l'entrepôt sur le plan cartésien
    2. La quantité de chaque type de produit disponible dans l'entrepôt. (entre 0 et 10 000 inclus)

La dernière section spécifie les différentes commandes. Elle est formée de la façon suivante:

- Une ligne contenant le nombre de commandes (entre 1 et 10 000 inclus)
- Trois lignes par commandes avec les informations suivantes

    1. Deux nombres entiers représentant la position du client sur le plan cartésien
    2. Le nombre de produits requis dans la commande (entre 1 et 10 000 inclus)
    3. Le type de chaque produit commandé séparé par des espaces, aucun ordre n'est garanti et les duplicats sont écrits individuellement. Par exemple, si le client veux 3 fois le produit 5 et une fois le produit 10, la liste pourrait ressembler à `5 5 10 5`.

Voici un exemple du contenu d'un fichier d'entrée ainsi que son interprétation:

Fichier	Interprétation
100 100 3 50 500	100 rangées, 100 colonnes, 3 drones, 50 tours, poids max de 500 par drone
3	Il y a 3 types de produits
100 5 450	Les poids de chacun des types: 100; 5; 450
2	Il y a deux entrepôts
0 0	Le premier entrepôt est situé en (0, 0)
5 1 0	Il contient 5 items de type 0 et 1 item de type 1
5 5	Le second entrepôt est situé en (5, 5)
0 10 2	Il contient 10 items de type 1 et 2 items de type 2
3	Il y a 3 commandes
1 1	La première commande doit être livrée en (1, 1)
2	Elle requiert 2 items au total
2 0	Elle requiert un item de type 2 et un item de type 0

Fichier	Interprétation
3 3	La deuxième commande doit être livrée en (3, 3)
1	Elle requiert 1 item au total
0	Elle requiert un item de type 0
5 6	La troisième commande doit être livrée en (5, 6)
1	Elle requiert 1 item au total
2	Elle requiert un item de type 2

6 Soumission

6.1 Format de fichier

La première ligne du fichier de sortie contient un seul entier, le nombre d'ordres a exécuter.

Le reste du fichier devrait simplement être chaque ordre, séparé chacun sur une ligne. Les ordres des différents drones peuvent être interchangés, cependant, pour chaque drone, ils doivent être placés en ordre chronologie.

Les drones sont identifiés par des entiers consécutifs en commençant par 0, donc de 0 jusqu'au nombre de drones non-inclu.

Par exemple, si le drone 0 possède les ordres suivants, (où la durée de l'ordre est entre parenthèses): `ordre0(1), ordre1(5), ordre2(3)` et que le drone 1 possède ces ordres là: `ordre3(4), ordre4(1), ordre5(1)` voici le timeline de l'ordonnancement des ordres:

Temps:	0	1	2	3	4	5	6	7	8
Ordre:	0;3	1	-	-	4	5	2	-	-

Comme on peut voir, même si l'ordre *3* arrive après l'ordre *0*, ils sont exécutés en même temps puisqu'il s'agit de différents drones, de là l'intérêt d'avoir une commande *wait*, sans elle, il serait impossible de faire attendre un drone après un autre.

Avec les différents symboles suivants:

- *R*: L'ID du drône
- *E*: L'ID de l'entrepôt
- *T*: L'ID du type de produit
- *C*: L'ID de la commande
- *Q*: La quantité (un entier positif)

On peut décrire les ordres via cette forme (les caractères en gras sont littéraux):

- **Load**: *R* - **L** - *E* - *T* - *Q*
- **Unload**: *R* - **U** - *E* - *T* - *Q*
- **Deliver**: *R* - **D** - *C* - *T* - *Q*
- **Wait**: *R* - **W** - *Q*

Voir la section Ordres pour savoir que font les ordres.

Voici un exemple du contenu d'un fichier de sortie ainsi que son interprétation:

Fichier	Interprétation
9	Il y a 9 commandes au total
0 L 0 0 1	Drone 0: **Load** 1 produit de type 0 à partir de l'entrepôt 0
0 L 0 1 1	Drone 0: **Load** 1 produit de type 1 à partir de l'entrepôt 0
0 D 0 0 1	Drone 0: Vole jusqu'à la commande 0 et **Deliver** 1 produit de type 0
0 L 1 2 1	Drone 0: Vole jusqu'à l'entrepôt 1 et **Load** 1 produit de type 2
0 D 0 2 1	Drone 0: Vole jusqu'à la commande 0 et **Deliver** 1 produit de type 2
1 L 1 2 1	Drone 1: Vole jusqu'à l'entrepôt 1 et **Load** 1 produit de type 2
1 D 2 2 1	Drone 1: Vole jusqu'à la commande 2 et **Deliver** 1 produit de type 2
1 L 0 0 1	Drone 1: Vole jusqu'à l'entrepôt 0 et **Load** 1 produit de type 0
1 D 1 0 1	Drone 1: Vole jusqu'à la commande 1 et **Deliver** 1 produit de type 0

6.2 Pointage

Chaque commande complétée se verra attribuer un pointage entre 1 et 100, dépendamment du nombre de tours nécessaire à sa complétion. Une commande est considérée complétée dès que chaque produit a été livré.

Pour une simulation ayant une durée ***T*** et une commande complétée au tour ***t***, le pointage attribué suivra cette formule et sera arront à l'entier supérieur si besoin est:

$\frac{T-t}{T} * 100$

Par exemple, pour une simulation durant 160 tours, si une commande a été complétée au tour 15, alors le pointage calculé sera (160 - 15) / 160 = 0.90625, puis multiplié par 100 et arrondi à la hausse pour un total de 91 points.

Le pointage attribué à un défi sera la somme de toutes les commandes complétés.

À noter qu'il y aura plusieurs défis différents, représentant plusieurs facettes du problème. Le pointage final sera la somme des meilleurs pointages de chaque défi.

6.3 Inscription et Soumission

Méthode simple : - *Inscription*: Exécuter le script `register.py` et répondre aux questions adéquatement. Un token unique vous sera renvoyé par le serveur. Il est important de ne pas perdre le token puisque celui sera utilisé pour identifier votre équipe lors de la soumission des réponses. - *Soumission*: Exécuter le script `send_solution.py`. Il faut donner votre token unique d'équipe, le challenge d'équipe et le fichier contenant la solution sous forme de string.

Méthode compliquée :

Si vous voulez implémenter votre propre système, il est possible de communiquer directement avec le serveur à l'aide d'une requête TCP. Pour ce faire, il faut envoyer des données en format JSON en suivant cette structure:

```
Inscription
{
    method: "register",
  team_name: "nom de l'équipe",
  participants: ["part1", "part2", "part3", "part4"]
```

```
        }

Soumission
{
            method: "challenge",
            team_id: "identifiant d'équipe",
          chall_id: "identifiant de challenge",
          solution: "solution en string ici"
        }
```

6.3.1 Identifiants de challenge

- 0 : *busy_day.in*
- 1 : *mother_of_all_warehouses.in*
- 2 : *redundancy.in*