

# Gestion d'un entrepôt

## Objectif

Optimiser le parcours des **transpalettes** dans un **entrepôt** afin de maximiser le remplissage de **camions** de transports partant d'un **entrepôt**.

## Présentation

En entrée du programme un fichier texte sera fourni décrivant les différentes contraintes concernant la modélisation des différentes données du problème à optimiser.

L'**entrepôt** peut être vu comme une grille contenant les informations suivantes :

- position des **colis** à livrer dans le camion ainsi que leur poids représenté par une **couleur**
- positions des **transpalettes** disponibles
- positions des **aires de dépôt** (position où un **transpalette** pourra remplir un **camion**)

Voici un exemple visuel de représentation du problème :

transpalette 1				
		colis 1		colis 4
		colis 2		transpalette 2
colis 3				
			aire de dépôt	



Le **transpalette 1** est en position [0, 0]

L'**aire de dépôt** est en position [3, 4]

D'autres informations seront disponibles comme :

- le poids maximale transportable par un camion (charge maximale)
- le nombre de tour avant qu'un camion soit disponible une fois celui-ci chargé
- le nombre de tours total à simuler

## Description du fichier en entrée

La première ligne contiendra 3 nombre séparés par un espace :

- nombre de cases en largeur de l'entrepôt
- nombre de cases en longueur de l'entrepôt
- nombre de tours à simuler.

Les n lignes suivantes décriront les **colis** présents dans l'entrepôt.

Chaque ligne contiendra 3 éléments :

- le nom du **colis** (une chaîne de caractères sans espace)
- la position du **colis** (2 nombres séparés par des espaces)
- la couleur du **colis** (une chaîne de caractères valant **GREEN**, **YELLOW** ou **BLUE**, pouvant être en majuscule ou en minuscule)

Les k lignes suivantes décriront les **transpalettes** présents dans l'entrepôt.

Chaque ligne contiendra 2 éléments :

- le nom du **transpalette** (chaîne de caractères sans espace)
- la position du **transpalette** (2 nombres séparés par des espaces)

Enfin les p dernières lignes du fichier seront composées des éléments suivants :

- le nom du **camion** (chaîne de caractères sans espace)
- la position de l'**aire de dépôt** (2 nombres séparés par des espaces)
- la charge maximale du **camion** (1 nombre)
- le nombre de tours avant que le **camion** ne soit de nouveau disponible

Voici un exemple de fichier avec 4 **colis**, 1 **transpalette**, 1 **camion** et un **entrepôt** de forme carré :

```
5 5 1000
colis_a_livrer 2 1 green
paquet 2 2 BLUE
deadpool 0 3 yellow
colère_DU_dragon 4 1 green
transpalette_1 0 0
camion_b 3 4 4000 5
```

### Définition des contraintes métier

Une position est une paire de nombres entier et positif (0-n). Le premier nombre représente les abscisses et le second les ordonnées. La position [0,0] est située en haut à gauche de l'entrepôt.

A chaque couleur d'un **colis** est associé un poids particulier.

Le **YELLOW** vaut 100 kg, le **GREEN** 200 kg et le **BLUE** 500 kg.

Un **transpalette** ne peut porter qu'un **colis** à la fois et n'a pas de contraintes de poids maximum. Il peut donc contenir un **colis** ou aucun. Il ne peut se déplacer que d'une case à chaque tour et ne peut pas aller en diagonale.

Un **transpalette** ne pourra occuper une case où se trouve déjà un **colis** ou un autre **transpalette**.

Une fois le **transpalette** sur l'**aire de dépôt**, celui-ci pourra remplir le **camion**. Il réalisera cette action en un tour.

Pour prendre un **colis**, un **transpalette** devra se trouver à côté de celui-ci (une des 4 directions adjacentes libres, hors diagonale). Il faudra un tour pour prendre le **colis** de son emplacement.

La charge maximale du **camion** sera exprimée en kilo (par exemple 1000 kg).

Un **camion** pourra soit attendre ("**WAITING**"), soit partir ("**GONE**"). Le **camion** restera dans l'état "**GONE**" jusqu'à son retour, soit le nombre de tours avant que le **camion** ne soit de nouveau disponible. Il peut donc y avoir plusieurs camions en **WAITING** ou en **GONE** devant l'**entrepôt**.

Le nombre de tours est un entier positif compris entre 10 et 100 000.

### Définition des contraintes applicatives

Le programme se terminera quand le nombre total de tours sera atteint ou quand il n'y aura plus de **colis** à livrer. On considère que le dernier **colis** est livré quand celui-ci est déposé dans le **camion**.

A chaque tour le programme devra afficher :

- le numéro du tour en cours préfixé du mot "tour"
- pour chaque **transpalette** : son nom, l'action choisie et les informations relatives à cette action
- pour chaque **camion** : son nom, l'action choisie et les informations relatives à cette action

Les actions possibles pour un **transpalette** sont :

- **GO**: cette action signifie que celui-ci se déplace, il sera suivi de sa nouvelle coordonnée.
- **WAIT**: le **transpalette** ne bouge pas.
- **TAKE**: le **transpalette** prend un **colis**. Le **colis** pris sera affiché (nom + couleur).
- **LEAVE**: le **transpalette** dépose le **colis** dans le **camion**. Le **colis** déposé sera affiché (nom + couleur).

Les états possible pour un camion :

- **WAITING**: le camion attend d'être chargé. On affichera sur la même ligne la somme des colis déjà chargés ainsi que son poids maximal séparé par un "/". Par exemple 250/1000.

- **GONE**: le camion est parti. On affichera sur la même ligne la somme des colis déjà chargés ainsi que son poids maximal séparé par un “/”.

Un nouveau saut de ligne sera effectué entre chaque tour.

A la fin du programme un emoji sera affiché représentant la condition d’arrêt:

- 🧐 : l'**entrepôt** est vide.
- 😊 : le nombre de tours max est atteint et il reste des **colis** à livrer.
- 🤯 : suivi d’un message d’erreur dans le cas d’un problème de données non valides ou de crash du programme.

Exemple de sorti console :

```
tour 1
transpalette_1 GO [1,0]
transpalette_2 GO [2,2]
camion_b WAITING 0/1000

tour 2
transpalette_1 TAKE colis_1 BLUE
transpalette_2 WAIT
camion_b WAITING 0/1000

...

tour 29
transpalette_1 WAIT
transpalette_2 LEAVE colis_2 YELLOW
camion_b WAITING 800/1000

tour 30
transpalette_1 WAIT
transpalette_2 WAIT
camion_B GONE 900/1000
```



**Dernière remarque, vous n’aurez pas le droit de dépendre d’une bibliothèque permettant de résoudre un problème algorithmique (Dijkstra, A\*, ...).**

### Livrable attendu

- Les sources du projet devront être disponibles sur Github ou Gitlab.
- Le repository devra contenir un fichier README.md ayant une section expliquant comment **compiler** et **lancer le projet**. Une autre section décrira l’organisation des sources et des packages. Enfin une dernière section décrira la stratégie retenue.

- Le repository devra contenir un “component diagram” correspondant au level 3 du C4 model (<https://c4model.com/>)

Un fichier “.golangci.yml” sera fourni et devra être passé sur votre projet. Le linter ne devra remonter aucune erreur.

### Bonus

Ce projet va vous permettre de jouer avec le langage GO, vous pouvez donc agrémenter ce projet d’une interface graphique (native à l’OS), de scripts de déploiement sur le cloud (GCP, AWS, AZURE, ...), de génération d’un serveur web avec interface graphique (HTML + Javascript), ...

Happy Coding 🎉