PS5.3 - Taxi Fleet

Problem description

Millions of people commute by car every day; for example, to school or to their workplace.

Self-driving vehicles are an exciting development for transportation. They aim to make traveling by car safer and more available while also saving commuters time.

In this problem, we'll be looking at how a fleet of self-driving vehicles can efficiently get commuters to their destinations in a simulated city.

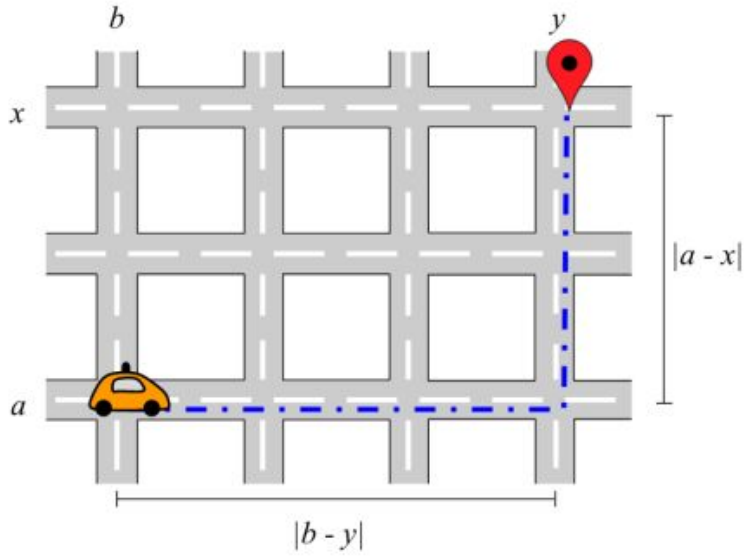
Your job

Given a list of pre-booked rides in a city and a fleet of self-driving vehicles, assign the rides to vehicles, so that riders get to their destinations on time.

Optimize your affectation to get the best score!

Problem simulation:

The parameters



Your are in a <rows> x <columns> square city

You have <nb_vehicle> vehicles

- Starting in 0/0

You have <nb_ride> rides

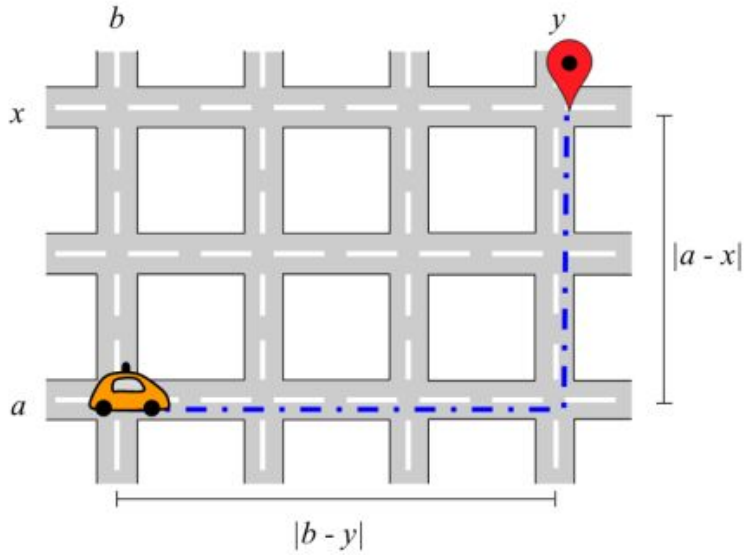
- Starting in [<fromX>,<fromY>]
- Starting in [<toX>,<toY>]
- Starting after <earliest_start>
- Finishing after <latest_finish>

You are given a <on_time_bonus>, a <fixed_fee>, and a <distance_fee>

The simulation is <nb_steps> long

Problem simulation:

Vehicle itinerary



For each assigned ride:

1. The vehicle goes from its current position to the start intersection of the ride
2. If it is on position before <earliest_start>, it waits until <earliest_start>
3. The vehicle does the ride

Moving duration:
It takes 1 step for each bloc driven.

Example:
If a vehicle starts in [1,2] and finish in [4,3], the move will last 4 steps (|1 - 4| + |2 - 3| = 4 steps).

Problem simulation:

The input file

First line:

<rows> <columns> <nb_vehicles>
<nb_rides> <on_time_bonus> <fixed_fee>
<distance_fee> <nb_steps>

<nb_rides> next lines:

<fromX> <fromY> <toX> <fromY>
<earliest_start> <latest_finish>

Your are in a <rows> x <columns> square city

You have <nb_vehicle> vehicles

Starting in 0/0

You have <nb_ride> rides

Starting in [<fromX>,<fromY>]

Starting in [<toX>,<toY>]

Starting after <earliest_start>

Finishing after <latest_finish>

You are given a <on_time_bonus>, a <fixed_fee>, and a <distance_fee>

The simulation is <nb_steps> long

Problem simulation:

The output file

First line:

<calculated_score>

<nb_vehicles> next lines:

<nb_assigned_rides> <1st_ride_id> <2nd_ride_id> ...

You have to calculate your <calculated_score>

The simulation is <nb_steps> long

For each vehicle

Provide the <nb_assigned_rides>

And the assigned rides <nth_ride_id>

Problem simulation:

Score calculation

The **total score** is the **sum** of the score of **each** rides **done**.

The score for a ride is:

$ *<fixed_fee> + <ride_distance> x <distance_fee>*

If the ride start on time, an additional <on_time_bonus> is added to the ride score.

Example:
For a 10 blocks long ride started on time, the score is:
<on_time_bonus> + <fixed_fee> + 10 x <distance_fee>

Benchmarking:

Measure your solution's efficiency

Run your code multiple times to estimate its running time **empirically**.

Objective: obtain an **average** measure of your code's execution time

Tools used:

- JMH: benchmarking
- a visualization of the results of your choice


ANY
QUESTIONS ?


Project execution
WEEK 4 : Design
WEEK 5 : Design
WEEK 6 : Dev
WEEK 7 : Dev
Finalize API Goals
Initialize Database
Create fixtures for testing
Admin Dashboard BIE Framework
Database API (continues)
Engineering
B/E Sorting (For All Dashboard Views) w/ pagination
B/E Filter (ton for all views)
B/E Crud for all DB Tables
Exporting
CRON
ADMIN API

Timeline

Every day:

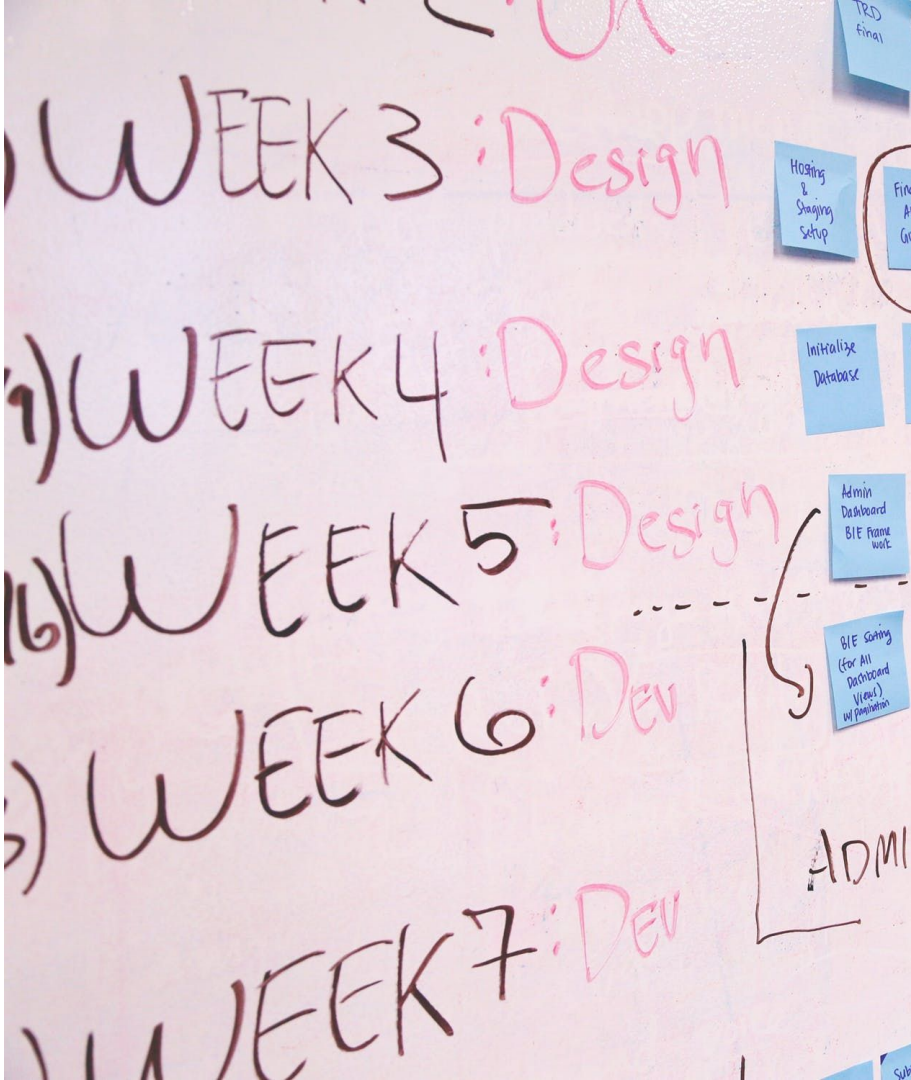
- facilitation
- automated delivery

Tuesday morning: Q&A session

Thursday: demonstrations

Saturday: code delivery

Sunday: report delivery

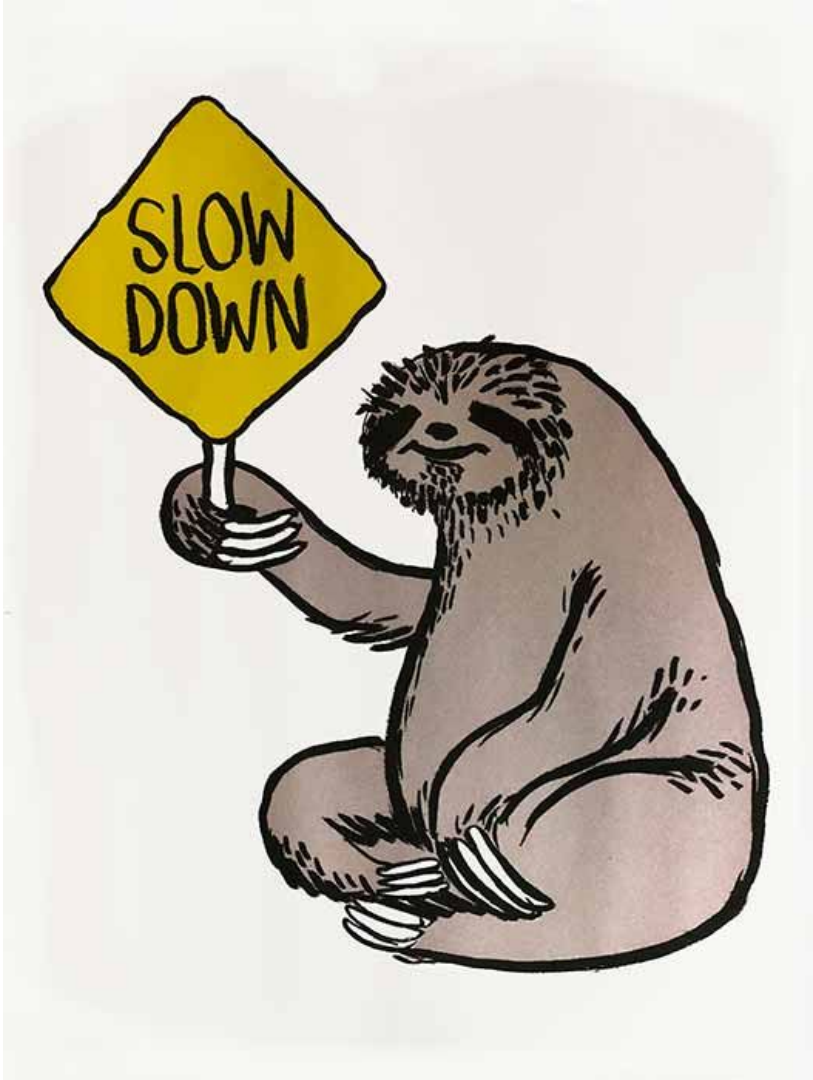


Before diving into the project

- 10:00 AM: All teams declared properly in GitHub Classroom

- Read the subject carefully!
(link will be on Slack)

- Use Slack and `#si3-ps5` channel to ask questions, they may be useful to others!



Daily delivery

Monday - 8PM (tag REL1)

Tuesday- 4PM (tag REL2)

Wednesday - 4PM (tag REL3)

Friday - 12AM (tag REL4)

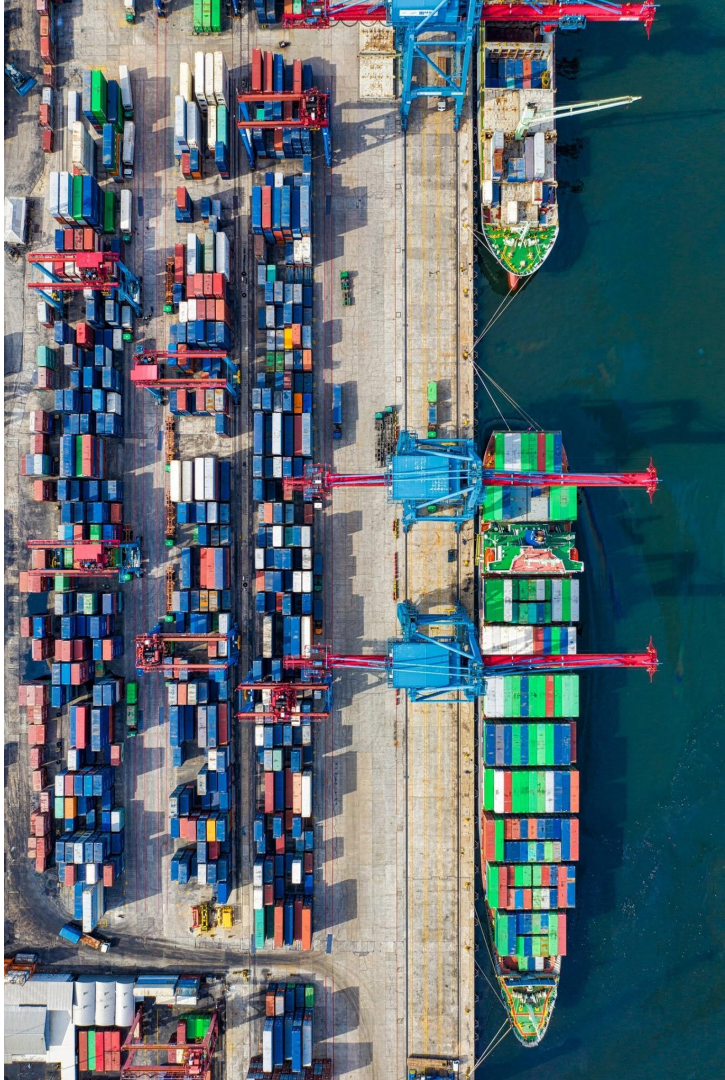
Saturday - 6PM (tag FINAL)

git clone

→ git checkout <tagname>

→ mvn clean package

→ mvn exec:java@run




3... 2... 1... Start!