

## Hitachi Hokudai Lab. & Hokkaido University Contest 2022

### An example of the summary

**Note:** The “transport\_only\_0” in sample\_B.cpp of the toolkit of problem B in the contest “Hitachi Hokudai Lab. & Hokkaido University Contest 2020” is used as an example.

The problem is regarding an optimization of transportation and charging/discharging schedules for multiple electric vehicles (EVs). Please see below for the website of the contest:

<https://atcoder.jp/contests/hokudai-hitachi2020>

- Header

Screenname: XX

Submission date: XX.XX.2023

- Body

(1) (a)

#### Overview

The “transport\_only\_0” is a simple algorithm that solves problem B by transporting only. In this algorithm, the following processes (i)-(iii) are repeated. (i) Each EV waits on the nanogrid until an order is placed. (ii) When an order is placed, the EV transports customers and goods in order of their order IDs. (iii) After the transportation, the EV moves to the nearest nanogrid and waits for a new order.

#### Strategy

\* Each EV waits on the nanogrid until an order is placed.

\* While the EV waits on the nanogrid, it is charged until its charge reaches a predetermined amount. (The predetermined amount is "safety\_energy" on line 438. It is charge for the EV to move for 50 steps.)

\* After that, the EV transports customers and goods in order of their order IDs.

\* The EV carries out a maximum of four transportations at once. Although it is desirable for the EV to carry out more transportation at once, the maximum number of simultaneous transports is set to four so that the constraints regarding the number of orders loaded on each EV ( $N_{\max}^{\text{trans}}$ ) can be easily satisfied.

(1)(b)

#### Implementation

\* The actions of each EV are managed by "command\_queue" in the source code.

\* In the pseudocode below, the n-ths EV and the corresponding "command\_queue" are represented as `EV[n]` and `queue[n]`, respectively.

\* By using the `move to` function, the move commands to each EV can be simply enqueued in the "command\_queue".

\* This algorithm is implemented in "transport\_only\_0" (lines 426-489) and "find\_transit\_path\_greedy" (lines 264-297) in the source code (submission number: XXXX).

### Pseudocode

```
``` text
```

```
Read the data written in "Input and Output Format 1."
```

```
for t = 1 to T_max do
```

```
    Read the data written in "Input and Output Format 2."
```

```
    for n = 1 to N_EV do
```

```
        // Processing to enqueue commands to queue[n]
```

```
        if queue[n] is empty
```

```
            // Moving EV[n] to the nanogrid and charging EV[n] are priorities to keep EV[n] moving.
```

```
            if EV[n] is not on the nanogrid
```

```
                if EV[n] does not have enough power left to reach the nanogrid
```

```
                    Fill queue[n] with 'stay.' // EV[n] will not move anymore.
```

```
                    continue
```

```
                else
```

```
                    Enqueue 'move' commands for EV[n] to reach the nearest nanogrid in queue[n].
```

```
                    continue
```

```
                end if
```

```
            else
```

```
                if EV[n] has less electricity than 'safety_energy'
```

```
                    Enqueue 'charge_from_grid' commands to queue[n] until EV[n]'s charge reaches 'safety_energy.'
```

```
                    continue
```

```
                end if
```

```
            end if
```

```
        // Processing for transportation
```

```
        if there are one or more orders to which the EV is not assigned
```

```
            Assign up to 4 orders to EV[n] in order of order ID.
```

```

    Greedy algorithm determines loading and unloading schedule for assigned orders.
    if EV[n] does not have the power needed to transport customers and goods on the
    schedule
        Enqueue 'charge_from_grid' commands to queue[n] to charge EV[n] until it
        reaches the power needed to transport customers and goods according to the schedule
        end if
        Enqueue 'move' commands to queue[n] so that EV[n] can transport customers and
        goods according to the schedule
        end if
    end if
    // Output the commands below
    if queue[n] is empty
        Output 'stay'
    else
        Output a command in queue[n]
    end if
end for
end for
Read the data written in "Input and Output Format 2."
Read the data written in "Input and Output Format 3."
...

```

- (2) (a) I did not have time, so I could only implement the transport algorithm. However, by using its relatively simple transport algorithm, cars were able to transport customers and goods to some extent. If I could implement the power management algorithm, I think I could get a higher score.
- (2) (b) It may be possible to introduce a shared taxi service that allows you to get on and off nearby with less waiting time than a bus. I felt that the application scene would expand, such as a shared bus, if the upper limit of the number of passengers could increase to more than four.
- (3) The visualizer helped me verify or debug my coding. It would be better if it displays indicator XX and has function YY.
- (4) The problem statement was long and complicated, but I had the impression that it was a problem with a view to practical use, which motivated me to tackle it. I think problem A is well balanced given the duration of the contest and its difficulty. Due to lack of time, I could not try problem B.