Level 5 – Standard extended

It is really inefficient to execute each request alone. And furthermore there are constellations where a single request will have to wait the whole time to be processed.

So the task for that level will be to allow the executing of multiple requests at the same time, where the ordering of the requests is still done by using the prioritization of the previous level.

The elevator adds a request to the current execution if it is in the same direction and it is possible to halt until the new additional request level.

To determine if such a stop is possible the new parameter "requiredStopTime" is added to the input format. That parameter defines how long an elevator will need to stop.

Therefore the following equation specifies if the new request can be inserted:

$$travelStartTimestamp + travelTime\ from\ travelStartLevel\ to\ requestLevel \geq requestTimestamp + requiredStopTime$$

(where "travel" means a single run from one level to another)

Level 5 – Standard extended

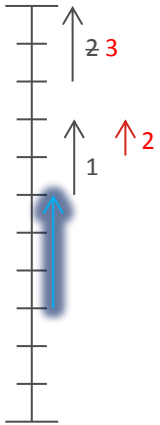
Restriction

An elevator does not add a request to the current processing if it travels to a request which is in the opposite direction or the same direction but before the current level (rule 2 respectively 3 of the previous level)

Hints

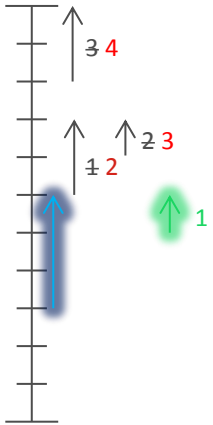
VIP control is needed later on again

Example 1



Explanation:
big blue arrow = the current request
black arrow = requests already queued
red arrow = new request transmitted while current request
numbers = new queue order

Example 2



Explanation:

big blue arrow = the current request

black arrow = requests already queued

big green arrow = new request transmitted before current request advances the threshold for stopping for the new request

numbers = new queue order

Input & Output

Input

requiredStopTime travelTimestamps…
numberOfRequests
    requestLevel requestGoal timestamp

Output

space separated request fulfillment timestamps in the **order of the input requests**.

Example input

2000 3993 6243 8493 10743…
2
0 20 10000
10 30 30000

Example output

58486 82729

Note

There are always 400 travelTimestamps.
In case of the given example input (so you can test it), the missing 396 values would always be the previous travelTimestamp + 2250.