Level 4 – Standard

Obviously a normal elevator works a bit different than the VIP elevator. So we introduce in this level a new “standard” elevator.

To do so, we go back to the single elevator model, that services all 400 levels (which also means, that there won’t be a start- and endlevel specified in the input).

This type won’t process the requests one by one in strictly chronological request order. But instead will pick the next request for processing with 3 priority rules explained on the next pages.

Of course when picking the next request it will only choose from already transmitted ones (= open requests). If no such requests exist at a time, the elevator gets idle.

Every time a new request gets transmitted after such an idle state, the elevator will still use its last travel direction for prioritization

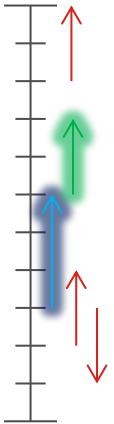
If there are multiple candidate requests left after filtering, the request first mentioned in the input file gets selected.

Hints

VIP control is needed later on again

Same direction

The highest priority is for requests with the same direction as the previous elevator travel direction where the request level is at or past the current elevator level.
If there is more than one such request, the closest one has highest priority

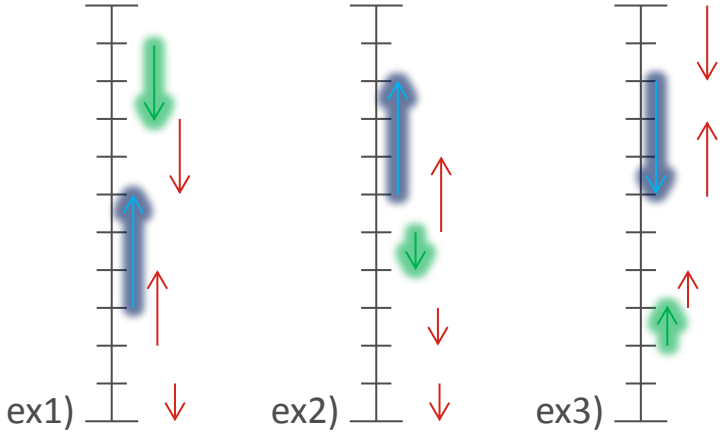


Explanation:
big blue arrow = the previous request
big green arrow = the selected request
red arrows = other open requests

Opposite direction

(Only if there is no request that matches rule 1)

Next on the priority list are requests with the opposite direction as the previous elevator direction. If there is more than one such request and previous travel direction was upwards, take the request with the highest request level (ex1 + ex2) or if the previous direction was downwards, take the request with the lowest request level (ex3).

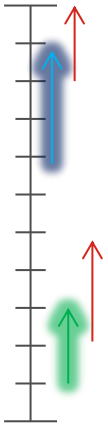


Explanation:
big blue arrow = the previous request
big green arrow = the selected request
red arrows = other open requests

Same direction, opposite collection

(Only if there is no request that matches rule 1 or 2)

The lowest priority will be for requests with the same direction as the previous elevator travel direction but where the request level is before the current elevator level.
If there is more than on such request, the one where the request level is furthest away (when looking in the opposite previous travel direction) will have the highest priority within that group.



Explanation:
big blue arrow = the previous request
big green arrow = the selected request
red arrows = other open requests

Input & Output

Input

travelTimestamps…
numberOfRequests
    requestLevel requestGoal timestamp

Output

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

Example Input

3993 6243 8493 10743…
3
0 10 10000
5 9 16000
40 30 17000

Example Output

34243 196465 127729

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.