7. DVD Elevator 2012

Program Name: DVDElevator2012.java Input File: dvdelevator2012.dat

You have taken an internship at Honey Badger Games working on a launch title for the new Phony BoxStation video game console. The game is currently loading too slowly and the lead programmer has asked you to implement an elevator loading algorithm to speed it up.

Data on optical disks, such as DVDs, are stored in a linear fashion from the inside to the outside of the disk. The laser always reads data in multiples of sector sizes as the laser moves across the disk, also from inside to outside. The laser cannot read any data when it is moving from the outside to the inside although it does take time to move the laser. You don't have direct control over the laser; you can only send file read requests to the operating system, which will process the read requests in the order it receives them.

An elevator algorithm is used to order requests in order to minimize unnecessary laser motion and increase disk read throughput by reading data as much of the time as possible when the laser is moving. Your boss has given you explicit instructions on how to implement the algorithm:

- By default the algorithm should be in an "idle" state while waiting for read requests.
- There are several rules for the laser position:
    - You can assume the operating system starts by having the laser positioned at the beginning of the first sector.
    - When a laser reads it starts at the beginning of a sector, and at the end of the read is positioned at the end of the sector just read (which is the beginning of the next sector). That means if the laser ends a read on sector 100 it is now actually on sector 101, so a read for sector 100 would require moving backwards.
    - In order to optimize reads, the algorithm should keep track of what would be the position of the laser at the end of the last read request sent to the operating system, hereafter referred to as the LEP (Laser End Position). Since the operating system may be in the middle of a read, the LEP is not necessarily the same as the current laser position.
- When a read request is received there are two possibilities for processing that request:
    - If the read would position the laser to start reading at or after the LEP, so that the laser can read immediately or move forward to read the data, the read request should be sent off right away to the operating system where it will be added to the end of the other requests.
    - If the read would position the laser to start reading before the LEP, so that the laser would have to move backward to read the data, the read request should be added to a batch list, as described below.
- The batch list should be processed as follows:
    - If the batch list is empty when a request is to be added, a 10-millisecond timer should be started so that all other requests received during that time can be put into a single request batch (where 1 millisecond = 1000 microseconds).
    - If the batch list isn't empty the request should be added to the list in sector read order, so all the requested data can be read as fast as possible without additional seeks. You can assume that no reads in any given batch will have their sectors overlapping.
    - When the timer on a batch in progress expires all the requests in the batch list should be dispatched to the operating system.
    - **Note**: Any request received at the exact end time of the batch time window should either be sent off before the batch or become part of the batch, depending on the read sector of the request and the LEP.
- When there is no work to do the system returns to the "idle state" with the laser positioned at the LEP, as described above.

The lead has asked that you test your code by writing a program that encompasses your algorithm and reads fake data from a text file in the order requests would be given to your code during the game.

(continued on next page)

7. DVD Elevator 2012 (cont.)

Input

There are an unknown number of lines in the input file. Each line will contain 3 long integer values:

- the time in microseconds since the start of the game until the request was made,
- the start sector for the request, and
- the end sector for the request.

The microsecond start time for any given line is guaranteed to never be lower than any previous lines (i.e., you can assume the order of input lines matches a timeline that never moves backwards).

Output

You should output all the requests from the input file, one per line, in the order they are sent to the optical disk hardware. Each line should be of the form "`Time X: Y to Z`" where `X` is the time in microseconds since the start of the game until the request is sent and `Y` and `Z` are the start and end sectors for the request respectively.

Example Input File

```
10 2 200
100 300 400
900 240 270
1100 220 230
10900 410 430
10901 0 10
10902 12 13
10903 11 12
11000 450 500
11001 440 450
```

Example Output to Screen

```
Time 10: 2 to 200
Time 100: 300 to 400
Time 10900: 410 to 430
Time 10900: 220 to 230
Time 10900: 240 to 270
Time 11000: 450 to 500
Time 20901: 0 to 10
Time 20901: 11 to 12
Time 20901: 12 to 13
Time 20901: 440 to 450
```