Disaster Recovery:

Your company has been using an archaic repository management system and is switching to a modern version control system. Unfortunately, much of the repositories' states were lost in the process of upgrading. Fortunately, you have a global log of commits applied to each repository that can be used to reconstruct much of the commit histories.

Each log entry corresponds to a commit applied to a particular repository and consists of a globally unique non-negative integral commit id, the non-negative integral timestamp of the commit, and the files changed in the commit. Each of these files is described by its path and an opaque identifier that's a function of the path and repository. Two commits for different repositories may contain the same file path, but the file's opaque identifier should be distinct, and vice-versa. And you may assume neither a file's path nor its opaque identifier contain whitespace.

To recover the commit histories, commits need to be combined to more accurately describe the set of commits applied to each repository. Commits can be combined with other commits containing at least one matching file path and opaque identifier. For example, the log below describes two repositories, where the first and last entries describe one of those two repositories:

id 0 timestamp 23534 include/foo.h da39a3 include/bar.h e615fe

id 2 timestamp 23534 include/baz.h 249e5d

id 1 timestamp 23606 include/bar.h e615fe src/bar.cpp 89bff2

The following log, however, is considered ambiguous because foo.py has conflicting opaque identifiers of ac819f and f918ca, but the three commits could otherwise be combined based on bar, py and baz. py:

id 38024 timestamp 74820 foo.py ac819f bar.py 0d82b9

id 49283 timestamp 19837 bar.py 0d8269 baz.py f28dc2

id 20391 timestamp 23488 baz.py f28dc2 foo.py f918ca

While the investigation is ongoing to better understand how the repositories' states were lost, you would like to provide employees with a tool to query commit histories based on the logs.

Specifically, queries are parameterized by a start timestamp, end timestamp, file path, and opaque identifier. The response for each query is the commit id of each commit corresponding to the repository referenced by the query's file path and opaque identifier, sorted by increasing timestamp followed by increasing commit id and filtered by the start and end timestamps, inclusive. Note that repositories should be determined based on all commits, not just the commits in the start and end timestamp range.

Input:

Input is provided via standard input. The first line of input consists of a single integer N representing the number of lines that follow, one per line (0 ≤ N ≤ 10'). Each line is either a valid log entry, or malformed.

* ﻿﻿Valid log entries consist of a line of words, delimited by a single space character.
* ﻿﻿Each word does not contain any whitespace.
* ﻿﻿Each two consecutive words make up a key-value pair and the first two keys are always 'id' followed by 'timestamp'.
* ﻿﻿Keys are always unique within a line.
* ﻿﻿The commit id and timestamp values are always non-negative 64-bit integers. All other words are opaque strings.
* ﻿﻿Commit IDs are guaranteed to be unique across lines.

Bonus: detect and discard malformed log entries that don't follow this format.

k

After the N log entries is another line consisting of a single integer R representing the number of queries that follow, one per line (0 ≤ R ≤ 105). Each query is a space-delimited sequence of start timestamp, end timestamp, file path, and opaque identifier.

6

id 8 timestamp 200 quicksort.cpp 839ad0 mergesort.cpp 0cdde1 bubblesort.cpp 248dd1

id 0 timestamp 500 array.h 163111 sequence.h 294d3f

id 6 timestamp 200 mergesort.cpp 0cdde1 bogosort.cpp 4213ff

id 4 timestamp 1000 array.h 163111 vector.h fcc2af

id 2 timestamp 300 bubblesort.cpp 248dd1 bogosort.cpp 4213ff

id 3 timestamp 300 bubblesort.cpp eaf88a bogosort.cpp 4f11aa

4

0 10000 quicksort.cpp 839ad0

0 500 vector.h fcc2af

0 100000 no\_found.h empty\_response

100 200 bogosort. cpp 4213ff

output

6 8 2

0

6 8

5

id 38024 timestamp 74820 foo.py ac819f bar.py 0d82b9

id 49283 timestamp 19837 bar.py 0d82b9 baz.py f28dc2

id 20391 timestamp 23488 baz.py f28dc2 foo.py f918ca can test this

id 2938 timestamp 101 gux.h d139af qux.cpp 718bc3

id 2939 timestamp 102 qux.h d139af

2

0 1000000 bar.py 0d82b9

0 1000000 qux.h d139af

Ambigious