Highest Scores

Overview

Given a data file containing scored records, in your favorite programming language, write a program to output the N highest record IDs ordered by descending score. The output should be well-formed JSON. Consider giving thought to the resource efficiency of your solution.

Detailed Specification

One of the really fun parts of Emerald is that we work with cutting-edge scientific instrumentation. One of the instruments our customers use frequently in the laboratory can analyze samples at a very high rate and generate scores for each sample based on the results of the run. We would like to know the highest scores that were seen over the course of a very large run.

Your task is to take a data file containing scored samples and produce the N highest scores and sample ids, ordered by descending score. The detailed requirements of the solution are described below.

Execution

Your solution should take two arguments: the path to the input data file and the number of records to output. For example:

```
python highest.py score_recs.data 5
```

Input format

The input data file is a series of key-value pairs (one per line) with the format `<score> : <record>`

In valid input files, the `score` is an integer and the `record` is a JSON dictionary. The `record` can be any kind of well-formed JSON doc (with the exception of no line breaks). The only constraint on the `record` is that a valid `record` will contain an `id` key that uniquely defines that record. All scores and ids are unique. A `record` that is not valid JSON or that does not contain an `id` field should be considered invalid and handled as described under **Exit Conditions**

An example input data file is:

```
8795136: {"id":"d2e257c282b54347ac14b2d8","x":"foo","payload":"someamountofdata"}
5317020: {"id":"619236365add4a0ca6e501fc","type":"purple","payload":"smalldata"}
.
.
.
2766123: {"id":"da9f77e6a0f076b000a6c0e0","payload":"reallyquitealotofdata"}
```

Data file generation

To test your code, we've included a sample node.js app that will generate 3 test files, one of which is rather large.

To generate the test files, run:

```
npm install chance
./gen.js
```

Output format

The output of your solution must be valid JSON in all cases (and you should consider validating that it is). The format of the output should be a list of JSON dictionaries, where each JSON dictionary has exactly two fields: `score` and `id`, which are the score and id of selected records respectively. The output must contain exactly N entries, which are the N highest scoring records in the input data file, and the records in the output must be sorted by descending order of score. An example output is:

```
$ python highest.py score_recs.data 5
[
    {
        "score":16774838,
        "id":"9ab7247c02044c65936a467016fff6b6"
    },
    {
        "score":16763774,
        "id":"c51a310f80604ef68a4cb2b83bffcb7e"
    },
    {
        "score":16761021,
        "id":"c1dbd109336242e0a64527ba8cffc0bd"
    },
    {
        "score":16755441,
        "id":"57b9ea55db954cbc8f452b34a2ffaaf1"
    },
    {
        "score":16753041,
        "id":"e8cafaf8cf2b41639422781fbdffa191"
    }
]
```

The output should be written directly to stdout and not written to an output file.

Exit Conditions

Upon successful running, your solution should exit with exit code 0. If the input data file is not valid, your solution should exit with code 2, while if the input file is not found, your solution should exit with code 1. Empty lines in the input file should be ignored rather than treated as invalid input.

Hint

Use a data structure that maintains a sorted data set and memory efficient.