Second Code Challenge Submission

Ioannis Karadimas

Third submission

Changelog - Submission 3

Improvement: Empty spaces at the end of company file lines

The match maker can now understand and correctly ignore empty spaces at the end of the companies file.

Changelog - Submission 2

New command added: validator

You can now validate results of the match making process by using a separate invocation, like below:

```
node --max_old_space_size=4096 validator.js ../Challenge\
Material/samples/large/companies.csv ../Challenge\
Material/samples/large/professionals.csv matches.csv
```

This will produce results like below:

```
Started
Validating couplings: ...................
Total match errors 0
Matches are valid
```

In case you encounter errors, they appear with the following form:

```
Started
Validating couplings: ...................
Professional [PRO_06023] would prefer to work in [COM_00117] which in
turn prefers him more than some of her current matches.
...
```

Correction: Replacing selected company professionals is now correct

This defect affected the way in which the algorithm replaces unstable couplings between companies and professionals with supposedly more stable ones. The previous approach was to replace the first selected professional whose rank was worse than the current subject's. This was initially estimated to improve performance for small to mid - level datasets, and it mostly works for simplistic examples, since it limits the necessary iterations to produce a viable result.

In larger datasets, however, it introduces errors that lead to performance degradation, since more errors lead in some cases to more replacements, with each replacement being an N $\log_N$ operation. The improvement introduced scales the total algorithmic complexity up into quadratic levels, but ensures that the worst selected employee is replaced within a company's hires, each time a better match has been found. This ultimately leads to fewer operations, and corrects any errors detected. This explains why the total run time stays more or less within the same levels as the previous submission with the given large sample. However, the gains may be more apparent with larger datasets, something that remains to be seen.

Correction: Total run time is now correctly reported.

A successfull execution of the match making invocation now produces output like the following:

```
Done. Total execution time (seconds): 16.469714146
```

Installation

This solution can be cloned from bitbucket after contacting I.Karadimas with a valid bitbucket e-mail account, in order to be able to keep up with any updates. Alternatively, a compressed file should accompany this document.

Cloning the repository

```
$ git clone https://bitbucket.org/yiannisk/linkedout
```

After either expanding the compressed file or cloning the repository, you need to install the packages this solution utilizes. This is accomplished by executing

```
$ cd <folder>/nodejs/
$ npm install
```

Usage

You can execute the solution with the provided (large) sample by giving the following command:

```
$ node --max_old_space_size=4096 main.js ../Challenge\
Material/samples/large/companies.csv ../Challenge\
Material/samples/large/professionals.csv
```

Note that this is a single line. This should produce the following lines of output:

```
Started
Done. Total execution time: 15 seconds or 971.340911 milliseconds
```

Along with a file called **matches.csv**, of a format compatible with the provided samples.

Caveats

Node.js has a default maximum heap size of about 1.7gb, hence the **--max_old_space_size** custom parameter value (specified in megabytes). Depending on your input size, you may encounter the error below. If you do, you should increase the maximum heap size to something bigger than what it currently is, and try again.

```
<--- Last few GCs --->

   62874 ms: Mark-sweep 1390.6 (1434.7) -> 1390.6 (1434.7) MB, 563.8 /
0.0 ms [allocation failure] [scavenge might not succeed].
   63474 ms: Mark-sweep 1390.6 (1434.7) -> 1390.6 (1434.7) MB, 600.1 /
0.0 ms [allocation failure] [scavenge might not succeed].
   64078 ms: Mark-sweep 1390.6 (1434.7) -> 1390.6 (1418.7) MB, 603.5 /
0.0 ms [last resort gc].
   64722 ms: Mark-sweep 1390.6 (1418.7) -> 1390.6 (1418.7) MB, 643.7 /
0.0 ms [last resort gc].
<--- JS stacktrace --->

==== JS stack trace =========================================
Security context: 0x395f107cfb51 <JS Object>
    2: professionalParser
```

```
[/home/yiannisk/projects/linkedout/nodejs/parsers.js:~47]
[pc=0x3d0192552839] (this=0x395f107e6119 <JS Global
Object>,line=0x1a0d1164631 <Very long string[20011]>)
    3: arguments adaptor frame: 3->1
    4: map [native array.js:1004] [pc=0x3d0192521222]
(this=0x30cac4564261 <JS Array[8001]>,bq=0x33dfcb8ce769 <JS Function
professionalParser (SharedFunctionInfo 0x1c1168...

FATAL ERROR: CALL_AND_RETRY_LAST Allocation failed - JavaScript heap out
of memory
 1: node::Abort() [node]
 2: 0x10c98dc [node]
 3: v8::Utils::ReportApiFailure(char const*, char const*) [node]
 4: v8::internal::V8::FatalProcessOutOfMemory(char const*, bool) [node]
 5: v8::internal::Factory::NewFillerObject(int, bool,
v8::internal::AllocationSpace) [node]
 6: v8::internal::Runtime_AllocateInTargetSpace(int,
v8::internal::Object**, v8::internal::Isolate*) [node]
 7: 0x3d01922079a7
Aborted
```

This is what an out-of-memory exception looks like in node.js

Environment

Node (node -v): 6.9.1
Npm (npm -v): 3.10.8, bundled up with Node v6.x but you may need to execute npm upgrade
OS (cat /etc/issue): Debian GNU/Linux 8 \n \l

Packages

```
yiannisk@ikaradimas-debian-vm:~/projects/linkedout/nodejs$ npm list
nodejs@1.0.0 /home/yiannisk/projects/linkedout/nodejs
├──┬ async@2.1.4 // synchronization, parallel execution, etc.
│  └─── lodash@4.17.2
├─── bluebird@3.4.6 // promises for node core libraries
└──┬ commander@2.9.0 // Command-line arguments parsing
   └─── graceful-readlink@1.0.1
```

References

1. Stable marriage problem - Wikipedia
2. Gale-Shapley algorithm (basically the same info but visually) - Video