CS600 Final Project - Search Engine

- CS600 Final Project - Search Engine
    - Attached Files
    - Details of approach to the problem
        - Inverted Index
            - Compressed Trie
        - Occurrence List
        - Ranking
        - Searching the inverted index
        - Web Crawler
    - Output from sample run
        - Sample Trie Test Run
        - Sample Run
    - Compile and Run
        - Stack
        - Running
        - Trie Test
    - Extras:
        - Libraries Used and Why
        - Stop Words
    - Assignment:

Attached Files

Project:

- searchEngine.js -- Main File to run the search engine
- invertedIndex.js
- crawler.js
- trie.js
- package.json
- stopwords.json

Testing:

- test.js

Report:

- README.md -- This file
- sampleRun.output.txt -- sample output from stdout.
- graphviz.gv -- graphviz generation for viewing the trie
- graphviz.pdf -- trie in pdf format for viewing
- sampleRun.png -- cropping of the trie for viewing

- test.output.txt -- sample output form sdtout when running sample test
- test.png -- png of the trie generated from the trie test
- input.txt -- search words used to test and show sample run
- node.png -- node data structure
- occurrenceList.png -- occurrence list data structure

Extra:

- .gitignore

Details of approach to the problem

A simplified search engine was implemented as per the details described in Section 23.5.4. A web crawler runs in the background and will index all the words in the pages of the site. The default site that is indexed is: https://nodejs.org/api/documentation.html . This crawler is running asynchronously, so once words are indexed, they can then be searched.

Inverted Index

Stores Key-Value pairs for easy lookup. The inverted index is implemented using a **Compressed Trie** with *variable length string* labels. As described in section 23.5.4, the inverted index should have been implemented with a compressed trie, but the type was not specified.

Key: The **Index Term** or word / part of a word that is used when searching

Value: location of the **Occurrence List**. In this case, the relative path that stores it. Since this is a simplified search engine, we are only storing one value. For a larger search engine, multiple files may be used to store occurrence lists and may be partitioned on multiple machines.

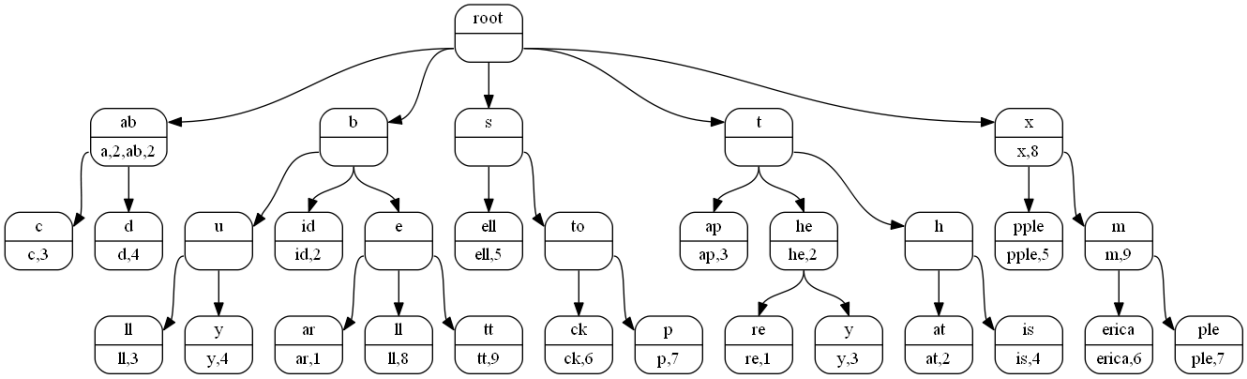
Compressed Trie

The compressed trie uses variable length string labels for an easier implementation.

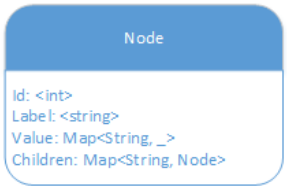
Insert and get methods take O(1) assuming that string lengths are constant.

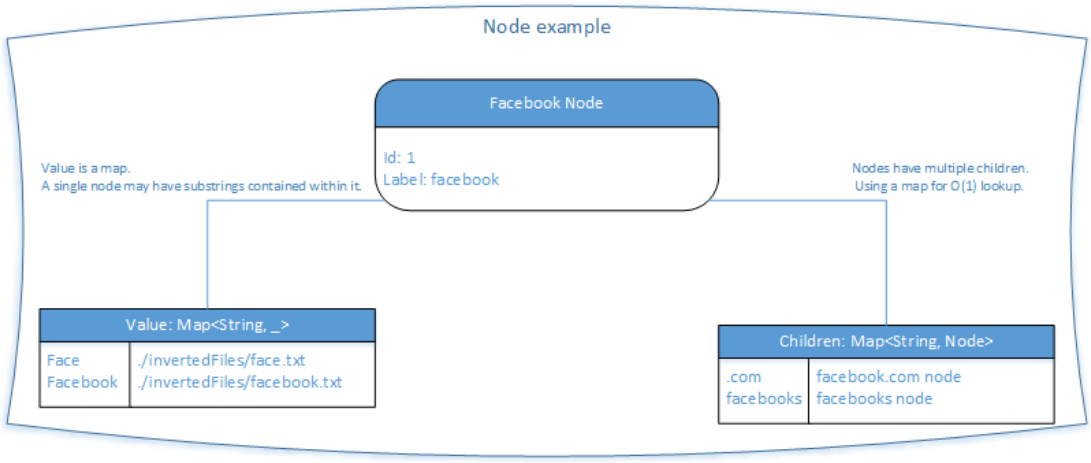
The value of a node uses a map. This is because 'a' and 'ab' may be stored in the same exact node 'ab' if the node 'ab' does not have any children.

An example of a compressed trie



A compressed trie is made up of several nodes:

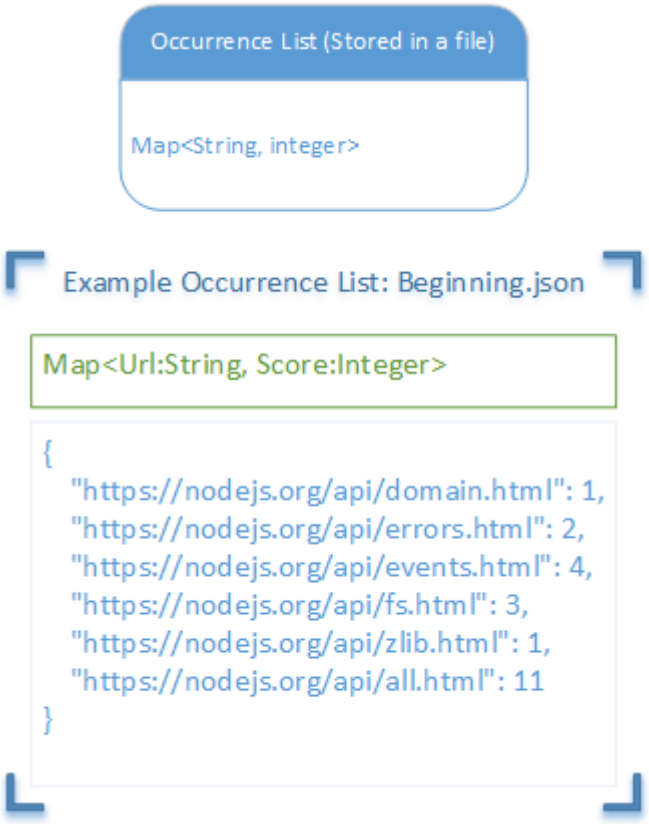




Occurrence List

Stored in a file in ./invertedFiles. This file contains a map of (key, values).

Key: The website url Value: A ranking for that website for this particular words.



Ranking

Websites are given a score based off of the number of times a word appears in the document. Before results are returned, websites are sorted by their score in descending order.

Searching the inverted index

1. Each word is found in the inverted index to obtain the location of the occurrence list.
2. The occurrence list file is opened and map (url, score) is read into memory.
3. Each list is reduced into one result by taking the intersection of each key. The values (scores) are added together.
4. Finally, the urls are sorted by score in descending order, and the results are returned to the user/

Web Crawler

Three libraries were used for the web crawler:

- simplecrawler
    - https://github.com/cgiffard/node-simplecrawler
    - Very simple web crawler
- striptags

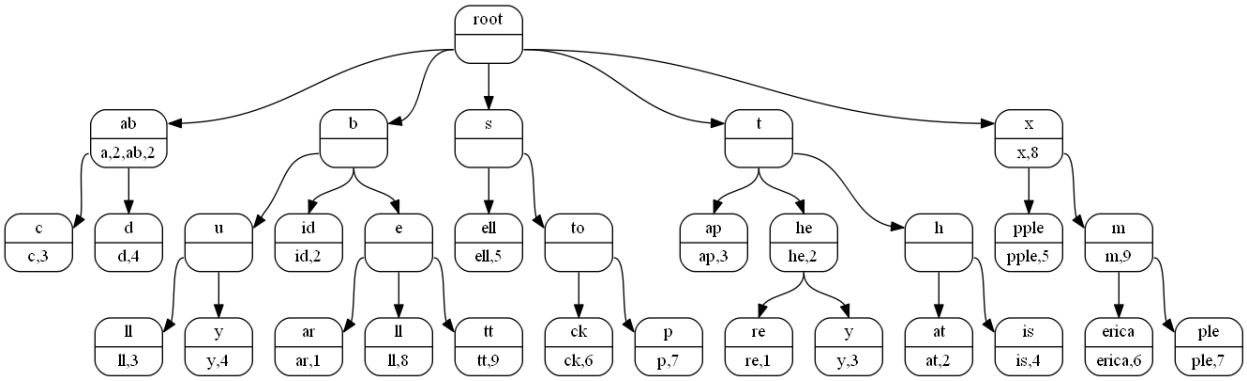
    - https://www.npmjs.com/package/striptags
    - Used to strip any tags in the page that was visited by the crawler
- cheerio
    - https://github.com/cheeriojs/cheerio
    - Used to easily go through the DOM of a website visited by the crawler

The crawler was configured to only go 2 pages deep.

Output from sample run

Sample Trie Test Run

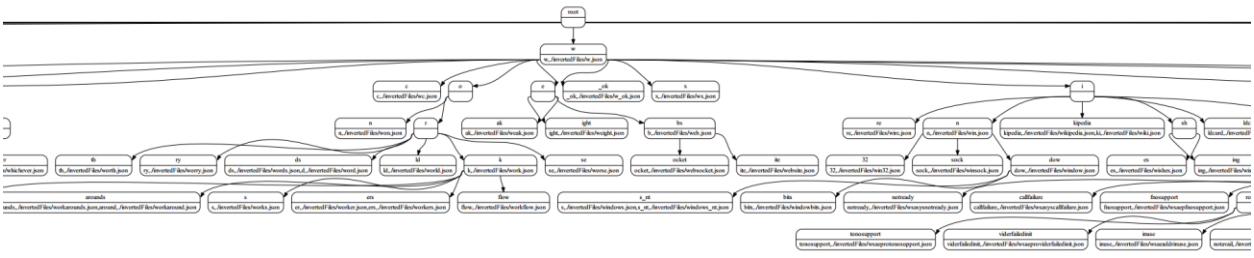
Used to test basic functionality of the compressed trie.




```
> 600@1.0.0 test H:\600
> node test.js > ./documentation/test.output.txt ; echo 'Output is located in
./documentation/test.output.txt'
digraph {
xroot0 [label="{<f0>root|<f1>}" shape=Mrecord];
xab2 [label="{<f0>ab|<f1>a,2,ab,2}" shape=Mrecord];
    xroot0:f1 -> xab2:f0;
xc1 [label="{<f0>c|<f1>c,3}" shape=Mrecord];
    xab2:f1 -> xc1:f0;
xd3 [label="{<f0>d|<f1>d,4}" shape=Mrecord];
    xab2:f1 -> xd3:f0;
xb5 [label="{<f0>b|<f1>}" shape=Mrecord];
    xroot0:f1 -> xb5:f0;
xu7 [label="{<f0>u|<f1>}" shape=Mrecord];
    xb5:f1 -> xu7:f0;
xll6 [label="{<f0>ll|<f1>ll,3}" shape=Mrecord];
    xu7:f1 -> xll6:f0;
xy8 [label="{<f0>y|<f1>y,4}" shape=Mrecord];
    xu7:f1 -> xy8:f0;
xid9 [label="{<f0>id|<f1>id,2}" shape=Mrecord];
    xb5:f1 -> xid9:f0;
xe15 [label="{<f0>e|<f1>}" shape=Mrecord];
    xb5:f1 -> xe15:f0;
xar4 [label="{<f0>ar|<f1>ar,1}" shape=Mrecord];
    xe15:f1 -> xar4:f0;
xll16 [label="{<f0>ll|<f1>ll,8}" shape=Mrecord];
    xe15:f1 -> xll16:f0;
xtt17 [label="{<f0>tt|<f1>tt,9}" shape=Mrecord];
    xe15:f1 -> xtt17:f0;
```

```
xs11 [label="{<f0>s|<f1>}" shape=Mrecord];
    xroot0:f1 -> xs11:f0;
xell10 [label="{<f0>ell|<f1>ell,5}" shape=Mrecord];
    xs11:f1 -> xell10:f0;
xto13 [label="{<f0>to|<f1>}" shape=Mrecord];
    xs11:f1 -> xto13:f0;
xck12 [label="{<f0>ck|<f1>ck,6}" shape=Mrecord];
    xto13:f1 -> xck12:f0;
xp14 [label="{<f0>p|<f1>p,7}" shape=Mrecord];
    xto13:f1 -> xp14:f0;
xt21 [label="{<f0>t|<f1>}" shape=Mrecord];
    xroot0:f1 -> xt21:f0;
xap22 [label="{<f0>ap|<f1>ap,3}" shape=Mrecord];
    xt21:f1 -> xap22:f0;
xhe23 [label="{<f0>he|<f1>he,2}" shape=Mrecord];
    xt21:f1 -> xhe23:f0;
xre18 [label="{<f0>re|<f1>re,1}" shape=Mrecord];
    xhe23:f1 -> xre18:f0;
xy24 [label="{<f0>y|<f1>y,3}" shape=Mrecord];
    xhe23:f1 -> xy24:f0;
xh25 [label="{<f0>h|<f1>}" shape=Mrecord];
    xt21:f1 -> xh25:f0;
xat20 [label="{<f0>at|<f1>at,2}" shape=Mrecord];
    xh25:f1 -> xat20:f0;
xis26 [label="{<f0>is|<f1>is,4}" shape=Mrecord];
    xh25:f1 -> xis26:f0;
xx28 [label="{<f0>x|<f1>x,8}" shape=Mrecord];
    xroot0:f1 -> xx28:f0;
xpple27 [label="{<f0>pple|<f1>pple,5}" shape=Mrecord];
    xx28:f1 -> xpple27:f0;
xm30 [label="{<f0>m|<f1>m,9}" shape=Mrecord];
    xx28:f1 -> xm30:f0;
xerica29 [label="{<f0>erica|<f1>erica,6}" shape=Mrecord];
    xm30:f1 -> xerica29:f0;
xple31 [label="{<f0>ple|<f1>ple,7}" shape=Mrecord];
    xm30:f1 -> xple31:f0;
}
```

Sample Run



Search edge cases tested. The words used can be found in .documentation/input.txt: - Empty String - One word - Two words - Three words - Two words and then narrowing results with 3 words - Word that will have no results

A graphviz file and pdf of the generated trie are located in the documentation folder. graphviz.gv graphviz.pdf

A cropping of the generated trie is shown in: sampleRun.png

```
H:\600>npm start

> 600@1.0.0 start H:\600
> node searchEngine.js

Url to start indexing (default: https://nodejs.org/api/documentation.html):
Type exit to exit at any time, or press ctrl+c.
Search: Done indexing the interwebs

No results
Search:
No results
Search: node
[ 'https://nodejs.org/api/all.html',
  'https://nodejs.org/api/process.html',
  'https://nodejs.org/api/addons.html',
  'https://nodejs.org/api/modules.html',
  'https://nodejs.org/api/child_process.html',
  'https://nodejs.org/api/repl.html',
  'https://nodejs.org/api/errors.html',
  'https://nodejs.org/api/stream.html',
  'https://nodejs.org/api/cli.html',
  'https://nodejs.org/api/debugger.html',
  'https://nodejs.org/api/crypto.html',
  'https://nodejs.org/api/http.html',
  'https://nodejs.org/api/fs.html',
  'https://nodejs.org/api/timers.html',
  'https://nodejs.org/api/cluster.html',
  'https://nodejs.org/api/buffer.html',
  'https://nodejs.org/api/dgram.html',
  'https://nodejs.org/api/globals.html',
  'https://nodejs.org/api/deprecations.html',
  'https://nodejs.org/api/console.html',
  'https://nodejs.org/api/os.html',
  'https://nodejs.org/api/documentation.html',
  'https://nodejs.org/api/tracing.html',
  'https://nodejs.org/api/tls.html',
  'https://nodejs.org/api/tty.html',
  'https://nodejs.org/en/',
  'https://nodejs.org/api/path.html',
  'https://nodejs.org/api/synopsis.html',
  'https://nodejs.org/api/events.html',
  'https://nodejs.org/api/readline.html',
  'https://nodejs.org/api/v8.html',
  'https://nodejs.org/api/net.html',
  'https://nodejs.org/api/punycode.html',
  'https://nodejs.org/api/dns.html',
    'https://nodejs.org/api/zlib.html',
  'https://nodejs.org/api/util.html',
  'https://nodejs.org/api/https.html',
  'https://nodejs.org/api/domain.html',
  'https://nodejs.org/api/vm.html',
  'https://nodejs.org/api/querystring.html',
  'https://nodejs.org/api/url.html',
  'https://nodejs.org/api/index.html',
  'https://nodejs.org/api/assert.html',
  'https://nodejs.org/api/string_decoder.html' ]
Search: node html
[ 'https://nodejs.org/api/all.html',
  'https://nodejs.org/api/path.html',
  'https://nodejs.org/api/crypto.html',
```

```
  'https://nodejs.org/api/debugger.html',
  'https://nodejs.org/api/http.html',
  'https://nodejs.org/api/documentation.html',
  'https://nodejs.org/api/tls.html',
  'https://nodejs.org/api/zlib.html' ]
Search: node asdf
[ 'https://nodejs.org/api/all.html',
  'https://nodejs.org/api/path.html' ]
Search: node asdf file
[ 'https://nodejs.org/api/all.html',
  'https://nodejs.org/api/path.html' ]
Search: node html path
[ 'https://nodejs.org/api/all.html',
  'https://nodejs.org/api/path.html',
  'https://nodejs.org/api/http.html',
  'https://nodejs.org/api/crypto.html',
  'https://nodejs.org/api/debugger.html',
  'https://nodejs.org/api/tls.html',
  'https://nodejs.org/api/documentation.html',
  'https://nodejs.org/api/zlib.html' ]
Search: node html path asdf
[ 'https://nodejs.org/api/all.html',
  'https://nodejs.org/api/path.html' ]
Search: node keys
[ 'https://nodejs.org/api/all.html',
  'https://nodejs.org/api/repl.html',
  'https://nodejs.org/api/crypto.html',
  'https://nodejs.org/api/tls.html',
  'https://nodejs.org/api/http.html',
  'https://nodejs.org/api/buffer.html',
  'https://nodejs.org/api/https.html',
  'https://nodejs.org/api/readline.html',
  'https://nodejs.org/api/url.html',
  'https://nodejs.org/api/querystring.html',
  'https://nodejs.org/api/util.html' ]
Search: node keys buffer
[ 'https://nodejs.org/api/all.html',
  'https://nodejs.org/api/buffer.html',
  'https://nodejs.org/api/crypto.html',
  'https://nodejs.org/api/tls.html',
  'https://nodejs.org/api/repl.html',
  'https://nodejs.org/api/http.html',
  'https://nodejs.org/api/https.html',
  'https://nodejs.org/api/util.html',
  'https://nodejs.org/api/readline.html',
  'https://nodejs.org/api/url.html',
  'https://nodejs.org/api/querystring.html' ]
Search: node keys util
[ 'https://nodejs.org/api/all.html',
  'https://nodejs.org/api/util.html',
  'https://nodejs.org/api/repl.html' ]
Search: invalidword
No results
Search: exit
```

Compile and Run

Stack

Latest version of node

---------- **MAKE SURE YOU RUN:** ----------

```
npm install
```

Running

```
npm start
```

Then enter a website to index. Search for words while the website is being indexed.

Trie Test

Simple tests for the trie.

```
npm test
```

Extras:

Libraries Used and Why

- simplecrawler
  - https://github.com/cgiffard/node-simplecrawler
  - Very simple web crawler
- striptags
  - https://www.npmjs.com/package/striptags
  - Used to strip any tags in the page that was visited by the crawler
- simplecrawler
  - https://github.com/cheeriojs/cheerio
  - Used to easily go through the DOM of a website visited by the crawler
- rmdir
  - https://www.npmjs.com/package/rmdir
  - Recursively remove directory

Stop Words

All stop words were acquired from the following url:

- http://algs4.cs.princeton.edu/35applications/stopwords.txt

Assignment:

```
Project: Search Engine
```

Implement the simplified Search Engine described in Section 23.5.4 for the pages of a small Web site. Use all the words in the pages of the site as index terms, excluding stop words such as articles, prepositions, and pronouns.

Submit the following four files:

1. A read me file that contains details of your approach to the problem.

2. Your coded, well-commented code file in your favorite language, such as Python, Java, C++,...

3. The input file that contains the few pages you have used as input, including some links to your other pages..

4. Output file that has samples of your run. Make sure you have tested the boundary conditions.

Q and A:

1. Should I use internet documents or create my own?

Answer: You may download several pages (5 to 10) from Internet, or develop them yourself, and use them as input. Make sure there are some hyperlink between the documents and you can add more pages to your existing ones that is gathered by Web Crawler.

2. Can I use Other Search Engine or Data Structures?

Answer: No. You shall implement what is described in Section 23.5.4. using the approach specified. You are not allowed to use Algorithms and Data Structures that are not covered in the textbook, but you can develop new algorithms based on the data structures that you have seen and proved in the textbook.

3. What criteria should I use for ranking?

Answer: I leave that to you to come up with some simple criteria and algorithm to implement ranking, such as the number of times a word has appeared in the document or any other idea you have. But please explain your approach.

4. Can I provide screenshots of my output?

Answer: Yes.