

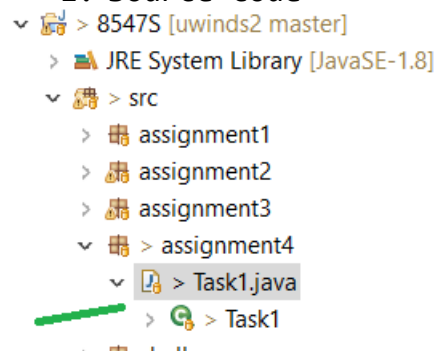
Assignment 4

As a student at the University of Windsor, I pledge to pursue all endeavours with honour and integrity and will not tolerate or engage in academic or personal dishonesty. I confirm that I have not received any unauthorized assistance in preparing for or writing this assignment. I acknowledge that a mark of 0 may be assigned for copied work.

Wen Dong #110057395

Task #1

1. Source code



2. Output of the java file looks as below:

Indices found by BoyerMoore:

hard: [159, 212, 442, 465, 476, 3585, 3787, 8850, 8948, 9158, 11982, ...

disk: [5, 140, 217, 447, 470, 496, 639, 3590, 3792, 4571, 4940, 4980 ...

hard disk: [212, 442, 465, 3585, 3787, 9158, 11982, 17219, 17559, 19252, ...

hard drive: [476, 8850, 8948, 18019, 25221, 30034, 34130, 51400, 56579]

hard dist: []

xltpu: []

Searching patterns for 100 times with each algorithm, average CPU time:

BoyerMoore: 1790893 ns

KMP: 2205835 ns

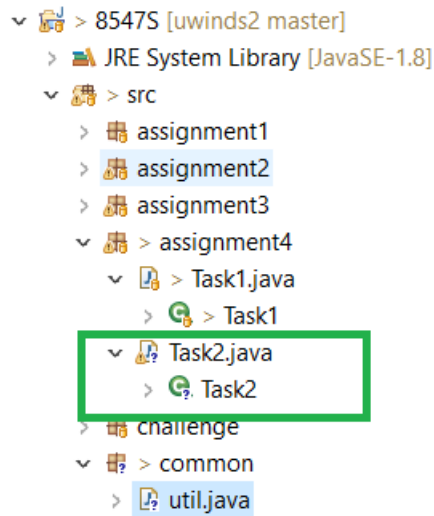
BruteForceMatch: 2301495 ns

Comments:

As we learned from class, BruteForceMatch runs at time complexity $O(nm)$ where n is the length of search string and m is the length of pattern. BoyerMoore is of worst-case time complexity $O(nm + s)$ where s is the size of alphabet, the worst-case may occur in images and bio-sequences, but unlikely in English text, for English text, it runs significantly faster than BruteForceMatch. KMP algorithm runs in optimal worst-case time $O(m+n)$. as we can see from the output of the program, it is consistent with what we learned in regard to the algorithm performance, i.e. BoyerMoore runs much faster than BruteForceMatch, and KMP runs a bit faster than BruteForceMatch.

Task #2 - a

1. Source code



2. Output of the java file looks as below:

Searching keys - [protein, complex, PPI, prediction, interaction]

Get keys by complete match-----

protein - [97, 207, 237, 245, 418, 426, 721, 729, 885, 2362, 2370]

complex - [159, 1211, 1331, 1564]

PPI - [1619]

prediction - null

interaction - [105, 253, 1283, 1963, 2031]

Get keys by prefixMatch-----

protein - [97, 207, 237, 245, 418, 426, 721, 729, 885, 2362, 2370]

proteins - [186, 804, 957, 1172, 1845]

complex - [159, 1211, 1331, 1564]

PPI - [1619]

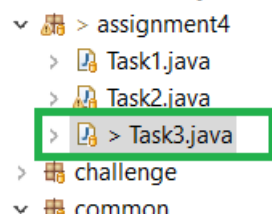
predictions - [1304]

interaction - [105, 253, 1283, 1963, 2031]

interactions - [434, 737, 893, 1825, 2378]

Task #3

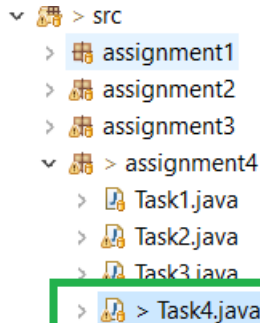
3. Source code



4. Run the java file and generate text file in folder output/

Task #4

1. Source code



```
src
├── assignment1
├── assignment2
├── assignment3
├── assignment4
│   ├── Task1.java
│   ├── Task2.java
│   ├── Task3.java
│   └── Task4.java
```

2. Output of the java file looks as below:

Found 173 emails, out of them 104 unique emails:

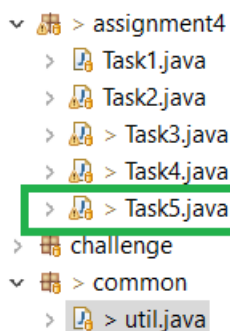
[public-pfwg-comments@w3.org, www-multimodal@w3.org, www-multimodal-request@w3.org, public-gld-comments@w3.org, ohto@w3.org, lalitha@tri.sbc.com, johan.hjelm@nrj.ericsson.se, www-mobile@w3.org, www-component-extension@w3.org, liam@w3.org, ...]

Found 107 phone numbers, out of them 68 unique phone numbers:

[185788651, 1065075071, 2147483648, +2147483647, 2147483647, 4294967295, 9223372036, 854775808, 854775807, 1844674407, 3709551615, 4294967296, 3709551616, (650) 812-4763, (650) 812-4777, 0123456789, 3398648111, 121.8850708, 201-555-0111, 2343543645, ...]

Task #5

1. Source code



```
assignment4
├── Task1.java
├── Task2.java
├── Task3.java
├── Task4.java
├── Task5.java
├── challenge
├── common
├── util.java
```

2. Output of the java file looks as below:

Found 9482 Links with domain w3.org ==> output/Linkswithdomainw3.org.dat
Found 2998 Links that contain references => output/Linksthatcontainreferences.dat
Found 10167 Links that contain folders ==> output/Linksthatcontainfolders.dat
Found 10580 Links with domain .net .com .org ==> output/Linkswithdomain.net.com.org.dat
output/AS.dat