Julian Weiss julian.weiss@rochester.edu

# CSC 172 PROJECT — THE SCIENCE OF DATA STRUCTURES
# **Hashing and Sorting**

*Hashing and Sorting* was one of the most extensive and intriguing, although ultimately very challenging and stubborn, programming tasks I've ever undertaken. When creating the *concordances* for (a) the Shakespearean Sonnets, (b) the set of most frequently used words today (pulled from an online source— 6,000 uniques), (c) the provided dictionary text file (presumably taken from a version of Linux) , and (d) the Barack Obama Reddit AMA, I chose a simplistic, although still sufficiently efficient algorithm for hashing. The primary block of code that determined the hashtable's specific functions was in the hashtable class I created:

```
int index = word.word.compareTo("") % capacity;
for( ; table[index] != null && !table[index][0].word.equals(word.word);
     index = (index+4)%capacity);
```

The idea here is to choose the word's index depending on the current capacity of the hash table, using quadratic probing if excessive collisions occur. If the same *key* is used as is present in the currently accessed hashtable element, it's added to the end of the current element's array (the hashtables are tables of *PlottedWord* arrays). However, this isn't problem-proof, and later in the same method I make sure to include an *expand* functionality that allows for increasing of hashtable size (utilizing prime numbers of two-orders of magnitude greater than the current capacity). Most of the additional time that's tacked on in this process stems from the algorithm I used to convert the external file into memory— a process where I prioritized legibility and understandability over efficient concepts I might not have a firm grasp over yet. In the future, further iterations of this concept would of course utilize lower-level buffers and tokenizers, and invest less effort over the algorithm that was required for this particular assignment.

Each set of words was chosen mostly out of curiosity— a site I go on regularly, Reddit, would contain the average amount of words I use during my day— is that equivalent to Shakespeare's daily vocabulary? And I also pulled 6,000 words from a frequency listing to see if my own sources could compare to the dictionary provided (and surely, they didn't). The HTML/content parsing I did could've certainly been better (especially because using the HTML parsing technique I did would be a nightmare in real-world applications), but it was more than adequate for the concordances.

Running the code is fairly straightforward. Everything will be processed entirely automatically except for the concordance index retrieval portion— which occurs right after the sonnets are loaded into a hashtable. The attached example output gives a firm demonstration of how to utilize that area of the program, if it's at all unclear.

The sorting algorithms used were the built-in Java functions, as they clearly optimized for speed, and it was never necessary to prescribe my own (which would nearly mimic a quicksort, except using *.compareTo()* rather than numerical comparisons). The important (and project-required/relevant) algorithm, however, was in checking the matches from Shakespeare's sonnets to the various other sources (the tricky part)— I had to create my own "merge," such as that seen in mergesorts, except I had to account for duplicates (thus, I created ArrayLists of the keys before sending them away), and deal with iterating through every element of each array (impossible to get anything better than $O(n_1n_2)$). Though, it worked beautifully for every sample!

**Elizabethanisms** (I prefer to call them Shakespeareans) **are written to the output.txt file, but I also attached them formatted at the end of this document.** The identical result is replicable by running the program (it will create its own output.txt).

Every included *.txt* file except for *output.txt* are currently used in the program as alternate concordances, and are obviously linked with methods named after them. The three *.java* files are: *SonnetsConcordance.java*, which does most of the heavy-lifting for the program itself (it's the Driver, as well as writer/reader); *Hashtable.java*, which contains my custom Hashtable implementation; and *PlottedWord.java*, my specific class that's used in the hashtables, which only specifies key and coordinate (word, line, wordNumber in-line).

## Complete Output

```
Launched sonnet program, pulling sonnets (this might take a while)...
Done creating sonnet concordance! Took 0.463 seconds for 17516 words.
That means there were 17516 words in all of the sonnets combined (with repeats).
Done sorting sonnet concordance, found 3213 unique elements.
That means only 3213 words were used in all of Shakespeare's sonnets

Give me a word and I'll tell you where it is in the sonnets! Type "stop" to move on.
although
Found word at following (line, word) coordinate(s):
(887, 0) (963, 0) (1026, 3) (1230, 4) (1457, 2) (1654, 0) (2254, 3) (2507, 0)
(2624, 0)
bedvow
Found word at following (line, word) coordinate(s):
(2860, 3)
obama
Couldn't find word!
stop

Parsing and pulling common words (this will take way longer)...
Done creating common words concordance! Took 0.279 seconds for 6036 words.
That means the sample I used only had 6036 words, total.
Done sorting common concordance, found 6019 unique elements.
That means the sample had 6019 words without repeats (depends on sample)
In comparison to Shakespeare, our most common words are 1.8733271086212262 as big!

Done comparing and retrieving Shakespearean words using common source! Took 0.0020
seconds to pull 1704 words.
That means, using the common source, there are apparently 1704 Shakespearisms (not
used today).

Parsing and pulling dictionary words (this will take a while)...
Done creating dict words table! Took 1.598 seconds for 454611 words.
Done comparing and retrieving Shakespearean words using dictionary! Took 0.03
seconds to pull 342 words.
That means, using the dictionary, there are actually 342 Shakespearisms (not used
today).
In comparison to Shakespeare, our full dictionary is 141.49112978524744 as big!

Let's try comparing against a better sample of everyday word usage-- the Barack
Obama AMA. Filling table (this will take the longest time)...
Done creating Obama concordance! Took 3.902 seconds for 40684 words.
Done sorting Obama concordance, found 5565 unique elements.
Done comparing and retrieving Shakespearean words using Obama AMA! Took 0.0010
seconds to pull 2523 words.
That means, using the Obama AMA, there are about 2523 Shakespearisms (not used
today).
In comparison to Shakespeare, our Obama-interviewing vocabulary is
1.7320261437908497 as big!


Thanks for playing!
```

## Elizabethanisms/Shakespeareans (output.txt)

```
abusd addeth          damaskd               featurd               lookd
adoting               dearpurchasd          feedst                lourst
advisd                deathdear             feelst                lovd
allayd                debateth              figurd                lovegod
alltheworld           decayd                fild                  lovekindling
alterd                deceivd               filld                 lovesuit
amazeth               deceivest             fixd                  lovst
anchord               decembers             flatterd              madmens
annexd                deemd                 fleshin               maketh
anothers              defacd                foild                 makst
aprils                deformedst            follydoctorlikeco     manys
assaild               deliverd              ntrolling             maskd
bearst                departest             forgetst              matcheth
beated                deservd               franticmad            merchandizd
beautys               deservst              freezings             mightst
bedvow                despisd               frownst               milliond
beggard               devisd                gatherd               miscalld
beguild               dialhand              gavst                 misplacd
belovd                dians                 gazeth                missd
besmeard              diest                 gildst                mixd
bestowst              dimmd                 gluttoning            mortgagd
betterd               disarmd               goest                 murdrous
bettring              disdaineth            grievd                needst
blamd                 diseasd               growst                neercloying
blessd                disgracd              hammerd               neighno
blessedfair           disposd               happies               newappearing
bodys                 distemperd            hateth                newfired
borrowd               distilld              heartinflaming        nourishd
buriest               doublevantage         helens                nurseth
burnd                 downrazd              ifi                   oerchargd
calld                 draind                illusd                oergreen
cancelld              dressd                illwresting           oerlook
carvd                 dyd                   impannelled           oerpressd
characterd            eisel                 impeachd              oerread
chargd                eithers               imprisond             oersnowed
cherubins             endowd                incertainties         oersways
childrens             endurd                intermixd             oertake
choppd                engrossd              junes                 oerworn
climbd                enjoyd                keepst                oppressd
colourd               enlargd               knowst                outstrippd
compard               errd                  languishd             overgoes
compild               esteemd               laughd                owst
conceald              everfixed             leapd                 passd
confessd              exchangd              learnd                perceivd
confind               expressd              learneds              perceivst
conquerd              eyd                   lifes                 perfectst
consumd               fadeth                lilys                 perjurd
consumst              falsespeaking         livd                  picturd
convertest            famishd               livesuch              piercd
crossd                famoused              lockd                 pitywanting
crownd                favourites            lodgd                 playd
crushd                featherd              longlivd              playst
```

poisond
polishd
possessd
possesseth
pourst
praisd
praisethat
presentabsent
presenteth
presentst
prevaild
preventst
prickd
privilage
profand
proposd
proudpied
provd
purposd
puttst
raisd
rangd
razd
rebukd
receivest
receivst
reckond
recurd
reeleth
refigurd
refusest
reignd
rememberd
removd
renewd
renewest
respose
restord
returnd
reviewest
richproud
robbd
rudst
ruind
runnst
savd
scapd
scornd
seald
seemd
seest
seeting
selfdoing
selfkilld
selfwilld
sendst
sharpst
shortnumberd
showst
slanderd
slandring
soulsgive
specialblest
spendst
staind
staineth
stampd
steeld
stelld
stickst
stirrd
stoln
strengthend
strumpeted
subdud
sufferd
sufficd
suretylike
swallowd
swartcomplexiond
swayst
sweetseasond
sweetst
sympathizd
tatterd
tempteth
tenderd
termd
theemyselfthat
thenchurlstheir
theyhast
thismy
thouall
thoughtsfrom
tird
toild
tonguesthe
touchd
transferrd
travelld
trimmd
truetelling
turnd
twicein
uneard
unfatherd
unlookd
unswayd
unthrifts
untrimmd
untutord
unwood
uplocked
usest
valleyfountain
vanishd
vexd
viewest
vowd
wakend
wanderst
warmd
weret
whoeer
whos
widowd
wilfulslow
womens
worthwide
wrackd
wretchcd
wretchs
yellowd
yourselfs