Annelise Thorn (anth6800)
Arwyn Lewis (arle8898)

DSA Group Project 2 Report

Question 1- Build a Table of Contents:

1. **Describe your approach, data structure used, and method to traverse the three and populate your tree**

For the first question, we wanted to create a table of contents for a Data Science technical book. We chose the book “Data Science: The Hard Parts” (Daniel Vaughan) and created classes and functions to add chapters/subchapters to print the table of contents of this book. We more closely followed case 1 for this question, meaning that the node is the main structure (only one class is implemented) and each node has a list of subchapters representing the child nodes. The “Node” class serves as the main structure, representing both the chapters and the subchapters. Each node also contains a title, which holds the text for each of the chapters or subchapters. The numbering is handled in the “print_toc” method when printing the chapters.

The Node class represents a chapter or subchapter in the book. The “add_subchapter” method creates a new subchapter as an instance of the Node class, and then adds the subchapter to the subchapters list. It then returns the newly created subchapter child node. The “print_toc” method uses recursion to print the title of the node and then its subchapters. It calculates the indentation based on the level of the chapter, and keeps track of the chapter numbers in “chapter_num”. It will loop through all the subchapters and call “print_toc” on each subchapter with the increased indentation. The “add_level_chapter” function is used to add a new chapter or subchapter to the specific level of the table of contents. It takes the current node at which a subchapter will be added to, the level of the hierarchy, and the title of the chapter/subchapter.

The “book” variable is serving as the root of the tree. It represents the top level of the top-level “chapter” of the book, and all subsequent chapters/subchapters are added below it. The subchapters serve as the child nodes, where within each Node instance there is a reference to the child nodes. There is a hierarchical structure to the tree, which allows for each node to have multiple children, but each child can only have one parent. It uses a recursive traversal method, since it prints the titles of the nodes while still maintaining the hierarchical structure with the addition of indentations.

2. **Provide the code for your classes/functions with comments to make it clear for a reader**

   See .ipynb file for code

3. **Provide a test code to validate your solution**

   See .ipynb file for code

4. **Challenge: Implement methods to calculate the depth of any given chapter and the height of the overall tree (This will not be graded)**

   See .ipynb file for code

Annelise Thorn (anth6800)
Arwyn Lewis (arle8898)

Question 2- Counting Letters/Words:

1. **Describe your approach to your solution. Justify your decision about the data structures selected and the algorithms used (if applicable). Discuss the complexity in a BIG O Notation when applied**

Distribution of Letters

To find the distribution of letters, we converted the text to lower case to count all instances of the same letter regardless of case, created a set and looped through the set to get each distinct letter in the text, and stored the frequency of each letter as tuples in a list. We used a set to eliminate duplicate letters in our loop and we used a list of tuples for storing frequency of letters because it allows us to easily append items. The time complexity for this algorithm is O(n) because the most time dependent operation is counting the occurrence of each letter and there are only 26 letters in the alphabet, it doesn't significantly affect the performance of the algorithm as n grows.

Top 40 Words

To find the 40 most common words, we used the split function to split the string into individual words, converted the words to sets, created a list of tuples for each unique word, sorted the word frequencies, and returned teh top 40 words. We used a set to eliminate duplicates in the loop and used a list of tuples for storing the frequency of words so we can easily append items and later sort through the list. The time complexity of the algorithm is O(wlogw) where w is the number of words in *Alice's Adventures in Wonderland*, because the sorting steps is the most time consuming part of the algorithm and needs to look at each word to compare it to others to put them in order. Overall, the time increases as the number of words grows.

Top 20 Bigrams

To find the 20 most common bigrams, we used the split function to split the string into individual words, created bigrams by iterating through the list of words and pairing each word with the one that follows it, appending these tuples to the bigrams list, counting the frequency of the bigrams, and sorting the bigrams. We used a set to eliminate duplicates in the loop and used a list of tuples for storing the frequency of bigrams so we can easily append items and later sort through the list. The time complexity of the algorithm is O(wlogw) where w is the number of words in *Alice's Adventures in Wonderland*, because the sorting steps is the most time consuming part of the algorithm and needs to look at each bigram to compare it to others to put them in order. Overall, the time increases as the number of bigrams grows.

Top 20 Trigrams

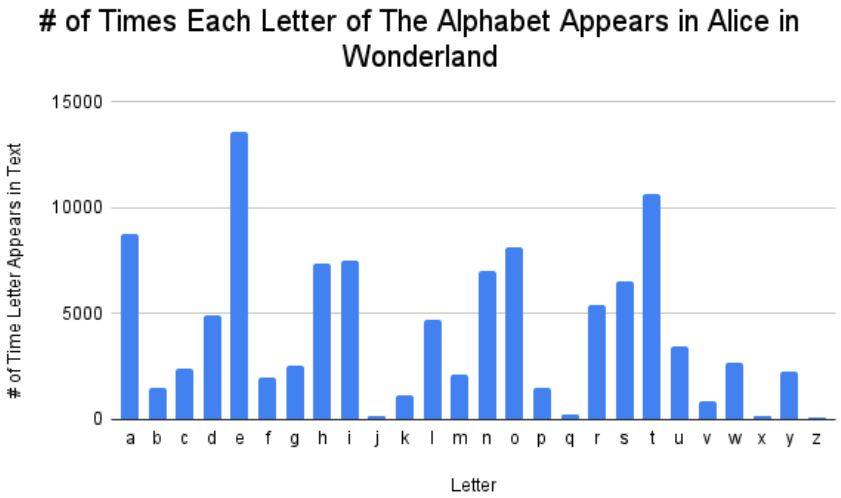
To find the 20 most common trigrams, we used the split function to split the string into individual words, created trigrams by iterating through the list of words and pairing each word with the next two consecutive words, counting the frequency of trigrams, and sorting the trigrams. We used a set to eliminate duplicates in the loop and used a list of tuples for storing the frequency of trigrams so we can easily append items and later sort through the list. The time complexity of the algorithm is O(wlogw) where w is the number of words in *Alice's Adventures in Wonderland*, because the sorting steps is the most time consuming part of the algorithm and

Annelise Thorn (anth6800)
Arwyn Lewis (arle8898)

needs to look at each trigram to compare it to others to put them in order. Overall, the time increases as the number of trigrams grows.

2. **Identify the distribution of each letter from a to z, including lower and upper cases. For example, *A* has the same treatment as *a*. ('A' is equal to 'a'). Provide frequencies in a summary table and a chart**

```
a: 8794
b: 1475
c: 2400
d: 4934
e: 13579
f: 2001
g: 2531
h: 7374
i: 7516
j: 146
k: 1158
l: 4718
m: 2107
n: 7020
o: 8147
p: 1524
q: 209
r: 5440
s: 6502
t: 10689
u: 3469
v: 847
w: 2676
x: 148
y: 2262
z: 78
```



3. **Remove stopwords (if you want, you can use the NLTK library to get the list of stopwords, nothing else)**

   See .ipynb file for code

4. **Count the words and present the top 40 words used in the text. Include a word cloud in your report**

```
Top 40 Most Common Words:
said: 462
alice: 386
little: 128
one: 101
know: 86
like: 85
went: 83
would: 83
could: 77
thought: 74
queen: 68
time: 68
see: 66
king: 61
dont: 60
well: 60
began: 58
im: 57
mock: 56
turtle: 56
gryphon: 55
hatter: 55
quite: 55
think: 53
way: 52
much: 51
go: 50
say: 50
first: 49
thing: 49
head: 48
voice: 47
come: 46
get: 46
got: 45
looked: 45
never: 45
must: 44
rabbit: 43
round: 41
```

Annelise Thorn (anth6800)
Arwyn Lewis (arle8898)

5. **Analyze bigrams (two words in sequence) and show the top 20. Include a word cloud in your report**

```
Top 20 Most Common Bigrams:
('said', 'Alice'): 123
('Mock', 'Turtle'): 54
('March', 'Hare'): 31
('I', 'dont'): 30
('said', 'King'): 29
('thought', 'Alice'): 26
('White', 'Rabbit'): 22
('said', 'Hatter'): 22
('I', 'think'): 20
('said', 'Mock'): 20
('said', 'Caterpillar'): 18
('said', 'Gryphon'): 18
('I', 'shall'): 16
('I', 'wish'): 15
('said', 'Duchess'): 15
('I', 'wonder'): 14
('I', 'know'): 14
('I', 'cant'): 14
('said', 'Cat'): 14
('I', 'must'): 13
```

6. **Analyze trigrams (three words in sequence) and present the top 20. Include a word cloud in your report**

```
Top 20 Most Common Trigrams:
('said', 'Mock', 'Turtle'): 20
('said', 'March', 'Hare'): 9
('said', 'Alice', 'I'): 8
('I', 'dont', 'know'): 6
('little', 'golden', 'key'): 5
('I', 'wish', 'I'): 5
('I', 'think', 'I'): 5
('poor', 'little', 'thing'): 5
('white', 'kid', 'gloves'): 5
('I', 'beg', 'pardon'): 5
('March', 'Hare', 'said'): 5
('said', 'Alice', 'Why'): 5
('Mock', 'Turtle', 'said'): 5
('certainly', 'said', 'Alice'): 4
('might', 'well', 'say'): 4
('wish', 'I', 'hadnt'): 4
('I', 'dont', 'think'): 4
('said', 'Caterpillar', 'Well'): 4
('Of', 'course', 'said'): 4
('join', 'dance', 'Will'): 4
```

7. **Provide a written analysis of the data you found in points 2, 4, 5, and 6**

Distribution of Letters

The most common letters in *Alice's Adventures in Wonderland* are “e”, “a”, and “t” which is ironic because that spells out “eat” and Alice gets into Wonderland by eating a cookie that says “eat me.” Despite that amusing finding, the distribution of letters in *Alice's Adventures in Wonderland* was not surprising. The most common letters were vowels and the most frequently used consonants in English words. The rarer letters, like “z” and “x” were almost never used. The writing of *Alice's Adventures in Wonderland* reflects standard English writing.

Top 40 Words

The most common words in *Alice's Adventures in Wonderland* include “said”, “I”, and “Alice” which reflects how dialogue heavy the text is. Alice's thoughts and conversations drive almost the entire novel so it comes at no surprise that these are the three most common words.

Annelise Thorn (anth6800)
Arwyn Lewis (arle8898)

Additionally, other characters' names like “Queen”, “King”, and “Hatter” pop up multiple times which indicate how important these individuals are in Alice’s journey in Wonderland.

Top 20 Bigrams

The most common bigrams in *Alice’s Adventures in Wonderland* include “said Alice”, "Mock Turtle”, and “March Hare.” These bigrams further reiterate how dialogue heavy the text is and who Alice interacts with most. The phrases “I don’t,” “I think,” and “I must” reiterate Alice’s internal dialogue as she attempts to navigate the world of Wonderland. She is constantly reflecting on the bizarre characters and world around her.

Top 20 Trigrams

The most common trigrams in *Alice’s Adventures in Wonderland* are “said Mock Turtle,” “said March Hare,” and “said Alice I,” which once again reiterates how conversation heavy *Alice’s Adventures in Wonderland* is. The most common trigrams just expanded upon the most common bigrams which should come at no surprise since they come from the same text. The only major differences were “little golden key” and “poor little thing” which are three word phrases that are commonly used in the novel and are not associated directly with dialogue.

8. **Challenge: Produce an analysis of word frequency distribution by sentence structure (e.g., average words per sentence, common sentence starters). You must determine sentence boundaries and analyze trends in your writer's sentence structure. (This will not be graded)**

   See .ipynb file for code