**Project Summary**

In Project 8 I implemented a priority queue using heap, which is a data structure that is essentially a complete tree and arranges data entries in order. It always returns the data entry with the highest priority. Then I used the heap to find the most common words in the reddit files and develop the trends of the frequency of specific words from 2008 to 2015.

**Task Solutions**

PQHeap

I implemented a max heap, meaning that the value of the parent node is always higher than its children. In the base project I used an array-based implementation of heap. The add method will add a new element to the next available space in the array and then call the private method reheapUp to reshape the heap. reheapUp method recursively compares the new element with its parent and swap the two if the new element is greater than its parent.

The remove method returns the element with highest priority, which is the head of the heap, and replace it with the last element in the array. Then it calls the method reheapDown to swap the new-head downwards to a proper position.

FindCommonWords

I first added a method addToHeap() in the BSTMap class, which traverses the tree in order and add each element into a PQHeap specified in the parameter. I also added addToPQ method in WordCounter class which adds the elements in its BST field to a heap. FindCommonWords class has fields of WordCounter, a PQHeap, and a comparator based on the value of a KeyValuePair. The main method reads in the word count files and construct a BST accordingly. Then the method printCommonWords adds the elements in BST into a heap and prints out the N most-frequently mentioned words and their frequencies. Below are the top 10 common words in descending order from 2008 to 2015. It shows that the most common words don’t change a lot across the 8 years, with “the”, “to”, “a” always being the top three, which shows that they are the fundamental components of the English language.

FORM

FindTrends

The class uses a BST to read in multiple word count files and prints out the frequencies of specific words across the 8 years. I designed the main method to require inputs of the base file name, the begin of file number, the end of file number and a set of words that the user wishes to analyze. Then I used a nested for loop to loop through each word count file and each word and generates the frequencies. I used the FileWriter class to directly writes the results into a CSV file.

I first picked the names of American politicians to analyze the trends. Below is the output the CSV file and its line graph.

FORM

GRAPH

The trends correspond to the social/historical context of these words. The word “obama” has extremely high frequency in 2008 because of his presidential campaign and the subsequent inauguration. The frequency decreased sharply from 2009 because the election was over. The word “clinton” had a peak in 2008 also because of the political campaign of Hillary Clinton in 2008. In addition, the word “romney” was mentioned more in 2012 than the other years, potentially because Romney nominated by Republican Party for the 2012 election.

**Extensions**

**More trends analysis**

Countries

I then generated the frequencies of a list of countries (china japan germany russia isreal syria england), selected from “the most mentioned countries in New York Times”. It turns out the frequencies of China and Israel are much higher than the others, therefore, in order to reveal more details of the frequency curve of other countries, I graphed Israel and China separately.

GRAPHS

Similar to the trends of the name of American politicians, all the significant changes in the graph are linked to some historical events. The trends for European countries such as England and Germany are relatively stable. China was mostly mentioned in 2008 because of the Beijing Olympics. The mentions of israel was surprisingly high in 2008 and has decreased over the 8 years, so as the trend for Palestine, which indicates that the 2008 Gaza War was peak of the conflict between the two nations. Syria had a higher mention rate in 2013 because of its civil war and the first report from the US on Syrian usage of chemical weapons.

The mention of Russia increased drastically in 2014 due to the conflict between Russia and Ukraine and the Crimean crisis in 2014, which became an international concern especially on social medias. I generated the mentions of “ukraine” and “crimean” and plotted them in the same graph of Russia. It shows that theses three words have the same trend in year 2014.

GRAPH.

Electronic Companies

The word trends can also be indicators of the performance of companies in new innovations, market shares, etc. I generated the frequencies of popular electronic companies

lenono dell acer asus samsung

it appears that the mentions of Samsung have increased significantly over the 8 years, which corresponds to Samsung’s expansion in market share of smart phones from less than 3% in 2008 to 24% in 2015. The frequency of “dell” decreased over time while the market share of dell in personal laptops also decreased slightly from 17% to 13% from 2008 to 2015. The trend for sony has experienced some drastic upheavals. It became the topic of concerns in 2011, not due to a new release but because of its playstation network hacks.

Use an ArrayList to generate the top N words

I created the FindCommonWordsAL class, which is a child class of FindCommonWords and uses the sort method of an arraylist to find the words with highest frequency. It overwrites the printCommonWords method of its parent class. Instead of a adding the entries to a heap, it generates an arraylist using the getEntrySet method of BSTMap. Then I used the sort method of arraylist with the comparator for KVP reversed to get the entries in descending order.

PIC

I used both the heap and the ArrayList to print out the top 10 commons words for all the 8 word count files and the analyzing time for each year.

PIC GRAPH

The timing result shows that using an ArrayList is slightly faster than using the heap.

An attempt to build a node-based implementation of heap

Implementing a node-based heap is more complicated than building an array-based heap as it’s challenging to find the position to insert an element into the tree and also makes sure that the tree is always complete. With some research online, I found that the concept of binary representation for integers is very useful for finding the position to insert new element into the tree. Because each node in the tree can have maximum 2 children, the maximum number nodes at level n is 2n. As the tree of a heap is always complete, the binary representation (with base 2) of size can indicate the components of the tree. The algorithm is that a “0” in the binary representation means the absence of an exponentiation of 2 and thus the first child of a node, and “1” means the second child.

For example, if there are 4 nodes in the tree and we want to add a 5th one, the binary representation of 5 is 101. As there’s only one node at the root, the first digit can be ignored and starting from the second digit, 0 means traversing to the left and 1 means traversing to the right, then we arrive at the position where the new node should be put.

So I created the NodePQHeap class, which has a private sub-class Node that holds the KeyValuePairs. And Node object will have 3 Node fields that respectively point to its left child, right child and parent. The parent field will make it easier to swap the node with its parent in the reheaping process. Then I wrote the add method following the algorithm above, which ensures a complete tree after adding a new element, but I run out of time to finish the reheap methods which should reshape the tree. Below is a snippet of the add method.

**Conclusion**

This project helped me gain a better understanding of an array-based implementation of heaps and its applications. It was interesting to analyze the trends of word frequencies and compare them with their social and political context. It shows that the trend of public opinions are timely indicators of social/political/historical incidences.