Project Analysis

List Heuristics

Gitesh Agarwal

Mohamad Saleh

Jason Benckert

# Worst-Case Runtime

The worst case runtime for each heuristic, in terms of the total number of words is displayed below. For each heuristic, the worst case runtime would be lowest number of words that each one can process in a certain amount of time. The graph displays the total words per millisecond for each. The second graph displays the distinct words per millisecond for each heuristic. The binary search heuristic and the map words heuristic are the best, in terms of runtime, out of the six heuristics we implemented, for both the total words and the distinct words instances. The sort at end heuristic has the worst runtime in both instances, as it can processes less than one word per millisecond on average.

# Analysis

Below are two linear graphs presenting the runtime of each of the six heuristics, including “Base” and “Naïve” for comparison, with each of our five text files. The first graph displays a comparison between the total number of words in a document and the number of words per millisecond that each heuristic was able to process. The second graph displays a comparison between the total number of unique words in a document and the number of words per millisecond. In both graphs, a higher number of words per millisecond represents a higher processing speed, since the heuristic was able to process more words per millisecond than another.

In both graphs, we can see an interesting trend. The “Map Words” heuristic is by far the best heuristic, as it is able to process more words per millisecond than any other heuristic. Also, it can consistently process more words per millisecond as the number of words in a text file increases, and there is little change in the shape of the line between total words and unique words. The “Binary Search” heuristic is also one of the best, as it can also process a large number of words per millisecond. However, as both of the graphs show, it is inconsistent as the word count increases. According to our data analysis, this heuristic performs better in a setting where there are a small number of unique words. This makes sense ideally since fewer unique words means a smaller database to search through to update the word count. The worst heuristic is definitely the “Sort At End” heuristic, which has the lowest processing speed, independent of the word count. This is likely due to our implementation of this heuristic, which results in an n2 runtime, where n is the word count. Other than this heuristic, the other three unmentioned heuristics perform better than the “Naïve” heuristic, which is a baseline determination of the runtime. “Naïve” basically runs a “straight” search to find the next word in the list, which, according to our data, proves to be highly inefficient. It seems that whether it be to move the next word to the front of the list proceeding the search, moving it one step forward in the list, or dynamically sorting the list by word count, doing something to list is better – in terms of speed – than doing nothing. Each of these three methodologies provides a better runtime, and an increase in processing power, than the “Naïve” heuristic.

|  |  |
| --- | --- |
| The Most Common Words | |
| 5 Text Files | **English Language** |
| the | the |
| of | of |
| and | and |
| in | a |
| a | to |
| to | in |
| is | is |

The most common words in our five text files were “the”, “of”, and “and”, by order of

commonality. As you can see in the table to the right, the most common words in our text files were a match, almost word for word, to the most common words in the English language. The words displayed in the table are listed by order of commonality.

|  |
| --- |
| Unusually Common Words |
| computer |
| Germany |
| Rome |
| city |
| Buddhism |
| s |

We didn’t find many words in our text files that we determined to be unusually common. However, we found common words like “computer”, “germany”, “rome”, and others listed in the table to the left, that were expected as they were directly related to the topic of the text file. One of the most surprising words we found to be common was “s”. It was most likely attached to another word by an apostrophe, which was identified as a word separator in the program, “RunHeuristics”.