Spencer Ross
CS 351
HW 4
3/30/2020

CS 351 - Assignment 4 Report

File summary:

The only file included is my “bitmapindex.py” file. This file contains the two functions “create_index()” and “compress_index()”. There are other supplemental functions within this file. The functions included take in a file and sort the data inside if specified to do so and then convert the data into a bitmap and compress the data using the compression method specified. Currently the only compression method supported is WAH.

Analysis:

Comparing the bitmap indexes between sorted and unsorted data, the sorted bitmap index has a large number of 0’s and 1’s in a row in the columns where the unsorted is in a random order. Looking at the compressed files, the sorted compressed file is much shorter which implies that it compressed more data into words than the unsorted one. This is because when the data gets sorted, there are more runs of the same number lined up, which can be compressed into a count of runs rather than a bunch of literals. For word size of 32, the sorted compressed file has about 111 thousand characters in it. While the unsorted compressed file has over 1.6 million characters! The unsorted file has nearly 14 times the amount of data than the sorted. If we use a smaller word size of 8 the ratio for sorted to unsorted is about 27 thousand to 1.5 million.

The smaller the word size the better compression we get for literals and even for the sorted data, but up to a certain point. If we go too small, then there isn’t much benefit from having a bunch of tiny words when we could have had one larger word. But if we have a word size larger than the amount of data entries, then the compressed file will be a bunch of literals with padding, and that doesn’t save any data.

For word size 32

The sorted had:			
	not much	literals	//ran outta time to calculate
	a ton	fills	
The unsorted had:			
	50,356	literals	
	1,260	fills	