Write Up and Analysis

The BitMap Index folder for this program contains the following files:

1. README.MD: The file contains the overall information about the program. Data samples with domain constraints and execution information
2. Data folder: Contains animals.txt and animals_text.txt. The folder contains sorted and unsorted bit map for animals_text.txt and compressed 32 bit version of sorted and unsorted bitmap indexes.
3. Output folder: This folder contains the output files generated by the program.
    1. animals_bitMap_sorted.txt: Sorted bitmap of animals file.
    2. animals_bitMap.txt: Unsorted bitmap of animals file.
    3. animals.txt: The main data file unsorted.
    4. animals_sorted.txt: Sorted animals data file.
    5. animals_compressed_32.txt: Generated after compressing unsorted animals data file with WAH 32bit compression.
    6. animals_sorted_compressed_32.txt: Generated after compressing sorted animals data file with WAH 32bit compression.
    7. animals_compressed_64.txt: Generated after compressing unsorted animals data file with WAH 64bit compression.
    8. animals_sorted_compressed_64.txt: Generated after compressing sorted animals data file with WAH 64bit compression.
    9. bitIndex.py: The main python program used to generate the bitmap, sort file and do WAH compression.

Size of the 6 bitmap indexes:

- animals_bitMap.txt: **1.7 MB**
- animals_bitMap_sorted.txt: **1.7 MB**
- animals_compressed_32.txt: **1.6 MB**
- animals_compressed_sorted_32.txt: **115KB**
- animals_compressed_64.txt: **1.6MB**
- animals_compressed_sorted_64.txt: **224KB**

The animals_bitMap.txt and animals_bitMap_sorted.txt have the same size, because there is no difference in the actual data of the files, they both contains the same amount of data tuples hence the same file size.

Next in comparison is the animals_compressed_32.txt this file has a size of 1.6 MB which is 0.1 MB less than the original file. The reason for not too much reduction in the size is due to the unsorted data. It is inefficient to compress the unsorted data because the compression algorithm is unable to convert data into runs and has more literals.

The animals_compressed_sorted_32.txt has a file size of 115KB only the reason for such a big compression is due the the sorted data tuples. The columns have more continuous ones and zeros. The WAH algorithm efficiently translates these runs of ones or zeros into a 32 bit. Which can compress up-to 900 runs of zeros or ones of 31bit size. One 32 word can hold up-to 27900 zeros or ones which reduces the file size significantly.

The animals_compressed_64.txt this file has a size of 1.6 MB which is 0.1 MB less than the original file, similar to animals_compressed_32.txt. The reason for not too much reduction in the size is due to the unsorted data. It is inefficient to compress the unsorted data because the compression algorithm is unable to convert data into runs and has more literals. Even though 64 bit can store more data but if not sorted most of the data is stored as a literal.

The animals_compressed_sorted_64.txt has a file size of 225KB only the reason for such a big compression is due the the sorted data tuples. The columns have more continuous ones and zeros. The 64 bit word can store more than than the 32 bit word but 64 bit compression has a bigger size. The reason 64 bit is a little more inefficient than 32 bit word is because a 64 bit word have some wasted bits because we are unable to utilize all the bits. Also, storing small literals 64 bit words waste extra space in the padding.

The number of fill words and literals for each compressed file

- animals_unsorted_32bit_compression.txt fill count: **1261**
- animals_unsorted_32bit_compression.txt literal count: **50355**
- animals_unsorted_64bit_compression.txt fill count: **19**
- animals_unsorted_64bit_compression.txt literal count: **25389**
- animals_sorted_32bit_compression.txt fill count: **49856**
- animals_sorted_32bit_compression.txt literal count: **1760**
- animals_sorted_64bit_compression.txt fill count: **23626**
- animals_sorted_64bit_compression.txt literal count: **1782**