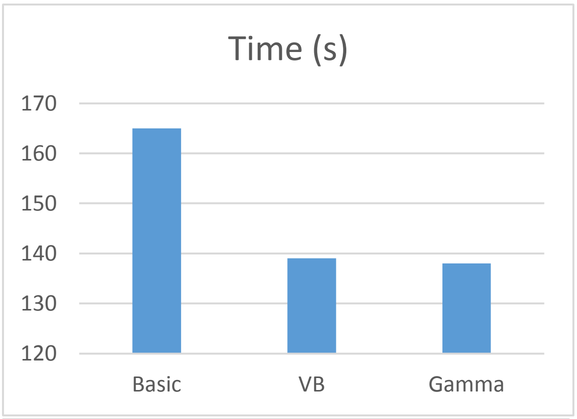
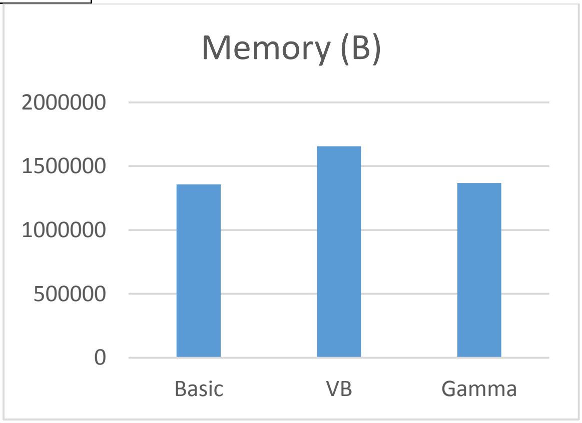
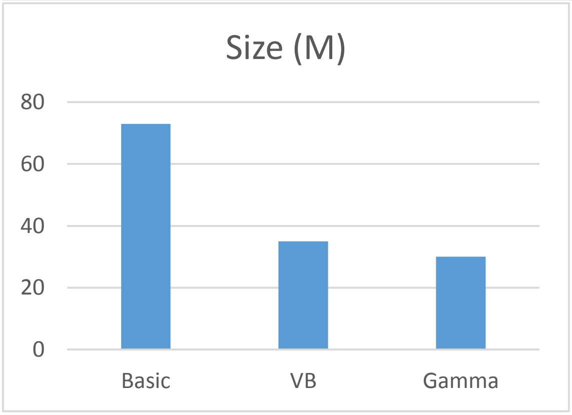
1. The code structure does not differ from the skeleton code structure.
   - Building index for each block
     - Parse all file and collect term id – doc id pairs
     - Sort the collected pairs based on the term id
     - Form posting lists for the sorted pairs
   - BSBI
     - Read a fixed chunk of the two blocks to be merged. A chink is a fixed number of posting lists, tunable based on memory availability.
     - Merge the chucks and write it into a merged block.
     - Repeat until all data from the block are read and merged.
     - Repeat until all blocks are merged.
   - Query Retrial:
     - Get a list of unique terms from the input query.
     - Check for invalid terms.
     - Retrieve the posting list (not the entire index) for each term and intersect them until all terms are processed.

Indexing: (Data collected from corn machines)

	Basic	VB	Gamma
Time (s)	165	139	138
Memory (B)	1358296	1657216	1368668
Size (M)	73	35	30



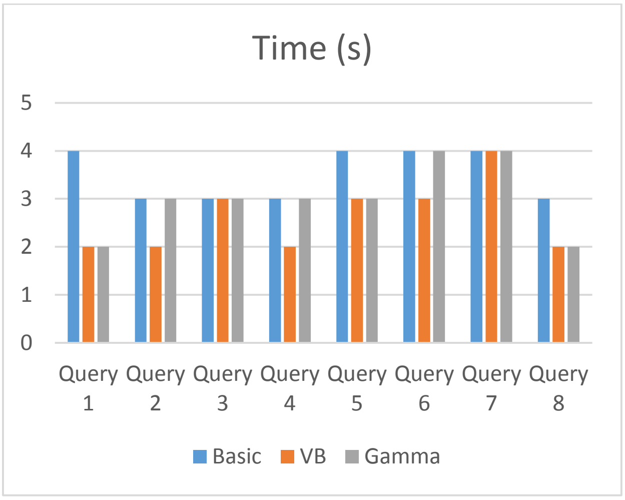


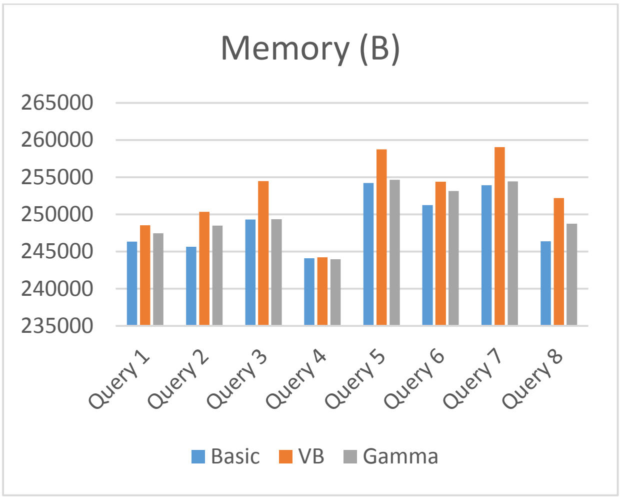


Query Retrival: (Data collected from corn machines)

Time (s)	Basic	VB	Gamma
Query 1	4	2	2
Query 2	3	2	3
Query 3	3	3	3
Query 4	3	2	3
Query 5	4	3	3
Query 6	4	3	4
Query 7	4	4	4
Query 8	3	2	2

Memory (B)	Basic	VB	Gamma
Query 1	246348	248512	247452
Query 2	245636	250356	248468
Query 3	249312	254456	249344
Query 4	244100	244236	243956
Query 5	254236	258728	254668
Query 6	251248	254400	253144
Query 7	253916	259048	254424
Query 8	246364	252200	248744





2. Having unequal sizes of blocks may decrease the performance of the BSBI algorithm because when the smaller block is fully read while merging two blocks, we simply read the rest of the data from larger block and write them to the merged block. This can be mitigated by sharding the corpus in blocks of equal size.

3. I define a chunk of an index file used in BSBI as a fixed number of posting lists instead of a fixed size of data. This would be bad if the size of a posting list grows too big to make the size of the chunk out of bounds t hold in memory. This would limit the scalability of the program to larger corpuses. This can be optimized to read a fixed size of data instead of a fixed number of posting lists to make the program more scalable.

4.

- While indexing a block, one can directly build the posting lists instead of multiple passes of collecting term id – doc id pairs, sorting them and then building the posting lists. This will consume less memory but may increase the processing time.
- One can create skip lists to increase the indexing and retrieval time. But this would increase the size of the index.
- One can also compress the dictionaries to save disk space. But this would introduce a one-time dictionary decompression cost per retrieval session.