Ashkan Zarkhah
610399196

- **Documentation**

  Here we describe our code line by line and summarize what we have done in our report section.

  - **Preprocessing and tokenizing**

    First of all, we needed a tokenizer for each of our documents. In our tokenizer, we first lowercased all of our document characters. Then because we had tokenized these documents before, we knew that we should replace all shortened forms (like I'm or didn't) with their original form (like I am or did not). After that, we tokenized our document with the nltk word tokenizer function and removed punctuation marks from it. Again because we had worked with these documents before, we know that some sentences are not split with a space character, and because of that some tokens are not split correctly, so we removed punctuation marks from the middle of our tokens and divided them into two tokens and also we removed dashes from combined words too (like 22-year-old). Also, some words were ended we a punctuation mark at the end of them and we cleared them too. After that, we filtered our 10 most repeated words (achieved from past works on the data) as stop words from our tokens, and at last, we put their stemmed form instead of them.

    After that we had our tokenizer, we went through all of our documents and added their tokens to our token's set and achieved our dictionary.


```
{'aid', 'radio', 'greater', 'an', 'lee', 'phoenix', 'success', 'what', 'jail', 'bay', 'theater', 'wrong', 'avail',
'fund', 'alreadi', 'scout', 'off', 'overpaid', '1992', 'death', 'show', 'slowli', 'later', 'hi', 'razor', 'head',
'can', 'contest', '1,000', '90', 'excitedli', 'counti', 'convers', 'they', 'allot', 'variabl', 'charg', 'clone', 's
mith', 'guy', 'than', 'brake', 'veterinarian', 'compani', 'nois', 'sent', 'seem', 'perfect', 'influenc', 'pick', 'b
urger', 'notifi', 'tax', 'depart', 'complic', 'citi', 'rent', 'resid', 'door', 'lane', 'read', 'are', 'inconsider',
'told', 'stain', '50x', 'money', 'dozen', 'vicki', 'foothil', 'suffer', 'blue', 'clear', 'tuner', 'doll', 'anoth',
'result', 'minor', 'toward', '9:00', 'refil', 'cours', 'front', 'nine', 'privat', 'heard', 'desper', 'massiv', 'oth
er', 'trash', 'groceri', 'homebuy', 'remark', 'event', 'rattl', 'higher', 'feder', 'materi', 'jerri', 'all', 'suc
h', 'whi', 'husband', 'extra', 'deliv', 'ok', 'grab', 'then', 'coupl', 'abus', 'thought', 'fight', 'caus', 'choos',
'gear', 'complaint', 'goal', 'afghan', 'home', 'day', 'retire', '.38', 'carson', 'long', '1993', 'same', 'canin',
'remain', 'creat', 'each', 'hill', 'station', 'arriv', 'watch', 'approv', 'surviv', 'tell', '500x', 'market', 'prop
erti', 'revolv', 'complain', 'offset', 'dom', 'street', 'budget', 'practic', 'joke', 'gave', 'got', 'estru', 'recei
v', 'rainstorm', 'bring', 'final', 'prison', 'least', 'milkplu', 'away', 'oper', 'visitor', 'cart', 'sara', 'contin
u', 'baldwin', 'at', 'out', 'whiskey', 'cigarett', 'also', 'play', 'way', 'canton', 'candi', 'erupt', 'chang', 'dro
ve', 'bookstor', 'do', 'him', 'babi', 'won', 'dollar', 'santa', 'everi', 'implant', 'spin', 'actual', 'appear', 'th
rew', 'turn', 'lot', 'but', 'yourself', 'provid', '1x', 'delight', 'teen', 'stray', '2.14', '510,000', 'exhibitor',
'yell', 'light', 'three', 'arcadia', 'barter', 'downtown', 'teenag', '10', 'warden', 'realtor', 'phone', 'item', 'w
ho', 'vendor', 'no', 'author', 'rais', 'medic', 'facil', 'tripl', 'breakfast', 'ruin', 'halt', 'allow', 'gangster',
'280', 'saturday', 'inferno', 'halloween', 'injur', 'compli', 'hous', 'nationwid', 'cost', '9', 'lowest', '87', 'mo
ve', 'john', 'angel', 'sinc', 'berserk', '20', 'blanket', 'readi', 'sunday', 'lake', 'milk', 'liquid', 'worm', 'hau
l', 'bigger', 'go', 'gallon', 'over', 'pretti', 'us', 'lucki', 'exchang', 'brand', '50th', 'four', 'necessari', 'tw
enti', 'middl', 'cell', 'refus', 'movi', 'shorter', 'pine', 'tri', 'complet', 'slice', 'leftov', 'empti', 'pasaden
a', 'negoti', 'glove', 'end', 'offici', 'job', 'cigar', 'morn', 'festiv', 'hawaiian', 'websit', 'reduc', 'player',
'friction', 'fresh', 'group', 'calib', 'thi', 'bullet', 'through', 'traffic', 'number', 'oil', 'reconsid', 'rate',
'occur', 'herman', '25', 'ban', 'start', 'found', 'talk', 'acr', 'pro', 'much', 'economi', 'johnson', 'part', 'basi
c', 'northvil', 'rain', 'autograph', 'guard', 'peopl', 'water', 'relat', 'spare', 'auto', 'sicker', 'their', 'twic
e', 'lose', 'stay', 'weekend', 'attitud', 'hour', 'april', 'sign', 'rental', '65', 'could', 'rage', 'room', 'roadsi
d', 'up', '12', 'tabl', 'smile', 'donna', 'slack', 'sixti', 'return', 'set', 'stand', 'nonfat', 'planner', 'never',
'just', 'follow', 'plate', 'walk', '110,000', 'full', 'geyser', '3', 'while', 'line', 'proprietor', 'newest', 'amer
ican', 'neighbor', 'larg', 'help', 'playground', 'good', 'one', 'luxuri', 'next', 'ha', 'femal', 'total', 'curren
t', 'sold', 'seashel', 'hope', 'made', 'suppos', 'brush', 'seeker', 'until', 'there', 'barney', 'bird', 'nanci', 'r
estor', 'gener', 'avenu', 'messag', '2.09', 'dirt', 'plu', 'coffe', 'it', 'princ', 'california', 'deal', 'invit',
'applianc', 'amount', 'hit', 'bottl', 'yet', 'exact', 'driver', 'store', 'person', 'tree', 'bummer', 'sometim', 'bo
th', 'remov', 'car', 'garbag', 'plot', 'alway', 'director', 'loan', 'cultur', 'think', 'limp', 'mild', 'attract',
'west', 'die', 'befor', 'nice', '5', 'post', 'thing', 'ice', 'eyesight', 'hot', 'kind', 'hydrant', 'earli', 'know',
'manag', 'dog', 'kick', 'holiday', 'friendli', 'scare', 'often', 'outdoor', 'time', 'bill', 'stationeri', 'mainte
n', 'wait', 'have', 'landfil', 'wash', 'lotteri', 'pass', 'carpent', 'loud', 'hamburg', 'buy', 'ammunit', 'arizon
a', 'certif', 'took', 'tire', 'sandwich', 'color', 'suv', 'old', 'profit', 'behind', 'suicid', 'emphysema', 'hire',
'studio', 'drop', '230,000', 'destroy', 'clockwork', 'cat', 'offer', 'ethnic', 'shave', 'suggest', 'local', 'atten
d', 'year', '99', 'her', 'shop', 'look', 'hand', 'either', 'creek', 'experi', 'brown', 'featur', 'mani', 'spew', 't
aken', 'on', 'futur', 'regular', 'fruit', 'averag', 'collis', 'state', 'bubbl', 'must', 'outsid', 'town', 'stolen',
```


Now that we had our dictionary, it was time to build a hash function to assign a term ID to each of our terms. To do so, we calculated our all term's representation in 256

base, module them to our dictionary size, and set them to be their term's ID and if their ID was assigned already (small probability) we increased the ID to find an empty ID.

- **Gamma Coding**

Before going to build our inverted indexes we needed to have functions to gamma encode, gamma decode, and merge our document ID gaps and our document frequencies. To do so first, we built a gamma encoder, in which we first added one to our input number (because the gamma code for 1 is 0, and zeros before the number will be omitted) and then calculated the length part and the offset part using Bit Manipulation and also with Bit Manipulation we made the gamma code and returned it. Then we built our decoder function, in which we started from the most valued bit and found numbers one by one from their length with Bit Manipulation, and for each number x we found, we added x-1 (because in building the gamma code we added one to the input) to our answers list. In the end, we developed our merge function, in which we shifted our first number to the left and added the second number to it.

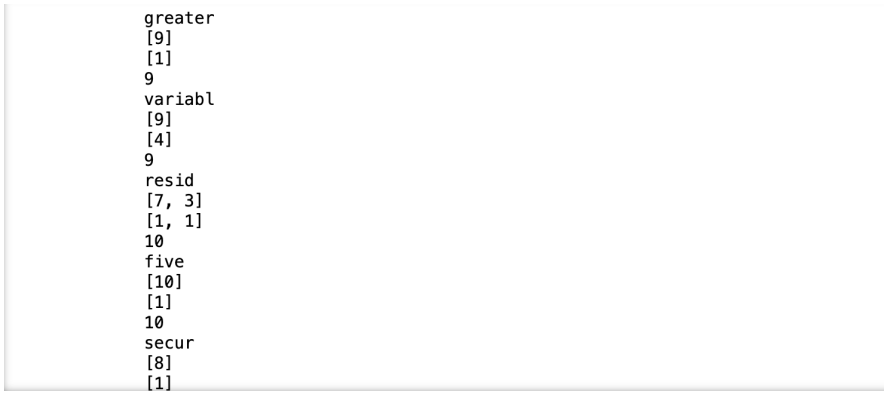
- **Inverted Index (BSBI)**

To simulate our disk loading and saving, we first defined a disk class to see it as an individual disk and only added our data to it and loaded them from it. In our disk class, we just saved a list of added data and returned them with their order of adding.

After that it was time to develop the BSBI algorithm and to do so, we decided to define every 5 document of our documents as a block and extract these information from our documents:

- **Inverted index:** here we saved gamma codes of our document ID gaps, except for the first document we saved its ID.
- **Document frequency:** here we saved gamma codes of each document ID's term frequency.
- **Last document:** here we saved last term's document ID to calculate the next gap for our inverted index

To do so, we first sorted all of our document's terms and counted each term's frequency and then merged its ID to the inverted index, its frequency to the document frequency and we replaced the new document's ID in the term's last document. Every time we achieved 5 documents, we saved them to our disk and cleared our inverted index, document frequency, and last document, and went for the next block.
Before going to merge our blocks, we checked each block's inverted index to see if it was correct and then we moved forward to merge the blocks.



```
greater
[9]
[1]
9
variabl
[9]
[4]
9
resid
[7, 3]
[1, 1]
10
five
[10]
[1]
10
secur
[8]
[1]
```

After building all blocks (here we have 15 documents and 3 blocks) it was time to merge them, and to do so, we looped through all blocks and merged them to our base variables. To merge our document frequency and last document, our job was simple but for inverted index document ID gaps, the first element of block inverted indexes were document IDs not their gap from the last block, so we calculated the gap from our last document and changed the gamma coded inverted index with Bit Manipulation and then we merged it to our base inverted index.
At last, we checked our inverted indexes and document frequencies again and saw it was completely correct.

```
greater
[9]
[1]
9
onto
[11]
[1]
11
line
[3, 12]
[4, 1]
15
playground
[12]
[1]
12
know
[2, 3, 7]
[1, 1, 2]
```

- **Report**

  Here we summarize what we have done on our code.
  - **Document Preprocessing**
    - Lowercased all document
    - Replaced all shortened forms
    - Tokenized with nltk tokenizer
    - Removed punctuation marks from all tokens
    - Replaced all tokens with their stemmed forms
  - **Inverted Index**
    - Implemented a disk class to simulate disk
    - Blocked every 5 documents together
    - Calculated inverted index's document ID's gaps and merged their gamma codes
    - Calculated each term's document ID's frequency and merged their gamma codes
    - Saved each block to the disk class
  - **Gamma Coding for Document ID Gaps**
    - Implemented Gamma encoder using Bit Manipulation
    - Implemented Gamma decoder using Bit Manipulation
    - Implemented Gamma codes merger using Bit Manipulation

- **Index Block Merging**
  - Calculated the first document ID's gap of each term's inverted index from the last block's inverted index
  - Calculated the new gamma-coded inverted index
  - Merged new inverted indexes to base variables
  - Merged new document frequencies to base variables
  - Updated last document base variables from new last documents
- **Implement Bit Manipulation for Gamma Coding**
  - As described before, we saved our codes as numbers and used Bit Manipulation to encode, decode, and merge our codes.
- **Key findings:**
  - NLTK stemmer doesn't exactly replace each term with its noun form but it replaces it with a unique prefix of its stem form to merge all term's form. (for example "groceri" for both grocery and groceries)
  - We don't need to have all the tokens together to build our inverted index or dictionary and it can be built partially.
  - To decrease our memory usage we must increase our time usage.
  - Only inverted indexes of blocks are not enough to merge their inverted indexes and we need the last document's ID to calculate the first gap.
- **Challenges faced:**
  - Because we needed to tokenize documents individually we needed to implement a function to do so and without all tokens, finding special cases was hard and we were forced to use our last exercise data.
  - Because we needed our dictionary to assign term IDs, we were forced to loop over documents twice.
  - Because the gamma code for 0 is not defined, we were forced to index our documents from one
  - Because the gamma code for 1 is 0 and zeros before numbers are not calculated, we were forced to return code of x+1 for x and decode x - 1 for x's code
  - For merging inverted indexes, we needed to change a part of our gamma codes, and implementing this part was so hard and full of bugs

- **References**
  - **os library (for including documents)**
  - **NLTK library (for tokenizing and stemming)**