NETWORK DATA STREAMING

VASU NEGI
UFID: 8495-3933

PROJECT-2

In order to run these programs on *nix based/MAC system, follow the instructions below:

******Please run the program in Python 3.7 Environment, as there are some functions in these programs which only work in Python 3******

* Go to project directory in terminal
* Type the command in the following way:
python file_name parameter1 parameter2 parameter3 parameter4
(number of parameters depend on the algorithm that you will run)

Bloom Filter:

Run: python Bloomfilter.py 1000 10000 7

Where the parameters are:
number_of_elements, number_of_bits, number_of_hashes

1. **class BloomFilter:**
   a. def __init__(self, number_of_elements, number_of_bits, number_of_hashes):
      This will initialize the variables as well as check for any wrong input and raise appropriate Error
   b. def check_input(*input_string):
      This is a static function that validates the input parameters.
   c. def lookup(self,key):
      lookup function to check where the key is in the filter
   d. def encode(self,key):
      This function will encode the key into the filter
   e. def generate_random_numbers(self):
      This function generate random numbers of a given range.
   f. def get_hash_functions(self,number):
      This function generates hash functions with the following function:

      (f XOR r), where XOR is a bitwise operator with 0 XOR 0 = 0, 1 XOR 0 = 1, 0 XOR 1 = 1, and 1 XOR 1 = 0.

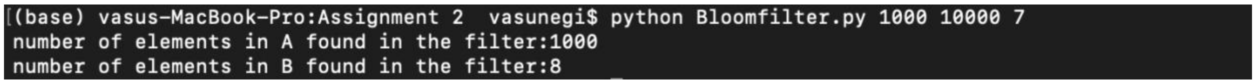
      where r is a random number in the self.random_numbers and f is an element.
   g. def get_elements(self):
      This function generate random elements.
   h. def execute1(self):

This function will generate elements (denoted as set A) randomly, encode them in the filter, look up them in the filter.

i. def execute2(self):

    This function will generate another set of elements randomly (denoted as set B) and look up them in the filter.

Following is the output result for:
number_of_elements = 1000, number_of_bits = 10000, number_of_hashes = 7



```
[(base) vasus-MacBook-Pro:Assignment 2  vasunegi$ python Bloomfilter.py 1000 10000 7
number of elements in A found in the filter:1000
number of elements in B found in the filter:8
```

```
(base) vasus-MacBook-Pro:Assignment 2  vasunegi$ python Bloomfilter.py 1000 10000 7
number of elements in A found in the filter:1000
number of elements in B found in the filter:8
```

Counting Bloom Filter:

Run: python CountingBloomFilter.py 1000 500 500 10000 7

Where the parameters are:
number_of_elements, number_of_elements_to_remove, number_of_elements_to_be_added, number_of_counters,number_of_hashes

2. **class Counting Bloom :**
    a. def __init__(self, number_of_elements, number_of_elements_to_remove, number_of_elements_to_be_added, number_of_counters,number_of_hashes):

        This will initialize the variables as well as check for any wrong input and raise appropriate Error
    b. def check_input(*input_string):

        This is a static function that validates the input parameters.
    c. def lookup(self,key):

        lookup function to check where the key is in the filter
    d. def encode(self,key):

        This function will encode the key into the filter.
    e. def remove(self,key):

        This function will remove the key from the filter.
    f. def generate_random_numbers(self):

        This function generate random numbers of a given range.
    g. def get_hash_functions(self,number):

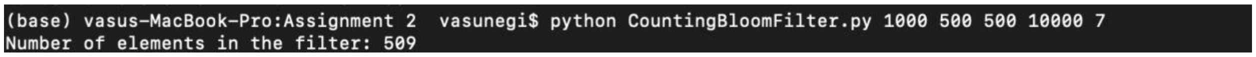
        This function generates hash functions with the following function:

        (f XOR r), where XOR is a bitwise operator with 0 XOR 0 = 0, 1 XOR 0 = 1, 0 XOR 1 = 1, and 1 XOR 1 = 0.

        where r is a random number in the self.random_numbers and f is an element.

h. def get_elements(self,number):
    This function generates random elements.
i. def execute(self):
    This function 1,000 elements (denoted as set A) randomly, encode them in the filter, look up them in the filter, and generate another 1.000 elements randomly (denoted as set B) and look up them in the filter.

Following is the output result for:
number_of_elements = 1000, number_of_elements_to_remove = 500, number_of_elements_to_be_added = 500 , number_of_counters = 10000,number_of_hashes = 7



```
(base) vasus-MacBook-Pro:Assignment 2  vasunegi$ python CountingBloomFilter.py 1000 500 500 10000 7
Number of elements in the filter: 509
```

(base) vasus-MacBook-Pro:Assignment 2 vasunegi$ python CountingBloomFilter.py 1000 500 500 10000 7
Number of elements in the filter: 509

Coded Bloom Filter:

Run: python CodedBloomFilter.py 7 1000 3 30000 7

Where the parameters are:
number_of_sets, number_of_elements, number_of_filters, number_of_bits, number_of_hashes

class CodedBloomFilter:

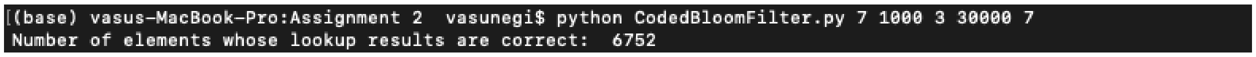
a. def __init__(self, number_of_sets, number_of_elements, number_of_filters, number_of_bits, number_of_hashes):
    This will initialize the variables as well as check for any wrong input and raise appropriate Error
b. def check_input(*input_string):
    This is a static function that validates the input parameters.
c. def lookup(self,key):
    lookup function to check where the key is in the filter
d. def encode(self,key):
    This function will encode the key into the filter
e. def calculate_set_code(self):
    This function will calculate the list of codes for each set.
f. def generate_random_numbers(self):
    This function generate random numbers of a given range.
g. def get_hash_functions(self,number):
    This function generates hash functions with the following function:

    (f XOR r), where XOR is a bitwise operator with 0 XOR 0 = 0, 1 XOR 0 = 1, 0 XOR 1 = 1, and 1 XOR 1 = 0.

    where r is a random number in the self.random_numbers and f is an element.

h. def get_elements(self):
    This function generate random elements.
i. def execute(self):
   This function generate 7 sets of 1000 elements each, their codes are 001 through 111 respectively, encode all sets in 3 filters according to the algorithm, and perform lookup on all elements in the 7 sets.

Following is the output result for:
number_of_sets = 7, number_of_elements = 1000, number_of_filters = 3, number_of_bits = 30000, number_of_hashes = 7



```
[(base) vasus-MacBook-Pro:Assignment 2  vasunegi$ python CodedBloomFilter.py 7 1000 3 30000 7
 Number of elements whose lookup results are correct:  6752
```

(base) vasus-MacBook-Pro:Assignment 2 vasunegi$ python CodedBloomFilter.py 7 1000 3 30000 7
Number of elements whose lookup results are correct: 6752