Coursework 3 Streaming Algorithm

Task1： DGIM

DGIM is an efficient algorithm in processing large streams. When it's infeasible to store the flowing binary stream, DGIM can estimate the number of 1-bits in the window. In this coding, you're given the stream_data_dgim.txt (binary stream), and you need to implement the DGIM algorithm to count the number of 1-bits. Write code below.

1. Set the window size to 1000, and count the number of 1-bits in the current window.

In [1]:

```
```

2. With the window size 1000, count the number of 1-bits in the last 500 and 200 bits of the bitstream.

In [2]:

```
```

3. Write a function that accurately counts the number of 1-bits in the current window. Caculate the accuracy of your own DGIM algorithm and compare the running time difference.

In [3]:

```
```

Task2: Bloom Filter

A Bloom filter is a space-efficient probabilistic data structure. Here the task is to implement a bloom filter by yourself.

Data loading:

From the NLTK (Natural Language ToolKit) library, we import a large list of English dictionary words, commonly used by the very first spell-checking programs in Unix-like operating systems.

In [4]:

```
import nltk
from nltk.corpus import words
nltk.download('words')
word_list = words.words()
```

Then we load another dataset from the NLTK Corpora collection: movie_reviews.

The movie reviews are categorized between positive and negative, so we construct a list of words (usually called bag of words) for each category.

In [5]:

```
from nltk.corpus import movie_reviews
nltk.download('movie_reviews')

neg_reviews = []
pos_reviews = []

for fileid in movie_reviews.fileids('neg'):
    neg_reviews.extend(movie_reviews.words(fileid))
for fileid in movie_reviews.fileids('pos'):
    pos_reviews.extend(movie_reviews.words(fileid))
```

Here we get a data stream (word_list) and 2 query lists (neg_reviews and pos_reviews).

1. Write a function that accurately determines whether each word in neg_reviews and pos_reviews belongs to word_list.

In [6]:

```

```

2. Implement the bloom filter by yourself and add all words in word_list in your bloom filter. Compare the running time difference between linear search on a list and multiple hash computations in a Bloom filter.

In [7]:

```

```

3. Use different bit array length ‘m’ and number of hash functions ‘k’ to implement the bloom filter algorithm. Then compare the impact of different m and k on the false positive rate.

In [8]:

```

```

Task3: Statistics Estimation

Here we use the query stream (neg_reviews) from task 2 to estimate 1) the number of distinct words appeared, and 2)the surprise number of the stream.

1. Write a function that accurately counts the occurrence times of each word in neg_reviews.

In [9]:

2. Implement the Flajolet-Martin alg. to estimate the number of distinct words occurred. Try multiple hash functions to improve the estimate.

In [10]:

3.Estimate the surpise number with limited memory to store words.

In [ ]: