**Міністерство освіти і науки України**

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Теплоенергетичний факультет**

**Кафедра автоматизації проектування  
енергетичних процесів і систем**

***Звіт***

***з лабораторної роботи №5***

***"*** ***Flajolet-Martin"***

***з дисципліни***

***"Обробка потокової інформації"***

**Виконав:**

Студент групи ТІ-01мп

Стеценко Д. О.

**Перевірив:**

Д.ф.-м.н.

Матичин І.І.

**м. Київ – 2021 р.**

**Завдання**:

1. Згенерувати нескінчений потік чисел, що можуть повторюватися.
2. Реалізувати алгоритм Флажоле-Мартена для оцінки кількості різних елементів потоку.
3. Порівняти результат, отриманий на попередньому кроці, з реальною кількістю різних елементів потоку.

**Передумови**:

Для реалізації конс’юмера даних була використана платформа **Node.js**, бібліотека **Rx.js** для створення і генерації потоку даних.

**Хід виконання**:

Алгоритм Фладжоле-Мартіна дозволяє приблизно оцінити кількість унікальних елементів.

Під час виконання роботи було визначено що алгоритм добре працює для кількості унікальних елементів > 1000.

Код основної програми зображений нижче та за посиланням:

<https://github.com/oddestdan/kafka-streams/tree/main/task-5>

const { interval, Subscription } = require('rxjs');

const { map } = require('rxjs/operators');

const { handleErrors } = require('./error-handling');

const {

  hash\_func,

  find\_rightmost\_bit,

  find\_rightmost\_unset\_bit,

} = require('./helpers');

const { printPadded } = require('../shared/utils');

let vector = 0;

let currentApprox = 0;

let uniqueValues = new Set();

const stream = interval(25).pipe(map(() => Math.round(Math.random() \* 10000)));

const subscription$ = new Subscription();

subscription$.add(

  stream.subscribe((*number*) => {

    uniqueValues.add(number);

    // compute the hash value bounded by (2 \*\* L)

    // this hash value will ensure uniform distribution

    // of elements of the stream in range [0, 2 \*\* L)

    const hash = hash\_func(number);

    // find rightmost bit and set it in the bit vector

    const rightmost\_bit = find\_rightmost\_bit(hash);

    vector = vector | rightmost\_bit;

    // find the rightmost unset bit in the bit vector that

    // suggests that the probability being 0

    // and return the approximate cardinality

    currentApprox = 2 \*\* find\_rightmost\_unset\_bit(vector);

    console.log(

      `| Value: ${printPadded(number)} | FM: ${printPadded(

        currentApprox

      )} | True: ${printPadded(uniqueValues.size)} |`

    );

  })

);

process.on('SIGINT', function () {

  console.dir(

    `Unique elements: Approx count is ${currentApprox},

    true count is ${uniqueValues.size}`

  );

  subscription$.unsubscribe();

});

handleErrors();

Код допоміжних функцій (хешування, знаходження необхідний бітів, тощо:

// size of the bit vector

const L = 64;

function dec2bin(*dec*) {

  return (dec >>> 0).toString(2);

}

function hash\_func(*x*) {

  return (3 \* x + 5) % 2 \*\* L;

}

function find\_rightmost\_bit(*n*) {

  if (n === 0) {

    return n;

  }

  const power = dec2bin(n).split('').reverse().indexOf('1');

  return 2 \*\* power;

}

function find\_rightmost\_unset\_bit(*n*) {

  if (n === 0) {

    return n;

  }

  const idx = dec2bin(n).split('').reverse().indexOf('0');

  const power = idx === -1 ? dec2bin(n).split('').length : idx;

  return power;

}

module.exports = {

  hash\_func,

  find\_rightmost\_bit,

  find\_rightmost\_unset\_bit,

};

Результат виконання програми:

$ node index

| Value: 4434 | FM: 2 | True: 1 |

| Value: 589 | FM: 2 | True: 2 |

| Value: 7519 | FM: 8 | True: 3 |

| Value: 5705 | FM: 8 | True: 4 |

| Value: 2285 | FM: 8 | True: 5 |

...

| Value: 5188 | FM: 8 | True: 21 |

| Value: 9016 | FM: 8 | True: 22 |

| Value: 7067 | FM: 8 | True: 23 |

| Value: 4352 | FM: 8 | True: 24 |

| Value: 2785 | FM: 16 | True: 25 |

| Value: 7730 | FM: 16 | True: 26 |

| Value: 7215 | FM: 16 | True: 27 |

...

| Value: 2964 | FM: 16 | True: 41 |

| Value: 7269 | FM: 16 | True: 42 |

| Value: 7108 | FM: 16 | True: 43 |

| Value: 9465 | FM: 64 | True: 44 |

| Value: 7969 | FM: 64 | True: 45 |

| Value: 2161 | FM: 64 | True: 46 |

...

| Value: 9446 | FM: 64 | True: 66 |

| Value: 4835 | FM: 64 | True: 67 |

| Value: 4122 | FM: 64 | True: 68 |

'Unique elements: Approx count is 64, true count is 68'