**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 №4***

***"*** ***Fading Window"***

***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 100 нескінчених потоків випадкових 0 та 1.
2. Написати програму, що в кожен момент часу визначає потік з найбільшою кількістю нещодавніх одиниць, використовуючи зважену суму (кількість одиниць у затухаючому вікні).
3. Зберігати в пам’яті зважену суму потоку тільки в тому випадку, якщо вона перевищує певне порогове значення (1/2).

**Передумови**:

Для реалізації конс’юмера даних була використана платформа **Node.js**, бібліотека **Kafka.js**, бібліотека **Rx.js** для створення і генерації потоку даних. Припустимо, що Zookeeper і брокер Kafka вже запущені.

**Хід виконання**:

1. Спочатку задаємо константу С і створюємо зміну для зберігання зважених сум. Потім створюємо потоки випадкових 0 і 1. Коли отримуємо наступне значення з потоку виконуємо наступні дії:
2. Якщо сума потоку існує в пам’яті: підраховуємо нову суму, множенням збереженого значення на 1 - с і додаванням останнього значення з потоку. Потім відбувається перевірка значення: якщо сума менше ½, видаляємо її з пам’яті.
3. Якщо суми немає в пам’яті: при отриманні одиниці ініціалізуємо нову суму, при отриманні 0, ігноруємо значення.
4. Також приблизно кожну секунду відбувається пошук потоку з найбільшим значенням і вивід цієї інформації на екран.

Код потоків даних координат зображений нижче та за посиланням:

<https://github.com/oddestdan/kafka-streams/tree/main/task-4>

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

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

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

let savedStreams = {};

const subscription$ = new Subscription();

const arr = new Array(100).fill(0);

const streams = arr.map((*\_*, *idx*) =>

  interval(50).pipe(map(() => Number(Math.random() > 0.95)))

);

const C = 0.05;

for (let i = 0; i < streams.length; i++) {

  streams[i].subscribe((*value*) => {

    if (i in savedStreams) {

      savedStreams[i] = savedStreams[i] \* (1 - C) + value;

      if (savedStreams[i] < 0.5) {

        delete savedStreams[i];

      }

    } else {

      if (value) {

        savedStreams[i] = 1;

      }

    }

  });

}

subscription$.add(

  interval(1000).subscribe(() => {

    let maxval = 0;

    let maxidx = null;

    const keys = Object.keys(savedStreams);

    for (const key of keys) {

      if (savedStreams[key] > maxval) {

        maxval = savedStreams[key];

        maxidx = key;

      }

    }

    console.log(`highest stream number is ${maxidx} with value ${maxval}`);

  })

);

process.on('SIGINT', function () {

  console.dir(savedStreams);

  subscription$.unsubscribe();

});

handleErrors();

Результат виконання програми:

$ node index

highest stream number is 28 with value 2.87276135790699

highest stream number is 17 with value 3.1821877415768918

highest stream number is 33 with value 5.282736908536293

highest stream number is 33 with value 5.420772087850865

highest stream number is 88 with value 5.980943219176952

highest stream number is 0 with value 6.937215526196748

{

'0': 6.663861554974646,

'1': 4.890376555828457,

'2': 1.9554814985771285,

'3': 3.575097582627733,

'4': 3.3977095108264055,

'5': 2.8364123652448328,

'6': 3.1501450306359855,

'7': 1.721033195705155,

'8': 2.104307050533861,

'9': 0.8097278682212583,

'10': 3.7950915870869215,

'12': 3.1681173396613045,

'13': 4.582855343676272,

'14': 4.700365631166471,

'15': 2.3618129809953587,

'16': 4.344096352783866,

'17': 2.7325980518194126,

'19': 4.087282208768468,

'20': 3.0987323477739843,

'21': 0.7034476949995692,

'22': 1.7843228455453015,

'23': 1.246900108225314,

'24': 3.163225552813328,

'25': 3.1401574665734393,

'26': 4.346906485832088,

'27': 2.505897947896462,

'28': 2.856739421173548,

'29': 3.249511420617313,

'30': 3.796273861562355,

'31': 4.071892799470467,

'32': 2.6064339386882183,

'33': 5.548394765214681,

'34': 4.665580865167629,

'35': 2.7773931957649283,

'36': 4.373909267703991,

'37': 3.845131311427262,

'38': 2.8313181588037275,

'39': 1.7639845040550257,

'40': 3.5121474920482476,

'41': 4.358974047297994,

'42': 3.387237342937621,

'43': 4.399847678038018,

'44': 2.28350455664786,

'45': 2.514789876916347,

'46': 4.071840598199641,

'47': 5.1721399326897455,

'48': 1.2420690747699017,

'49': 2.664695115090629,

'50': 2.0748476143687578,

'51': 5.515640364402922,

'52': 2.067334199743829,

'53': 0.9254271532878537,

'54': 2.1781904252199973,

'55': 2.363969463187318,

'56': 5.557100707260657,

'57': 5.918189591650678,

'58': 2.3805635291748826,

'59': 4.45808289211634,

'60': 1.2246743231559634,

'61': 3.6265649789947654,

'62': 3.9370887654038116,

'63': 2.816539542984519,

'64': 5.0937067470681,

'65': 3.231784587380799,

'66': 1.5703746774527583,

'67': 3.125386957628552,

'68': 1.8580947816600393,

'69': 1.9553868019182075,

'70': 3.583139149286764,

'71': 1.7396858713192274,

'72': 1.996031359275037,

'73': 0.99,

'74': 1.8899951885923938,

'75': 3.739359299470834,

'76': 1.8754382542587165,

'77': 2.115350208986153,

'78': 2.410381482690825,

'79': 5.662053592778934,

'80': 0.8016305895390458,

'81': 1.9370037554131116,

'82': 2.1316253736879225,

'83': 1.7644404296500926,

'84': 2.1406710532871727,

'85': 4.1592524093131,

'86': 2.820053789609385,

'87': 7.049657330798721,

'88': 4.842936402816544,

'89': 5.03566486531449,

'90': 3.0473928004985993,

'91': 5.267478017850035,

'92': 2.775026492272558,

'93': 1.1801224205842706,

'94': 4.025141961755995,

'95': 4.73853448791139,

'96': 1.7822630602526188,

'97': 1.5312945980068624,

'98': 1.81096060454735,

'99': 2.265248575813517

}