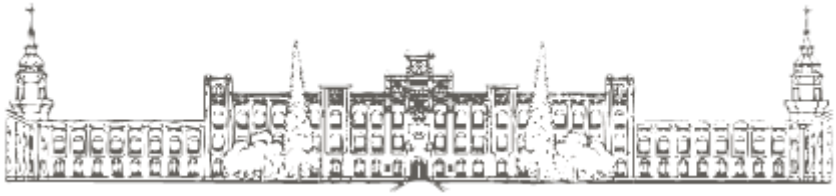


Міністерство освіти і науки України

Національний технічний університет України «КПІ

імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

ЗВІТ

лабораторної роботи №4

з дисципліни «Моделювання систем»

Перевірила:

Дифучина О. Ю.

Виконав:

Студент Гр. ІП-01

Пашковський Є. С.

Київ 2023

Завдання

1. Розробити модель масового обслуговування, яка складається з N систем масового обслуговування. Число N є параметром моделі. Кількість подій в моделі оцінюється числом N+1. **20 балів.**
2. Виконати експериментальну оцінку складності алгоритму імітації мережі масового обслуговування. Для цього виконайте серію експериментів, в якій спостерігається збільшення часу обчислення алгоритму імітації при збільшенні кількості подій в моделі. **40 балів.**
3. Виконати теоретичну оцінку складності побудованого алгоритму імітації. **30 балів.**
4. Повторіть експеримент при зміні структури мережі масового обслуговування. **10 балів.**

Хід роботи

Код для виконання завдання:

```
// index.ts
import Create from './Create';
import Model from './Model';
import ProcessingSystem from './ProcessingSystem';

const createModelOneChain = (n: number) => {
  const create = new Create(1);

  const processingSystems: ProcessingSystem[] = [];

  for (let i = 0; i < n; i++) {
    const ps = new ProcessingSystem(1, 1);

    if (i > 0) processingSystems[i - 1].setNextProcessingSystem(ps);

    processingSystems.push(ps);
  }

  create.setNextElements([processingSystems[0].getQueue()]);

  return new Model([
    create,
    ...processingSystems
      .map((ps) => ps.getElements())
      .reduce((acc, el) => [...acc, ...el], []),
  ]);
};

const createModelTwoChains = (n: number) => {
  const create = new Create(1);

  const processingSystems: ProcessingSystem[] = [];

  for (let i = 0; i < n; i++) {
    const ps = new ProcessingSystem(1, 1);

    if (i > 1) processingSystems[i - 2].setNextProcessingSystem(ps);

    processingSystems.push(ps);
  }

  create.setNextElements([
    processingSystems[0].getQueue(),
    processingSystems[1].getQueue(),
  ]);

  return new Model([
```

```
    create,
    ...processingSystems
      .map((ps) => ps.getElements())
      .reduce((acc, el) => [...acc, ...el], []),
  ]);
};

const results: { [N: number]: number } = {};

const k = 5;

for (let N = 10; N <= 300; N += 10) {
  const Nresults: number[] = [];

  for (let i = 0; i < k; i++) {
    const model = createModelOneChain(N);

    const startTime = Date.now();
    model.simulate(10000);
    const endTime = Date.now();
    Nresults.push(endTime - startTime);
  }

  results[N] = Nresults.reduce((acc, el) => acc + el, 0) / k;
}

console.table(results);
```

```
import Element from './Element';
import ModelObject from './ModelObject';
import Process from './Process';
import Queue from './Queue';

export default class ProcessingSystem<T extends ModelObject = ModelObject> {
  private queue: Queue<T>;
  private processes: Process<T>[];

  constructor(
    meanProcessingTime: number,
    processesCount = 1,
    queueLength = Infinity
  ) {
    const queue = new Queue<T>('ps queue', queueLength);

    const processes: Process<T>[] = [];

    for (let i = 0; i < processesCount; i++) {
      processes.push(new Process<T>('ps process' + i, meanProcessingTime));
    }
```

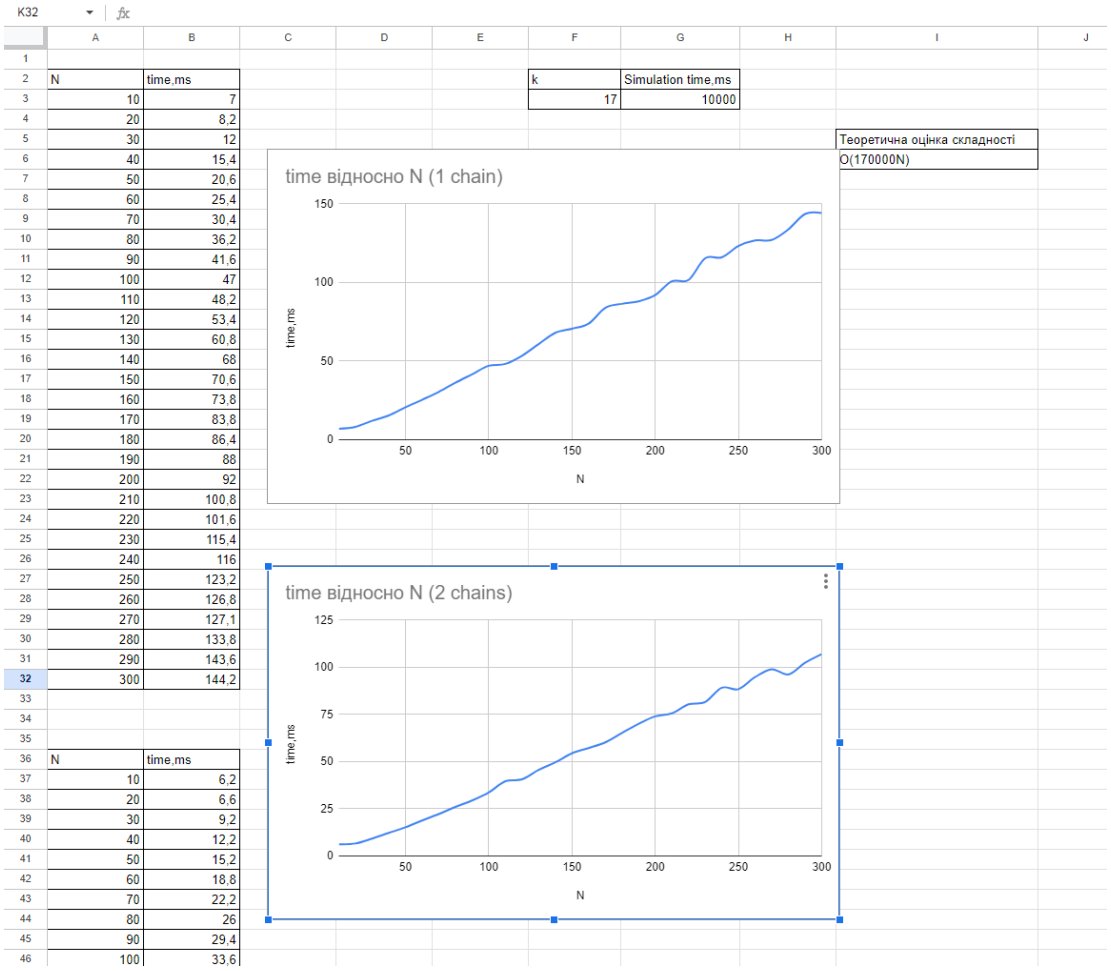
```
    queue.setNextElements(processes);

    this.queue = queue;
    this.processes = processes;
  }

  public getElements() {
    return [...this.processes, this.queue];
  }

  public getQueue() {
    return this.queue;
  }

  public setNextElement(element: Element<T>) {
    for (const process of this.processes) {
      process.setNextElements([element]);
    }
  }

  public setNextProcessingSystem(processingSystem: ProcessingSystem<T>) {
    const nextQueue = processingSystem.getQueue();

    this.setNextElement(nextQueue);
  }
}
```

Результати роботи коду:

```
PS E:\Projects\system-modelling-labs> npx ts-node .\lab4\src\
```

(index)	Values
10	7
20	8.2
30	12
40	16
50	20.4
60	26.2
70	31.6
80	34.4
90	39.6
100	47.2
110	52.4
120	55.4
130	61.4
140	69
150	73.4
160	79.4
170	77.6
180	86
190	90.6
200	95.4
210	99.8
220	99.6
230	122.6
240	114
250	118.8
260	125.6
270	129.8
280	151.6
290	138.6
300	163

Результати у вигляді таблиць та графіків:



Висновки: під час виконання цього завдання було протестовано швидкодію моделі масового обслуговування з N СМО при збільшенні цього параметра та при зміні структури мережі.