Program.cs

using System;

namespace SAaCSim

{

class Program

{

static void Main(string[] args)

{

ConsoleKeyInfo keyInfo;

do

{

double p = 0;

double n1 = 0;

double n2 = 0;

Console.Write("Enter p: ");

p = Double.Parse(Console.ReadLine());

Console.Write("Enter n1: ");

n1 = Double.Parse(Console.ReadLine());

Console.Write("Enter n2: ");

n2 = Double.Parse(Console.ReadLine());

Task18 task18 = new Task18(ρ: p, π1: n1, π2: n2);

task18.Execute();

Console.WriteLine("Press Escape to exit.");

keyInfo = Console.ReadKey();

Console.Clear();

} while (keyInfo.Key != ConsoleKey.Escape);

}

}

}

Task20.cs

using System;

using System.Collections.Generic;

using System.Linq;

namespace SAaCSim

{

class Task18

{

private readonly double ρ;

private readonly double π1;

private readonly double π2;

private readonly int ticks;

private readonly Dictionary<string, byte> SystemStates = new Dictionary<string, byte>

{

["000"] = 0,

["100"] = 1,

["101"] = 2,

["111"] = 3,

["211"] = 4,

["011"] = 5,

["001"] = 6

};

private Request π1CurrentTick;

private Request π2CurrentTick;

private Request qCurrentTick;

private Request π1NextTick;

private Request π2NextTick;

private Request qNextTick;

private const Request Empty = null;

private List<Request> requests;

private Random random;

private int[] P;

public Task18(double ρ, double π1, double π2, int ticks = 10000)

{

this.ρ = ρ;

this.π1 = π1;

this.π2 = π2;

this.ticks = ticks;

}

public void Execute()

{

random = new Random();

requests = new List<Request>();

π1CurrentTick = Empty;

π2CurrentTick = Empty;

qCurrentTick = Empty;

P = new int[7];

Run();

Display();

}

private void Display()

{

Console.WriteLine(" p: {0:0.00}", ρ);

Console.WriteLine("n1: {0:0.00}", π1);

Console.WriteLine("n2: {0:0.00}", π2);

Console.WriteLine("---------------");

Console.WriteLine("P1 - 000: {0:0.00000}", P[0] / (double) ticks);

Console.WriteLine("P2 - 100: {0:0.00000}", P[1] / (double) ticks);

Console.WriteLine("P3 - 101: {0:0.00000}", P[2] / (double) ticks);

Console.WriteLine("P4 - 111: {0:0.00000}", P[3] / (double) ticks);

Console.WriteLine("P5 - 211: {0:0.00000}", P[4] / (double) ticks);

Console.WriteLine("P6 - 011: {0:0.00000}", P[5] / (double) ticks);

Console.WriteLine("P7 - 001: {0:0.00000}", P[6] / (double) ticks);

Console.WriteLine("---------------");

Console.WriteLine("A: {0:0.000}", requests.Count(x => x.State == RequestState.Completed) / (double) ticks);

Console.WriteLine("Pотк: {0:0.000}", requests.Count(x => x.State == RequestState.Discarded) / (double) requests.Count);

Console.WriteLine("Pbl: {0:0.000000}", P[SystemStates["211"]] / (double)ticks);

Console.WriteLine("Q: {0:0.000}", requests.Count(x => x.State == RequestState.Completed) / (double)requests.Count);

Console.WriteLine("Loch: {0:0.000}", (P[5] + P[3] + P[4]) / (double)ticks);

Console.WriteLine("Lc: {0:0.000}", (P[1] + P[6] + 2 \* (P[2] + P[5]) + 3 \* (P[3] + P[4])) / (double)ticks);

Console.WriteLine("Woch: {0:0.000}", (P[3] + P[4] + P[5]) / (double)requests.Count(x => x.State == RequestState.Completed));

Console.WriteLine("Wc: {0:0.000}", requests.Sum(x => x.ExistingTime) / (double)requests.Count(x => x.State != RequestState.Discarded));

Console.WriteLine("K1: {0:0.000}", (P[1] + P[2] + P[3] + P[4]) / (double)ticks);

Console.WriteLine("K2: {0:0.000}", (P[3] + P[4] + P[5] + P[6]) / (double)ticks);

}

private void Run()

{

for (int i = 0; i < ticks - 1; ++i)

{

π1CurrentTick = π1NextTick;

π2CurrentTick = π2NextTick;

qCurrentTick = qNextTick;

π1NextTick = Empty;

π2NextTick = Empty;

qNextTick = Empty;

CalculateP();

Process\_π2();

Process\_Queue();

Process\_π1();

Process\_ρ(i);

requests.ForEach(x => x.TicksPassed());

}

}

private void CalculateP()

{

string CurrentState = π1CurrentTick == null ? "0" : ((int) π1CurrentTick.State).ToString();

CurrentState += qCurrentTick == null ? "0" : ((int)qCurrentTick.State).ToString();

CurrentState += π2CurrentTick == null ? "0" : ((int)π2CurrentTick.State).ToString();

P[SystemStates[CurrentState]]++;

}

private void Process\_π2()

{

if (π2CurrentTick == Empty)

{

π2NextTick = Empty;

}

else if (π2CurrentTick.State == RequestState.Processing)

{

if (random.NextDouble() < π2)

{

π2NextTick = π2CurrentTick;

}

else

{

π2CurrentTick.State = RequestState.Completed;

π2NextTick = Empty;

}

}

}

private void Process\_Queue()

{

if (qCurrentTick == Empty)

{

qNextTick = Empty;

}

else if (qCurrentTick.State == RequestState.Processing)

{

if (π2NextTick == Empty)

{

π2NextTick = qCurrentTick;

qNextTick = Empty;

}

else if (π2CurrentTick.State == RequestState.Processing)

{

qNextTick = qCurrentTick;

}

}

}

private void Process\_π1()

{

if (π1CurrentTick == Empty)

{

π1NextTick = Empty;

}

else if (π1CurrentTick.State == RequestState.Processing)

{

if (random.NextDouble() < π1)

{

π1NextTick = π1CurrentTick;

}

else

{

if (qNextTick == Empty)

{

if (π2NextTick == Empty)

{

π2NextTick = π1CurrentTick;

π1CurrentTick = Empty;

}

else

{

qNextTick = π1CurrentTick;

π1NextTick = Empty;

}

}

else if (qNextTick.State == RequestState.Processing)

{

π1CurrentTick.State = RequestState.Pending;

π1NextTick = π1CurrentTick;

}

}

}

else if (π1CurrentTick.State == RequestState.Pending)

{

if (qNextTick == Empty)

{

π1CurrentTick.State = RequestState.Processing;

qNextTick = π1CurrentTick;

π1NextTick = Empty;

}

else if (qNextTick.State == RequestState.Processing)

{

π1CurrentTick.State = RequestState.Pending;

π1NextTick = π1CurrentTick;

}

}

}

private void Process\_ρ(int i)

{

if (random.NextDouble() >= ρ)

{

if (π1NextTick?.State == RequestState.Processing || π1NextTick?.State == RequestState.Pending)

{

requests.Add(new Request { ExistingTime = 0, CreationTact = i, State = RequestState.Discarded });

}

else

{

π1NextTick = new Request { ExistingTime = 0, CreationTact = i, State = RequestState.Processing };

requests.Add(π1NextTick);

}

}

}

}

}

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ и РАДИОЭЛЕКТРОНИКИ

Факультет КСиС

Кафедра ПОИТ ФКСиС

Лабораторная работа № 3

Построение и исследование аналитической модели дискретно -стохастической системы массового обслуживания

Вариант № 18

Ст. гр. 951001 Преподаватель:

Мельник Н.И.

Минск

2022