УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра программного обеспечения информационных технологий

Отчет

По лабораторной работе №4

по дисциплине

Системный анализ и машинное моделирование

вариант 34

Выполнил: Студент гр. 951006

Шестакович З.В.

Проверил: Мельник Н. И.

Минск 2022

1. **Листинг кода программы**

using System.Globalization;

var P0000 = new State("P0000", 0, false, 0, false);

var P0100 = new State("P0100", 0, true, 0, false);

var P0001 = new State("P0001", 0, false, 0, true);

var P1100 = new State("P1100", 1, true, 0, false);

var P0101 = new State("P0101", 0, true, 0, true);

var P0011 = new State("P0011", 0, false, 1, true);

var P1101 = new State("P1101", 1, true, 0, true);

var P0111 = new State("P0111", 0, true, 1, true);

var P1111 = new State("P1111", 1, true, 1, true);

var stateCountPairs = new List<(State State, int Count)>()

{

(P0000, 0),

(P0100, 0),

(P0001, 0),

(P1100, 0),

(P0101, 0),

(P0011, 0),

(P1101, 0),

(P0111, 0),

(P1111, 0),

};

var random = new Random();

var state = P0000;

var generatedCount = 0;

var declinedCount0 = 0;

var queueLength1 = 0;

var channelLength1 = 0;

var processedCount1 = 0;

var processedOnQueue1 = 0;

var declinedCount1 = 0;

var l1 = 0;

var queueLength2 = 0;

var channelLength2 = 0;

var processedCount2 = 0;

var processedOnQueue2 = 0;

var l2 = 0;

var requestLength = 0;

Console.Write("Enter N: ");

var count = (int)ParseFloat(Console.ReadLine());

Console.Write("Enter p: ");

var probP = ParseFloat(Console.ReadLine());

Console.Write("Enter pi1: ");

var probPi1 = ParseFloat(Console.ReadLine());

Console.Write("Enter pi2: ");

var probPi2 = ParseFloat(Console.ReadLine());

Run();

var stateProbPairs = stateCountPairs.Select(pair => (State: pair.State, Prob:(double)pair.Count / count)).ToList();

var stateProbPairsString = string.Join("\r\n", stateProbPairs.Select(pair => $"{pair.State.Name}: {pair.Prob}"));

var sum = stateProbPairs.Select(pair => pair.Prob).Sum();

var a = (double)processedCount2 / count;

var q = (double)processedCount2 / generatedCount;

var p = 1 - q;

var lq1 = (double)queueLength1 / count;

var lq2 = (double)queueLength2 / count;

var lc1 = (double)requestLength/ count;

var wq1 = processedOnQueue1 == 0 ? 0: (double)queueLength1 / processedOnQueue1;

var wq2 = processedOnQueue2 == 0 ? 0 : (double)queueLength2 / processedOnQueue2;

var wq = wq1 + wq2;

var wc = (double)channelLength1 / processedCount1 + (double)channelLength2 / processedCount2 + wq;

var k1 = (double)channelLength1 / count;

var k2 = (double)channelLength2 / count;

var message =

$@"{stateProbPairsString}

Sum: {sum}

A: {a}

Q: {q}

P: {p}

Lq1: {lq1}

Lq2: {lq2}

Lc: {lc1}

Wq1: {wq1}

Wq2: {wq2}

Wq: {wq}

Wc: {wc}

K1: {k1}

K2: {k2}";

Console.WriteLine(message);

void Run()

{

for (int i = 0; i < count; i++)

{

var p = NextBoolean(probP);

var pi1 = NextBoolean(probPi1);

var pi2 = NextBoolean(probPi2);

Action(p, pi1, pi2);

}

}

void Action(bool p, bool pi1, bool pi2)

{

if (state == P0000)

{

if (p)

{

state = P0000;

}

else if (!p)

{

state = P0100;

generatedCount++;

}

}

else if (state == P0100)

{

if (p && pi1)

{

state = P0100;

}

else if (p && !pi1)

{

state = P0001;

processedCount1++;

}

else if (!p && pi1)

{

state = P1100;

generatedCount++;

}

else if (!p && !pi1)

{

state = P0101;

generatedCount++;

processedCount1++;

}

}

else if (state == P0001)

{

if (p && !pi2)

{

state = P0000;

processedCount2++;

}

else if (p && pi2)

{

state= P0001;

}

else if (!p && !pi2)

{

state = P0100;

processedCount2++;

generatedCount++;

}

else if (!p && pi2)

{

state = P0101;

generatedCount++;

}

}

else if (state == P1100)

{

if (p && pi1)

{

state = P1100;

}

else if (!p && pi1)

{

state = P1100;

generatedCount++;

declinedCount0++;

}

else if (p && !pi1)

{

state = P0101;

processedCount1++;

processedOnQueue1++;

}

else if (!p && !pi1)

{

state = P1101;

generatedCount++;

processedCount1++;

processedOnQueue1++;

}

}

else if (state == P0101)

{

if (p && pi1 && !pi2)

{

state = P0100;

processedCount2++;

}

else if (!p && !pi1 && !pi2)

{

state = P0101;

generatedCount++;

processedCount1++;

processedCount2++;

}

else if (p && pi1 && pi2)

{

state = P0101;

}

else if (p && !pi1 && !pi2)

{

state = P0001;

processedCount1++;

processedCount2++;

}

else if (p && !pi1 && pi2)

{

state = P0011;

processedCount1++;

}

else if (!p && !pi1 && pi2)

{

state = P0111;

generatedCount++;

processedCount1++;

}

else if (!p && pi1 && pi2)

{

state = P1101;

generatedCount++;

}

else if (!p && pi1 && !pi2)

{

state = P1100;

generatedCount++;

processedCount2++;

}

}

else if (state == P0011)

{

if (p && !pi2)

{

state = P0001;

processedCount2++;

processedOnQueue2++;

}

else if (p && pi2)

{

state = P0011;

}

else if (!p && !pi2)

{

state = P0101;

generatedCount++;

processedCount2++;

processedOnQueue2++;

}

else if (!p && pi2)

{

state = P0111;

generatedCount++;

}

}

else if (state == P1101)

{

if (p && pi1 && !pi2)

{

state = P1100;

processedCount2++;

}

else if (!p && pi1 && !pi2)

{

state = P1100;

generatedCount++;

declinedCount0++;

processedCount2++;

}

else if (p && !pi1 && !pi2)

{

state = P0101;

processedCount1++;

processedOnQueue1++;

processedCount2++;

}

else if (p && !pi1 && pi2)

{

state = P0111;

processedCount1++;

processedOnQueue1++;

}

else if (!p && !pi1 && pi2)

{

state = P1111;

generatedCount++;

processedCount1++;

processedOnQueue1++;

}

else if (!p && !pi1 && !pi2)

{

state = P1101;

generatedCount++;

processedCount1++;

processedOnQueue1++;

processedCount2++;

}

else if (p && pi1 && pi2)

{

state = P1101;

}

else if (!p && pi1 && pi2)

{

state = P1101;

generatedCount++;

declinedCount0++;

}

}

else if (state == P0111)

{

if (p && pi1 && !pi2)

{

state = P0101;

processedCount2++;

processedOnQueue2++;

}

else if (p && !pi1 && !pi2)

{

state = P0011;

processedCount1++;

processedCount2++;

processedOnQueue2++;

}

else if (p && !pi1 && pi2)

{

state = P0011;

processedCount1++;

declinedCount1++;

}

else if (!p && !pi1 && !pi2)

{

state = P0111;

generatedCount++;

processedCount1++;

processedCount2++;

processedOnQueue2++;

}

else if (!p && !pi1 && pi2)

{

state = P0111;

generatedCount++;

processedCount1++;

declinedCount1++;

}

else if (p && pi1 && pi2)

{

state = P0111;

}

else if (!p && pi1 && !pi2)

{

state = P1101;

generatedCount++;

processedCount2++;

processedOnQueue2++;

}

else if (!p && pi1 && pi2)

{

state = P1111;

generatedCount++;

}

}

else if (state == P1111)

{

if (p && pi1 && !pi2)

{

state = P1101;

processedCount2++;

processedOnQueue2++;

}

else if (!p && pi1 && !pi2)

{

state = P1101;

generatedCount++;

declinedCount0++;

processedCount2++;

processedOnQueue2++;

}

else if (p && !pi1 && !pi2)

{

state = P0111;

processedCount1++;

processedOnQueue1++;

processedCount2++;

processedOnQueue2++;

}

else if (p && !pi1 && pi2)

{

state = P0111;

processedCount1++;

processedOnQueue1++;

declinedCount1++;

}

else if (!p && !pi1 && !pi2)

{

state = P1111;

generatedCount++;

processedCount1++;

processedOnQueue1++;

processedCount2++;

processedOnQueue2++;

}

else if (!p && !pi1 && pi2)

{

state = P1111;

generatedCount++;

processedCount1++;

processedOnQueue1++;

declinedCount1++;

}

else if (p && pi1 && pi2)

{

state = P1111;

}

else if (!p && pi1 && pi2)

{

state = P1111;

generatedCount++;

declinedCount0++;

}

}

var ql1 = state.QueueLength1;

var ql2 = state.QueueLength2;

var v1 = AsInt(state.IsProcessed1);

var v2 = AsInt(state.IsProcessed2);

queueLength1 += ql1;

channelLength1 += v1;

l1 += ql1 + v1;

queueLength2 += ql2;

channelLength2 += v2;

l2 += ql2 + v2;

requestLength += ql1 + v1 + ql2 + v2;

IncrementStateCount(state);

}

int AsInt(bool b) => b ? 1 : 0;

bool NextBoolean(float p) => random.NextDouble() < p;

float ParseFloat(string s) => float.Parse(s, CultureInfo.InvariantCulture);

void IncrementStateCount(State state)

{

for (int i = 0; i < stateCountPairs.Count; i++)

{

var pair = stateCountPairs[i];

if (Equals(state, pair.State))

{

pair.Count++;

stateCountPairs[i] = pair;

return;

}

}

}

record class State(string Name, int QueueLength1, bool IsProcessed1, int QueueLength2, bool IsProcessed2);

1. **Результат работы программы**

