Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра программного обеспечения информационных технологий

ОТЧЁТ

по лабораторной работе №4

по курсу «Системный анализ и машинное моделирование»

Вариант 12

|  |  |
| --- | --- |
| Выполнил:  студент гр. 351001  Владимирова Н.Г. | Проверил:  ст. преподаватель кафедры ПОИТ  Мельник Н.И. |

Минск 2016

# **ЗАДАНИЕ**

Автозаправочная станция (АЗС) имеет *n* колонок; площадка возле нее допускает одновременное ожидание не более *m* автомашин. Поток автомашин, пребывающих на АЗС, простейший с интенсивностью *λ*. Время обслуживания показательное со средним значением tобсл.

1. λ=1 маш/мин, tобсл = 2 мин.

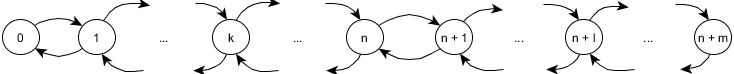
Найти вероятности отказа *Pотк* и среднее значение длины очереди *Lоч* при *n = 2* для значений *m* от 3 до 7;

1. λ=1 маш/мин, tобсл = 3 мин, *n=3*. Определить экономически обоснованное число мест ожидания, если заправка одной машины приносит доход 5 у.е., а аренда одного места ожидания стоит 20 у.е./час.

Построить аналитическую и имитационную модели и сравнить результаты исследования.

1. **ДИАГРАММА ИНТЕНСИВНОСТЕЙ ПЕРЕХОДОВ**

Диаграмма интенсивностей переходов представлена на рисунке ниже:



*λ -* интенсивность потока автомашин, пребывающих на АЗС;

*µ -* интенсивность канала.

Состояния:

1. *0 –* каналы свободны, очередь пуста;
2. *1 –* один канал занят, очередь пуста;
3. *k – k* каналов занято, очередь пуста;
4. *n –* все каналы заняты, очередь пуста;
5. *n + 1 –* все каналы заняты, один элемент в очереди;
6. *n + l –* все каналы заняты, *l* элементов в очереди;
7. *n + m –* все каналы заняты, очередь заполнена.
8. **АНАЛИТИЧЕСКАЯ МОДЕЛЬ**

Для нахождения вероятностей состояний построим систему уравнений по диаграмме, дополним ее нормировочным уравнением:

|  |  |
| --- | --- |
| *P0 ⋅ λ = P1 ⋅ µ*  *P1⋅ λ = P2 ⋅ 2µ*  *P2⋅ λ = P3 ⋅ 3µ*  *…*  *Pk ⋅ λ = Pk+1 ⋅ (k + 1)µ*  *Pn-1 ⋅ λ = Pn ⋅ nµ* | *Pn ⋅ λ = Pn+1 ⋅ nµ*  *Pn+1 ⋅ λ = Pn+2 ⋅ nµ*  *…*  *Pn+m-1 ⋅ λ = Pn+m ⋅ nµ*  *P0 + P1 + … + Pn+m = 1* |

Пусть *P0 = p, *. Выразим все вероятности через *p*:

*P1 = ωp*

*P2 = ωP1 =ω2p*

*P3 = ωP2 = ω3p = ω3p*

*…*

*Pk = ωPk-1 = ωkp (k = 1..n)*

*…*

*Pn = p*

*Pn+1 = Pn = p*

*…*

*Pn+i = Pn+i-1 = p (i = 1..m)*

*…*

*Pn+m = Pn+m-1 = p*

Подставим полученные вероятности в нормировочное уравнение:

*p + ωp + ω2p + … + p + p + … + p = 1*

Получим *p*:



1. **ХАРАКТЕРИСТИКИ СМО**
2. *n = 2 канала, λ = 1маш/мин, µ =  = 0.5маш/мин, m = 3..7*

При указанных значениях получим:

*=2*



Отказ происходит, когда *m* мест в очереди заняты:

*Pотк = Pn+m = p = *

Cреднее значение длины очереди *Lоч*:



* *m = 3*

*Pотк = 0.1818*

*Lоч = 1.0909*

* *m = 4*

*Pотк = 0.1538*

*Lоч = 1.5385*

* *m = 5*

*Pотк = 0.1333*

*Lоч = 2*

* *m = 6*

*Pотк = 0.1176*

*Lоч = 2.4706*

* *m = 7*

*Pотк = 0.1053*

*Lоч = 2.9474*

1. *n = 3 канала, λ = 1маш/мин = 60маш/час,*

*µ =  = 20маш/час*

Заправка одной машины приносит доход 5 у.е.

Аренда одного места ожидания стоит 20 у.е./час.

При указанных значениях получим:

*=3*



Вероятность отказа *Pотк* равна вероятности состояния *Pn+m*:

*Pотк = Pn+m = p = *

Абсолютная пропускная способность (среднее число машин, заправленных за час):

*A = λQ = λ(1 – Pотк) = (маш/час)*

Тогда прибыль, полученная от заправки машин, составит:

*5A = (у.е./час)*

Затраты на аренду *m* мест ожидания составят *20m (у.е./час)*

Учитывая прибыль и затраты, составим функцию зависимости суммарного дохода от количества мест ожидания:

*g(m) = 5A – 20m =  - 20m (у.е./час)*

Для того, чтобы определить максимальную прибыль, необходимо найти максимум функции *g(m)*. Для этого найдем производную фунцкии и точки экстремума:

*g’(m) = *

Функция имеет максимальное значение в точке *m = 0.984*. Т.к. *m* должно быть целым числом, рассмотрим два ближайших целых:

* при *m = 0 g(0) = 196.15 (у.е./час)*
* при *m = 1 g(1) = 202.86 (у.е./час)*

Таким образом, экономически выгодное число мест ожидания *m = 1*, доход при этом составит *202.86 (у.е./час)*.

# **ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПРОГРАММЫ**

public class CarChannel

{

private readonly Random rnd = new Random();

private readonly int serviceTime;

private int currentServiceTime;

private int currentTimeInChannel;

public CarChannel(int serviceTime)

{

this.serviceTime = serviceTime;

}

public int CarsProcessed { get; set; }

public bool IsFree { get; private set; } = true;

public void DoIteration()

{

if (IsFree)

{

return;

}

if (currentTimeInChannel == currentServiceTime)

{

IsFree = true;

currentTimeInChannel = 0;

CarsProcessed++;

}

else

{

currentTimeInChannel++;

}

}

public void SetCarForProcessing()

{

IsFree = false;

currentServiceTime = GetCurrentServiceTime();

}

private int GetCurrentServiceTime()

{

return (int)Math.Round(-serviceTime \* Math.Log(1 - rnd.NextDouble()) \* 60);

}

}

public class CarQueue

{

private readonly int queueLength;

public CarQueue(int queueLength)

{

this.queueLength = queueLength;

}

public int Count { get; private set; }

public bool Push()

{

if (Count == queueLength)

{

return false;

}

Count++;

return true;

}

public bool Pop()

{

if (Count == 0)

{

return false;

}

Count--;

return true;

}

}

public class GasStation

{

private readonly Random rnd = new Random();

private readonly int minutesCount;

private readonly int queueLength;

private readonly int intensity;

private List<CarChannel> channels;

private CarQueue queue;

private readonly int channelsCount;

private readonly int serviceTime;

private int totalQueueLength;

private int totalRefusalCount;

private int totalCarsCount;

int placesCountWithMaxIncome = 0;

double maxIncome = 0;

public GasStation(int minutesCount, int queueLength, int channelsCount, int serviceTime, int intensity)

{

this.minutesCount = minutesCount;

this.queueLength = queueLength;

this.intensity = intensity;

queue = new CarQueue(queueLength);

channels = new List<CarChannel>();

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

{

channels.Add(new CarChannel(serviceTime));

}

}

public GasStation(int channelsCount, int serviceTime, int intensity)

{

this.channelsCount = channelsCount;

this.serviceTime = serviceTime;

this.intensity = intensity;

}

public void DoWork()

{

var arrivedCars = GetArrivedCarsPositions(minutesCount);

Work(arrivedCars, minutesCount);

}

public void CalculateIncome(int carProcessedIncome, int waitingPlaceRent)

{

int minutesCount = 5000;

var arrivedCars = GetArrivedCarsPositions(minutesCount);

int placesCount = 0;

while (placesCount != 50)

{

queue = new CarQueue(placesCount);

channels = new List<CarChannel>();

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

{

channels.Add(new CarChannel(serviceTime));

}

Work(arrivedCars, minutesCount);

var totalCarsProcessed = channels.Sum(car => car.CarsProcessed);

var income = totalCarsProcessed \* carProcessedIncome;

var hoursCount = (double) minutesCount / 60;

var cost = hoursCount \* waitingPlaceRent \* placesCount;

var profit = (income - cost) / hoursCount;

if (profit > maxIncome)

{

maxIncome = profit;

placesCountWithMaxIncome = placesCount;

}

placesCount++;

}

}

public SystemCharacteristics GetSystemCharacteristics()

{

return new SystemCharacteristics

{

PlacesCountWithMaxIncome = placesCountWithMaxIncome,

MaxIncome = maxIncome,

AverageQueueLength = (double) totalQueueLength/(minutesCount \* 60),

RefusalProbability = totalCarsCount == 0 ? 0 : (double) totalRefusalCount/totalCarsCount

};

}

private void Work(bool[] arrivedCars, int minutesCount)

{

for (int i = 0; i < minutesCount \* 60; i++)

{

int arrivedCarsCount = arrivedCars[i] ? 1 : 0;

totalCarsCount += arrivedCarsCount;

channels.ForEach(channel =>

{

if (!channel.IsFree)

{

return;

}

if (queue.Pop())

{

channel.SetCarForProcessing();

}

else

{

if (arrivedCarsCount == 0)

{

return;

}

arrivedCarsCount--;

channel.SetCarForProcessing();

}

});

for (int j = 0; j < arrivedCarsCount; j++)

{

totalRefusalCount += queue.Push() ? 0 : 1;

}

totalQueueLength += queue.Count;

channels.ForEach(channel => channel.DoIteration());

}

}

private bool[] GetArrivedCarsPositions(int minutesCount)

{

var result = new bool[minutesCount \* 60];

if (intensity != 0)

{

for (int i = 0; i < minutesCount\*60; i += 60)

{

result[(i + GetCurrentArrivedCarPosition()) % result.Length] = true;

}

}

return result;

}

private int GetCurrentArrivedCarPosition()

{

return (int)Math.Round(-1.0 / intensity \* Math.Log(1 - rnd.NextDouble()) \* 60);

}

}

public class SystemCharacteristics

{

public double RefusalProbability { get; set; }

public double AverageQueueLength { get; set; }

public int PlacesCountWithMaxIncome { get; set; }

public double MaxIncome { get; set; }

}

public partial class MainForm : Form

{

private int minutesCount = 50000;

private int accuracy = 7;

public MainForm()

{

InitializeComponent();

}

private void btnCalculate\_Click(object sender, EventArgs e)

{

int channelsCount = int.Parse(txtChannelsCount.Text);

int minQueueLength = int.Parse(txtMinQueueLength.Text);

int maxQueueLength = int.Parse(txtMaxQueueLength.Text);

int intensity = int.Parse(txtIntensity.Text);

int serviceTime = int.Parse(txtServiceTime.Text);

txtCharacteristics.Text = string.Empty;

for (int queueLength = minQueueLength; queueLength <= maxQueueLength; queueLength++)

{

var gasStation = new GasStation(minutesCount, queueLength, channelsCount, serviceTime, intensity);

gasStation.DoWork();

var characteristics = gasStation.GetSystemCharacteristics();

txtCharacteristics.Text += $"m = {queueLength} {Environment.NewLine}" +

$" P(refusal) = {Math.Round(characteristics.RefusalProbability, accuracy)} {Environment.NewLine} " +

$" L(queue) = {Math.Round(characteristics.AverageQueueLength, accuracy)} {Environment.NewLine} " +

$"{Environment.NewLine}";

}

}

private void btnCalcMaxIncome\_Click(object sender, EventArgs e)

{

int channelsCount = int.Parse(txtChannelsCount.Text);

int intensity = int.Parse(txtIntensity.Text);

int serviceTime = int.Parse(txtServiceTime.Text);

txtCharacteristics.Text = string.Empty;

var gasStation = new GasStation(channelsCount, serviceTime, intensity);

gasStation.CalculateIncome(5, 20);

var characteristics = gasStation.GetSystemCharacteristics();

txtCharacteristics.Text += $"m = {characteristics.PlacesCountWithMaxIncome} {Environment.NewLine}" +

$"income = {Math.Round(characteristics.MaxIncome, accuracy)}";

}

}