Белорусский государственный университет информатики и радиоэлектроники

Кафедра ПОИТ

Отчет по лабораторной работе № 3

«Аналитическое моделирование дискретно-стохастической СМО и построение её имитационной модели»

Проверил:

Огородник Р.В.

Выполнил:

студент группы 150501

Белов А.В.

Минск 2014

**Задание 1**.

Построить граф состояний P-схемы.



n = {2, 1} – кол-во тактов до поступления очередной заявки

j = {0, 1, 2} – кол-во заявок в очереди

t1 = {0, 1} – кол-во заявок в канале №1

t2 = {0, 1} – кол-во заявок в канале №2

Общий вид кодировки состояния системы:

{n, j, t1, t2}

По графу построим аналитическую модель и, решив ее, определим вероятности состояний.



























Решив систему уравнений (при **pi1**=0,6 **pi2**=0,5), получили:

P2000 = 0

P1000 = 0.149

P2010 = 0.274

P1010 = 0.228

P2011 = 0.195

P1001 = 0.039

P1011 = 0.073

P2111 = 0.027

P1111 = 0.01

P2211 = 0.004

P1211 = 0.001

Исходя из полученных данных, рассчитаем теоретические значения средней длины очереди (Lоч) и абсолютной пропускной способности (A):

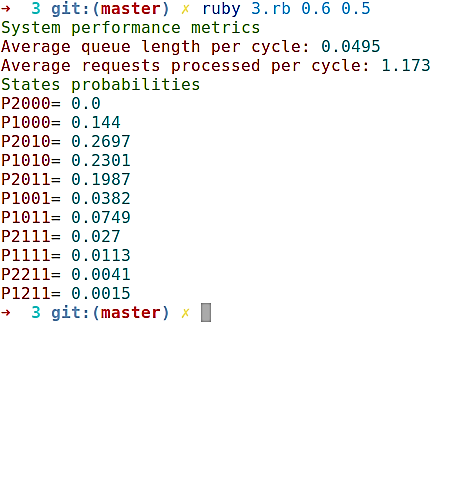




**Задание 2**

Для СМО из задания 1 построить имитационную модель и исследовать ее (разработать алгоритм и написать имитирующую программу, предусматривающую сбор и статистическую обработку данных для получения оценок заданных характеристик СМО). Распределение интервалов времени между заявками во входном потоке и интервалов времени обслуживания – геометрическое с соответствующим параметром (pi1, pi2).

**Результат работы программы:**



**Вывод:**

В ходе лабораторной работы была аналитически смоделирована дискретно- стохастическая СМО и разработана программа, имитирующая поведение данной СМО. Построенная модель позволяет статистически подсчитать характеристики СМО. Статистическое значение искомой характеристики оказывается близким к теоретически рассчитанному. Значит имитационная модель построена верно. Было также замечено, что на выходные данные влияют параметры СМО, такие как **pi1**, **pi2**.

**Листинг программы:**

require 'colored'

class Generator

def initialize

@pi1 = ARGV[0].to\_f

@pi2 = ARGV[1].to\_f

@cycles = ARGV[2].to\_f

@pi1 = 0.6 if @pi1 <= 0 or @pi1 > 1.0

@pi2 = 0.5 if @pi2 <= 0 or @pi2 > 1.0

@cycles = 10000 if @cycles < 1 or @pi2 > 1e7

@cycles\_berore\_request = 2

@queue = 0

@service\_1 = 0

@service\_2 = 0

@services\_count = 0

@queue\_processed = 0

@log = File.open("logfile.txt", "w")

@log.puts "2000"

@states = { "2000" => 0,

"1000" => 0,

"2010" => 0,

"1010" => 0,

"2011" => 0,

"1001" => 0,

"1011" => 0,

"2111" => 0,

"1111" => 0,

"2211" => 0,

"1211" => 0

}

end

def no\_service\_1

rand < @pi1

end

def no\_service\_2

rand < @pi2

end

def request?

@cycles\_berore\_request -= 1

if @cycles\_berore\_request == 0

@cycles\_berore\_request = 2

return true

end

return false

end

def count\_states

@states["#{@cycles\_berore\_request}#{@queue}#{@service\_1}#{@service\_2}"] += 1

end

def generate\_new\_state

if @service\_1 == 1

if no\_service\_1

else

@service\_1 = 0

end

end

if @service\_2 == 1

if no\_service\_2

else

@service\_2 = 0

end

end

if @queue > 0

if @service\_1 == 0 and @service\_2 == 1

@queue -= 1

@service\_1 = 1

elsif @service\_1 == 1 and @service\_2 == 0

@queue -= 1

@service\_2 = 1

elsif @service\_1 == 0 and @service\_2 == 0

if @queue == 2

@service\_1 = 1

@service\_2 = 1

@queue = 0

elsif @queue == 1

@service\_1 = 1

@queue = 0

end

end

end

unless request?

@log.puts "#{@cycles\_berore\_request}#{@queue}#{@service\_1}#{@service\_2}"

count\_states

@queue\_processed += @queue

@services\_count = @services\_count + @service\_1 + @service\_2

return

end

if @queue == 0

if @service\_1 == 0

@service\_1 = 1

elsif @service\_2 == 0

@service\_2 = 1

else

@queue += 1

end

elsif @queue == 2

return

elsif @queue == 1

@queue += 1

end

@log.puts "#{@cycles\_berore\_request}#{@queue}#{@service\_1}#{@service\_2}"

count\_states

@queue\_processed += @queue

@services\_count = @services\_count + @service\_1 + @service\_2

end

def run

@cycles.to\_i.times do

generate\_new\_state

end

puts "System performance metrics".green

puts "Average queue length per cycle: ".red + "#{@queue\_processed/@cycles.to\_f}".cyan

puts "Average requests processed per cycle: ".red + "#{@services\_count/@cycles.to\_f}".cyan

puts "States probabilities".green

@states.each do |state, count|

puts "P#{state}".red + "=" + " #{count/@cycles.to\_f}".cyan

end

end

end

generator = Generator.new

generator.run