Scheduling Simulator Program Generator

1. Introduction

Contained in the scheduling simulator framework is a package called '`programgenerator`'. Contained within this package is a program called ProgramGenerator that may be used to build the pseudo-programs used in simulations.

A '`make generator`' compiles the program to the '`bin`' directory, and (assuming the simulator framework has also been compiled – with a '`make framework`') the following command executes it:

```
java -ea -cp bin ProgramGenerator
```

Here is an example of its use:

```
CPU Bounds (lambda, min burst, max burst): 8, 1000, 5000
IO Bounds (lambda, min burst, max burst): 5, 2000, 10000
Device identifiers (<integer>, ..., <integer>): 1, 2, 3
Length (number of instructions - must be odd number): 11
Program name: testone.prg

# Program name: testone.prg
# CPU Generator (lambda=8.0, lower_bound=1000, upper_bound=5000)
# IO (lambda=5.0, lower_bound=2000, upper_bound=10000, devices=[1,
2, 3])
CPU 1048
IO 2794 3
CPU 1270
IO 2873 3
CPU 1362
IO 2033 3
CPU 3355
IO 8561 1
CPU 1108
IO 2517 2
CPU 1115
```

The program asks for a number of parameters, including the name of the program that is to be generated. It writes the program to file ('`testone.prg`' in the example) and simultaneously prints it on the screen.

2. Probability distributions

The program asks for two sets of 'bounds' on the instructions it generates, one for CPU instructions, and the other for I/O instructions. The values entered define the maximum and minimum values for burst length (or duration) and the probability distribution for the values within this range.

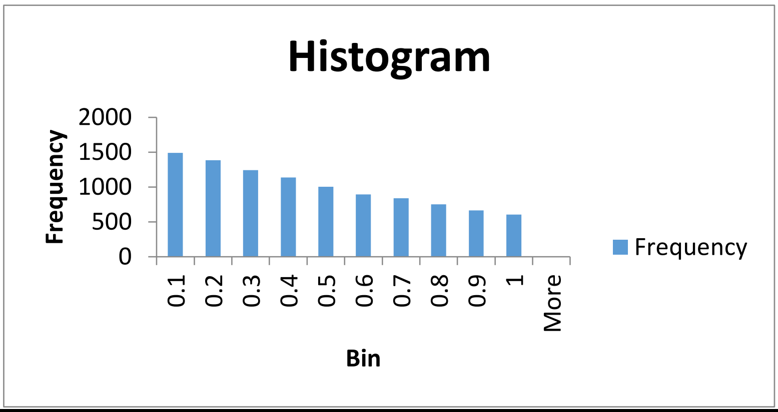
The chapter on CPU/Process scheduling in the course text (Operating Systems Concepts 9th edition) contains the observation that the durations of CPU bursts have been extensively measured and they tend to have a frequency curve that may be characterised as exponential or hyper-exponential. Drawing on this, the program generator uses a random number generator with an exponential probability distribution. It generates values in the range [0, 1], with values closer to 0 having an exponentially higher probability of occurring.

The ‘lamba’ parameter that is asked for by the program serves to define the shape of the probability curve.

We have charted the frequency distribution for values of lambda between 1 and 9. (By generating 10,000 numbers in each case and then building a histogram in Microsoft Excel). Higher values generate a curve closer to that for CPU bursts depicted in the text.

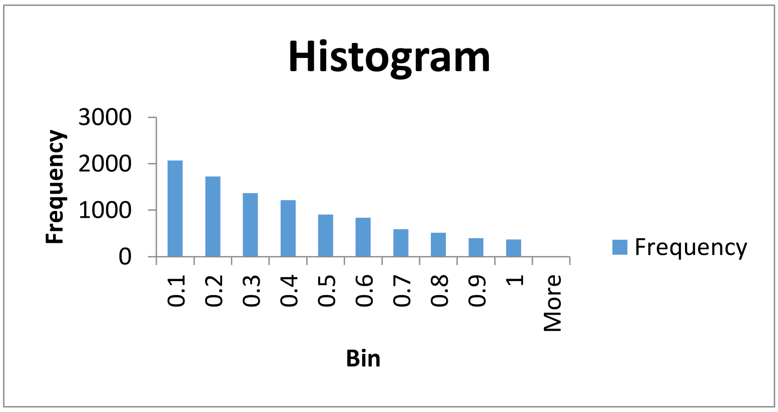
2.1 Lamba = 1

Bin	*Frequency*
0.1	1488
0.2	1384
0.3	1241
0.4	1134
0.5	1002
0.6	894
0.7	837
0.8	753
0.9	662
1	605
More	0



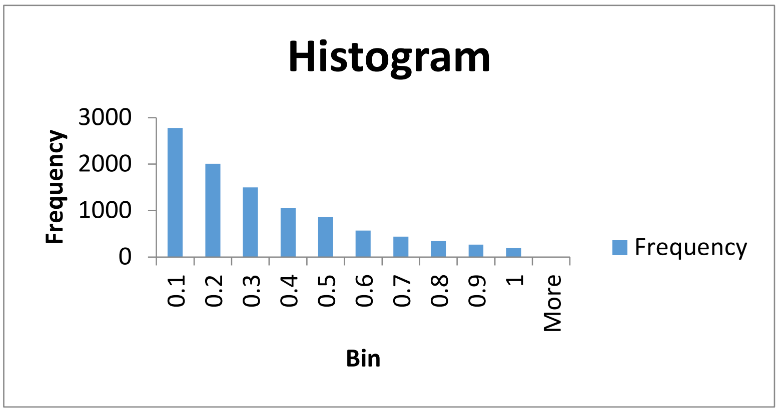
2.2 Lambda = 2

Bin	*Frequency*
0.1	2070
0.2	1728
0.3	1365
0.4	1217
0.5	906
0.6	841
0.7	593
0.8	515
0.9	398
1	367
More	0



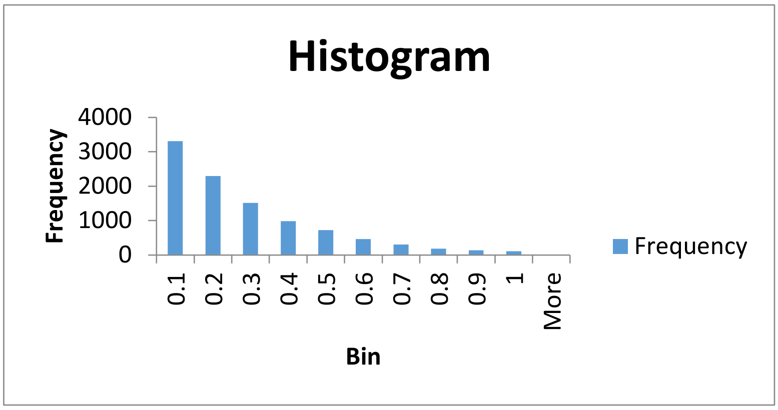
2.3 Lambda = 3

Bin	*Frequency*
0.1	2775
0.2	2007
0.3	1495
0.4	1055
0.5	859
0.6	569
0.7	442
0.8	340
0.9	265
1	193
More	0



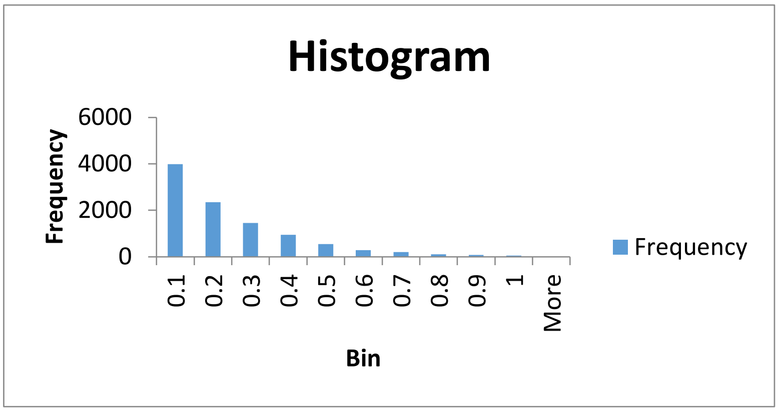
2.4 Lambda = 4

Bin	*Frequency*
0.1	3307
0.2	2290
0.3	1511
0.4	985
0.5	720
0.6	465
0.7	299
0.8	186
0.9	133
1	104
More	0



2.5 Lambda = 5

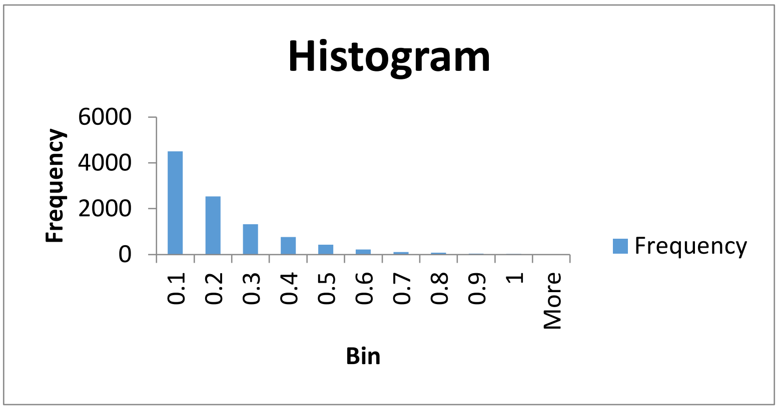
Bin	*Frequency*
0.1	3980
0.2	2345
0.3	1459
0.4	942
0.5	549
0.6	286
0.7	199
0.8	112
0.9	77
1	51



More	0

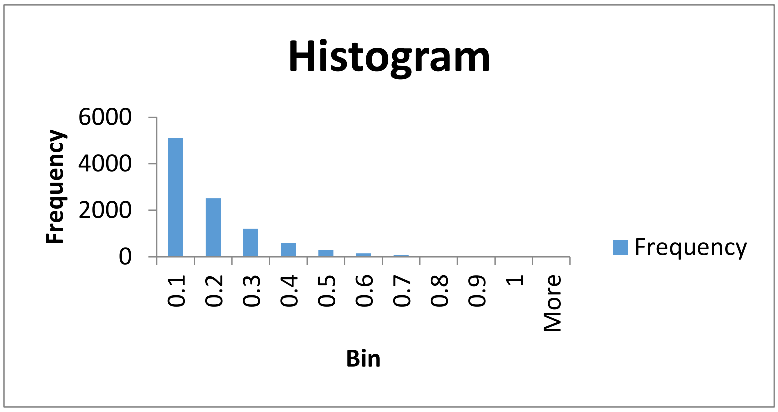
2.6 Lambda = 6

Bin	*Frequency*
0.1	4506
0.2	2535
0.3	1316
0.4	760
0.5	422
0.6	217
0.7	105
0.8	73
0.9	42
1	24
More	0



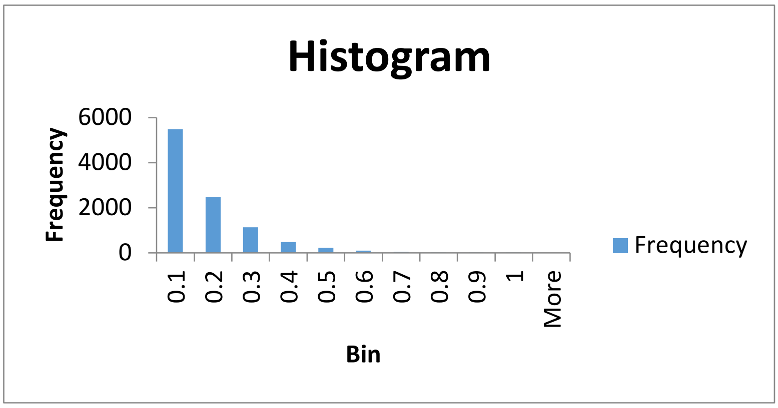
2.7 Lambda = 7

Bin	*Frequency*
0.1	5098
0.2	2518
0.3	1208
0.4	598
0.5	299
0.6	145
0.7	76
0.8	23
0.9	21
1	14
More	0



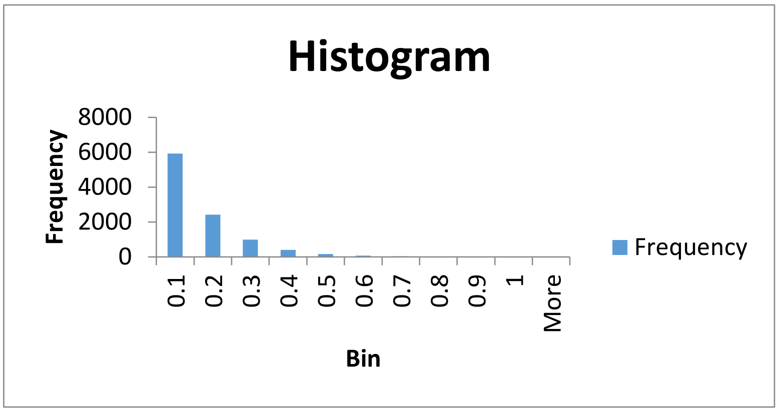
2.8 Lambda = 8

Bin	*Frequency*
0.1	5487
0.2	2481
0.3	1131
0.4	485
0.5	232
0.6	104
0.7	47
0.8	20
0.9	9
1	4
More	0



2.9 Lambda = 9

Bin	*Frequency*
0.1	5917
0.2	2419
0.3	994
0.4	397
0.5	157
0.6	74
0.7	28
0.8	8
0.9	4
1	2
More	0



END