Matthew Huynh

CS350

February 2, 2011

HW2

Collaboration: I discussed the homework with Susana Fong.

Exercise 1

a) mean = 1\*0.25 + 2\*0.42 + 3\*0.33 = 2.08

stdev = sqrt( 0.25\*(1-2.08)2 + 0.42\*(2-2.08)2 + 0.33\*(3-2.08)2 ) = 0.76

b)

|  |  |  |
| --- | --- | --- |
| P(x=3) | .253 | 0.015625 |
| P(x=4) | (3)(.252)(.42) | 0.07875 |
| P(x=5) | (3)(.252)(.33)+(3)(.422)(.25) | 0.194175 |
| P(x=6) | (.423)+(3)(.25)(.42)(.33) | 0.281988 |
| P(x=7) | (3)(.332)(.25)+(3)(.422)(.33) | 0.256311 |
| P(x=8) | (3)(.332)(.42) | 0.137214 |
| P(x=9) | (.333) | 0.035937 |

c) mean = 6.24, stdev = 1.31

d)

|  |  |
| --- | --- |
| P(x≤3) | 0.015625 |
| P(x≤4) | 0.2098 |
| P(x≤5) | 0.491788 |
| P(x≤6) | 0.758099 |
| P(x≤7) | 0.885313 |
| P(x≤8) | 0.92125 |
| P(x≤9) | 1 |

e) P(x≤5) = 0.491788

f)

|  |  |
| --- | --- |
| P(x=1) | 0.015625 |
| P(x=2) | 0.07875 |
| P(x=3) | 0.905625 |

g) mean = 2.89, stdev = 0.36

Exercise 2

q = 8, Tq = 0.1s, Ts = 0.02s

a) λ = q / Tq = 8 / 0.1s = 80 packets/s

b) w = λ\*Tw = 8\*(Tq-Ts) = 8\*0.08 = 0.64

c) 8-0.64 = 7.36 => 8 processors

d) The arrival rate is Markovian and the service rate is exponentially distributed.

Exercise 3

a) (100 choose 100) (0.98100) = 0.132619556

b) (100 choose 99) (0.9899 ) (0.021) = 0.270652155

c) P(0 failures) + P(1 failure) + P(2 failures) + P(3 failures) =

(100 choose 100)(0.98100) + (100 choose 99)(0.9899)(0.02) +

(100 choose 98)(0.9898)(0.022) + (100 choose 97)(0.9897)(0.023) = 0.858961563

d) 0.9850

e) (1-p)/p = (1-0.02)/0.02 = 49

f) mean = np = 500\*0.98 = 490

Exercise 4

a) f(x) = (1/(0.05\*sqrt(2pi))\*e-0.5\*(((x-0.2)^2)/(0.05^2))

b) P(x=2.1) – P(x=2) = z-score(x=2.1) – z-score(x=2) =

z-score((2.1-2)/0.5) – z-score((2-2)/0.5) = z-score(0.2) – z-score(0) = 0.5793 – 0.5

= 0.0793

c) P(x>2.5) > 1 - z-score(x=2.5) = 1 - 0.8413 = 0.1587

Exercise 5

import java.util.\*;

public class RandGen {

public static void main(String[] args) {

Random uniRand = new Random();

double T = 2.7;

int N = 100;

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

{

System.out.println(getReal(T, uniRand.nextDouble()));

}

}

static double getReal(double mean, double uniform)

{

double U = uniform;

double lambda = 1/mean;

double V = (-1\*Math.log(1-U))/lambda;

return V;

}

}

I choose lambda = 0.37 (mean = 2.7). The lambda from the empirical results ended up being 0.35, which is close to the analytical CDF, but it is still only an approximation. This is probably because the uniform number generator inside a computer can only emulate randomness in a pseudo fashion, but cannot replicate it precisely.