CS 350: Homework #2

Due 3PM, Tuesday, February 9

Kristel Tan (ktan@bu.edu)

January 30, 2016

Question 1

Part (a)

Pr {in a sequence of 20 connections, none gets refused} $= 0.96^{20}$

Part (b)

Pr {in a sequence of 20 connections, one gets refused}

$$= \binom{20}{1}(0.04)(0.96)^{19} = \frac{20!}{1!(20-1)!}(0.04)(0.96)^{19} = 20(0.04)(0.96)^{19}$$

Part (c)

Pr {in a sequence of 20 connections, at most three get refused} $= \sum_{i=20}^{3} \binom{20}{i}(0.04)(0.96)^{20-i}$

Part (d)

Pr {respond to 4 or more consecutive requests} =

Pr {x ≥ 4} $= \binom{n}{4} * (0.96)^4 * (0.04)^{n-1}$ where n = the number of trials

Part (e)

average # requests successfully served before a refusal occurs $= \frac{1}{0.04} = 25 \; requests$

Part (f)

average # requests successfully served out of 50 requests $= \lambda = np = 50 * 0.96 = 48 \, requests$

Part (g)

Let N (a very large number of requests) = 100. We can calculate the mean and standard deviation using N and the given 96% availability. Then, we can calculate the z-score and find its respective probability as shown below.

$$\mu = np = 100 * .96 = 96$$
$$\sigma = \sqrt{np(1-p)} = \sqrt{96(0.04)} = 1.9596$$
$$z = \frac{X-\mu}{\sigma} = \frac{95-96}{1.9596} = -0.510$$
$$\Pr\{z = -0.510\} = 0.695$$

Part (h)

Part(e) refers to the reliability of the web server because it essentially asks us about the likelihood that the server will be "up" to process requests before one gets refused. This is a characteristic of the process-center metric that we label as reliability.

Question 2

Part (a)

The probability mass function for a Poisson distribution is given by the following function. We can apply the problem's parameters to λ (average rate of event arrival of 200 packets/sec = 2 packets/ms) and t (time period of 10ms).

$$f(x) = \frac{(\lambda t)^x}{x!} e^{-\lambda t}, \, where \; x = 0, 1, 2, ...$$
$$f(x) = \frac{((2 packets / ms) * 10ms)^x}{x!} e^{-(2 packets/ms)*10ms}, \, where \; x = 0, 1, 2, ...$$
$$f(x) = \frac{2^x}{x!} e^{-2}, \, where \; x = 0, 1, 2, ...$$

Part (b)

The probability of receiving more than 3 packets can result in an infinite number of possibilities. To solve this, we can compute the probability that 3 or less packets arrive within the 10 millisecond time frame and subtract it from 1 as show below.

$$\begin{aligned}\Pr\{X > 3\} &= \Pr\{X = 4\} + \Pr\{X = 5\} + ... \\ &= 1 - \Pr\{X \le 3\} \\ &= 1 - (\Pr\{X = 0\} + \Pr\{X = 1\} + \Pr\{X = 2\} + \Pr\{X = 3\}) \\ &= 1 - \left(\frac{2^0}{0!} e^{-2} + \frac{2^1}{1!} e^{-2} + \frac{2^2}{2!} e^{-2} + \frac{2^3}{3!} e^{-2} \right)\end{aligned}$$

Part (c)

$f(x) = 1 - e^{-0.2x}$

Part (d)

$f(x) = 1 - (1 - e^{-\lambda x}) = 1 - (1 - e^{-0.2*8}) = 1 - e^{-0.16}$

Part (e)

$$\mathrm{var} = \frac{1}{\lambda^2} = \frac{1}{0.2^2} = 25 \rightarrow \sigma = \sqrt{25} = 5$$

- - -

Part (a)

$$capacity = \frac{1}{4.8} = 0.2083$$

Part (b)

$$f(x) = \left(\frac{\lambda t^x}{x!}\right) e^{-\lambda t},\ x = 0, 1, 2, ... \rightarrow Poisson\ Distribution$$

Part (c)

$$\rho = \lambda T_S = 0.2(4.8) = 0.96$$
$$w = \frac{p^2}{1-p} = \frac{0.96^2}{1-0.96} = \frac{0.9216}{.04} = 23.04$$

Part (d)

$$\mu = \frac{1}{T_S} = \frac{1}{4.8} = 0.21$$
$$T_w = \frac{w}{\lambda} = \frac{23.04}{0.2} = 115.2\,ms$$

Part (e)

$$slowdown = \frac{T_q}{T_S} = \frac{1}{1-p} = \frac{1}{(1-0.96)} = \frac{1}{0.04} = 25\,ms$$

Question 3

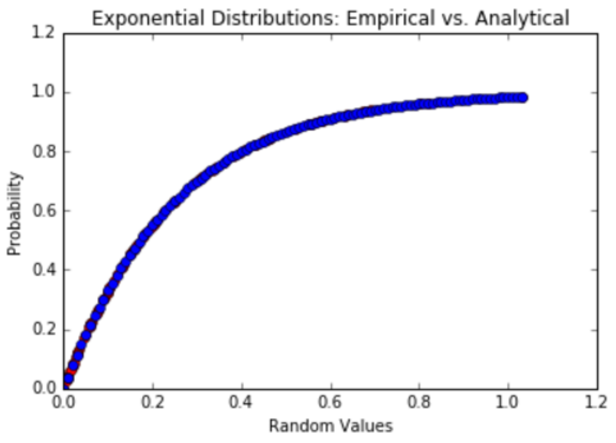
The following Python code is what I produced to help plot the CDF and normalized histogram for this problem. It uses to helper functions *empirical_ exp()* and *analytical_ exp()* to generate random values using the cumulative distribution function. Function *Q3ab()* then plots the results. Please read comments on the code for further details about the individual functions.

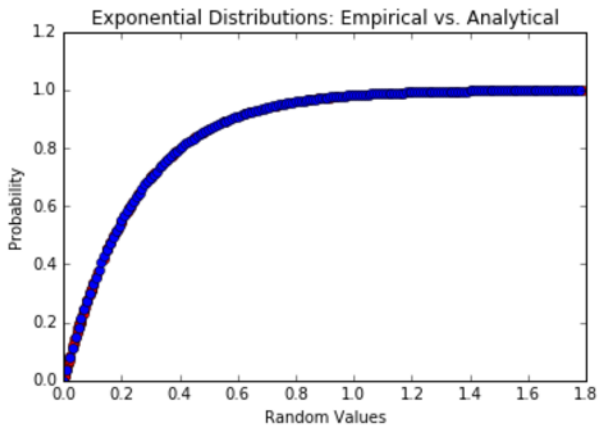
```python
import math
import random
from matplotlib import pyplot as plt

# Question 3

# empirical_exp() returns a random value using the
# cumulative distribution function for a uniformly
# distributed random variable y

def empirical_exp(y, lamb):
   return -math.log(1.0 - y) / lamb

# analytical_exp() returns a random value using the
# cumulative distribution function for an exponentially
# distributed random variable x

def analytical_exp(x, lamb):
   return 1 - math.exp(-(lamb) * x)

# Question 3a & 3b
# Q3ab() generates and plots the CDF using 100
# exponentially distributed random values (obtained
# empirically and analytically) and a given lambda of 4.

def Q3ab():

   y_data = []
   emp_data = []
   for i in range(100):
       y = random.uniform(0, 1)
       e = empirical_exp(y, 4)
       emp_data.append(e)
       y_data.append(y)

   x_data = np.arange(0, max(emp_data), 0.01)
   ana_data = []
   for i in x_data:
       a = analytical_exp(i, 4)
       ana_data.append(a)

   plt.figure(1)
   plt.plot(emp_data, y_data, 'ro')
   plt.plot(x_data, ana_data, 'bo')
   plt.title('Exponential Distributions: Empirical vs. Analytical')
   plt.ylabel('Probability')
   plt.ylim(0.0, 1.2)
   plt.xlabel('Random Values’)

   bins = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2]
```

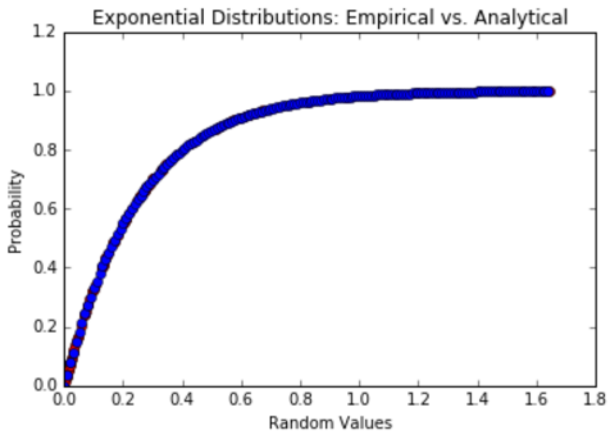
```
    plt.figure(2)
    plt.hist(emp_data, bins, normed = True, color = 'r')
    plt.title('Relative Frequency of Random Variable Queries')
    plt.ylabel('Frequency')
    plt.ylim(0.0, 3.6)
    plt.xlabel('0.1-second Intervals')

    plt.show()

Q3ab()
```

Part (a)

Some sample CDF plots produced by the code are shown below.



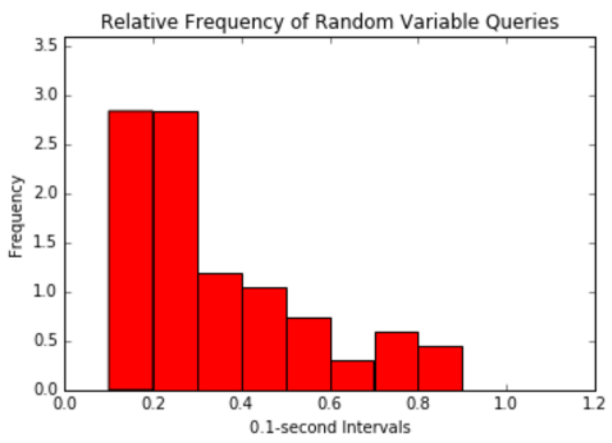


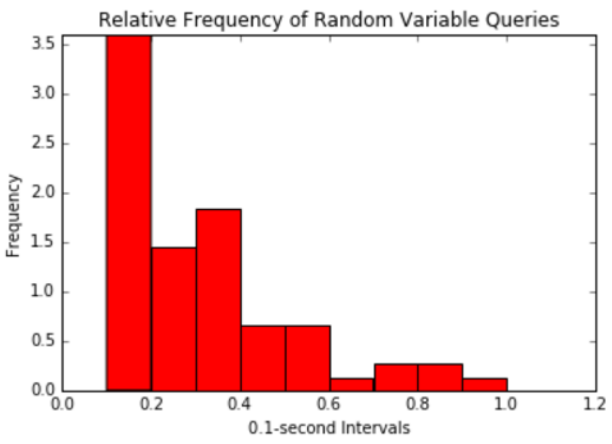


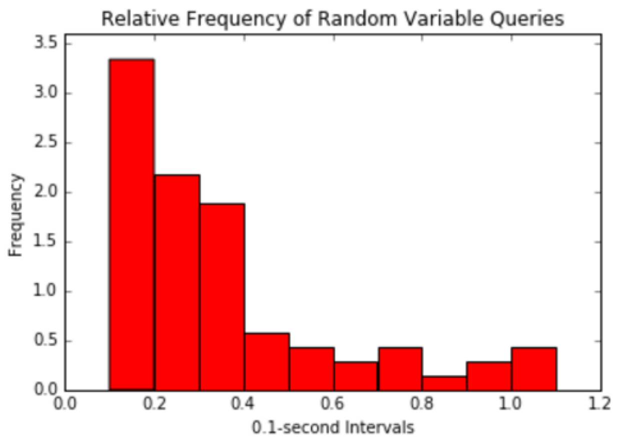
If you look closely, you can observe that the empirical and analytical CDFs are actually almost directly on top of each other. Therefore, we can say that the empirically and analytically obtained CDFs of the random variables match.

Part (b)

Some sample histogram plots produced by the code are shown below.





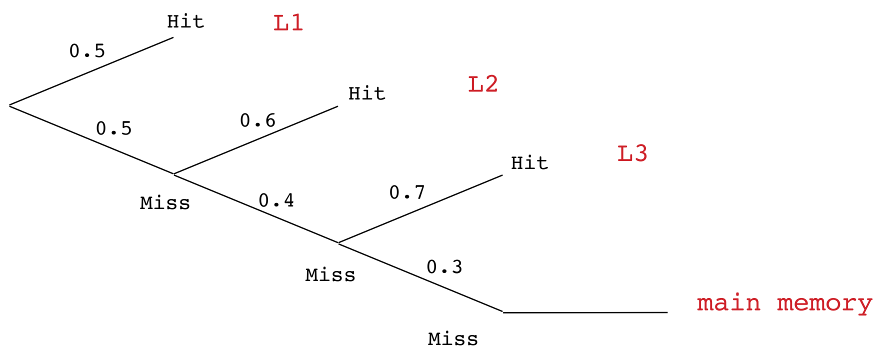


The formula of the distribution that best characterizes the empirical results of these histograms is the PDF of the exponential distribution because they follow an exponential decline. That is,

$$f(x) = \left\{ \begin{array}{ll} \lambda e^{-\lambda e} & x \geq 0 \\ 0 & x < 0 \end{array} \right\}$$

Question 4

The diagram on the next page depicts the probability tree for each cache level as given in the problem.



Part (a)

L1 access time = 1 clock cycle
L2 access time = 1 + 2 = 3 clock cycles
L3 access time = 3 + 7 = 10 clock cycles
L4 access time = 10 + 20 = 30 clock cycles

PMF —> f(x) = 0.5 when t = 1

f(x) = (0.5)*(0.6) = 0.3 when t = 3

f(x) = (0.5)*(0.4)*(0.7) = 0.14 when t = 10

f(x) = (0.5)*(0.4)*(0.3) = 0.06 when t = 30

Part (b)

$$\mu_t = (0.5)1 + (0.3)(3) + (0.14)(10) + (0.06)(30) = 4.6$$

Part (c)

$$\sigma = \sqrt{(1-4.6)^2(0.5) + (3-4.6)^2(0.3) + (10-4.6)^2(0.14) + (30-4.6)^2(0.06)} = 7.074$$

Part (d)

The following Python code includes a small function that returns a random value that is distributed according the PDF defined in part(a). In other words, it returns a random value in the choice it t values based on the probabilities that each cache will be accessed.

```
# Question 4d

# Q4d() returns a random value in the list of t values
# with their calculated probabilities taken into account
# from the PMF described in part(a)

def Q4d():

    ts = [1, 3, 10, 30]
```

```python
    probs = [0.5, 0.3, 0.14, 0.06]
    n = random.uniform(0, 1)
    cumulative_prob = 0.0

    for t, t_prob in zip(ts, probs):
        cumulative_prob += t_prob
        if n < cumulative_prob:
            break

    return t
```

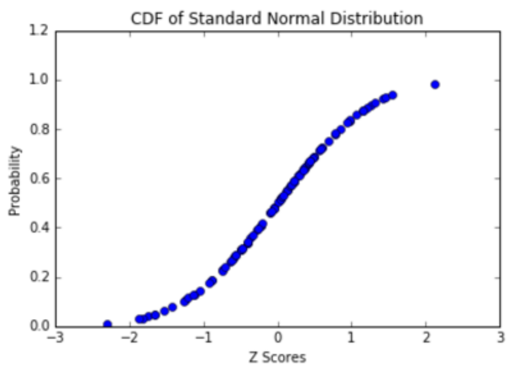
Question 5

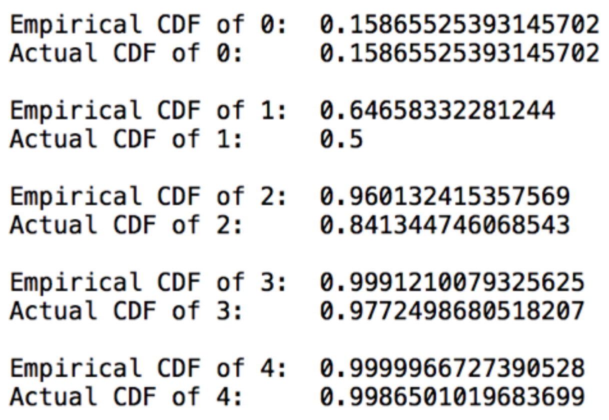
The following Python code is what I produced to help plot the CDF for this problem. It uses to helper functions *zrand()* and *norm_ dist_ cdf()* to generate empirical random values using the cumulative distribution function of a normal distribution. Function *Q5ab()* plots these results. It then compares these values with the actual CDF values for random variables 0, 1, 2, 3, and 4. Please read comments on the code for further details about the individual functions.

```python
# Question 5

# zrand() returns a random value that is distributed according to
# a standard normal distribution by mapping a relationship between
# a uniform random variable from [0, 1] and a standard normal
# random variable

def zrand(mu, sigma, y):
   return mu + (math.sqrt(2)* sigma * scipy.special.erfinv((2*y) - 1))

def norm_dist_cdf(mu, sigma, x):
    return 0.5 * (1 + math.erf((x - sigma) / (math.sqrt(2)* sigma )))

# Question 5a & 5b

# Q5ab() uses the two functions above to generate 100
# standard-normal random values. It then creates a plot of the
# empirical CDF and prints a comparison of the values
# with the actual CDF according to the normal distribution.

def Q5ab():

    zrand_data = []
    y_data = []

    for i in range(100):
        y = random.uniform(0,1)
        y_data.append(y)
        z = zrand(0, 1, y)
        zrand_data.append(z)


    plt.figure(1)
    plt.plot(zrand_data, y_data, 'bo')
    plt.title('CDF of Standard Normal Distribution')
    plt.ylabel('Probability')
    plt.ylim(0.0, 1.2)
    plt.xlabel('Z Scores')
    plt.show()

    mu = sum(zrand_data) / 100
    sigma = 0

    for z in zrand_data:
        sigma += pow(z - mu, 2)
    sigma = sigma / 99

    print('\n')

    print('Empirical CDF of 0: ', norm_dist_cdf(mu, sigma, 0))
    print('Actual CDF of 0:    ', norm_dist_cdf(0, 1, 0), '\n')

    print('Empirical CDF of 1: ', norm_dist_cdf(mu, sigma, 1))
    print('Actual CDF of 1:    ', norm_dist_cdf(0, 1, 1), '\n')

    print('Empirical CDF of 2: ', norm_dist_cdf(mu, sigma, 2))
    print('Actual CDF of 2:    ', norm_dist_cdf(0, 1, 2), '\n')

    print('Empirical CDF of 3: ', norm_dist_cdf(mu, sigma, 3))
    print('Actual CDF of 3:    ', norm_dist_cdf(0, 1, 3), '\n')

    print('Empirical CDF of 4: ', norm_dist_cdf(mu, sigma, 4))
    print('Actual CDF of 4:    ', norm_dist_cdf(0, 1, 4), '\n')

    print(norm_dist_cdf(72, 256, 80) - norm_dist_cdf(72, 256, 66))

Q5ab()
```

Part (a)

A sample CDF plot and empirical vs analytical comparisons produced by the code are shown below.





```
Empirical CDF of 0:  0.15865525393145702
Actual CDF of 0:     0.15865525393145702

Empirical CDF of 1:  0.64658332281244
Actual CDF of 1:     0.5

Empirical CDF of 2:  0.960132415357569
Actual CDF of 2:     0.841344746068543

Empirical CDF of 3:  0.9991210079325625
Actual CDF of 3:     0.9772498680518207

Empirical CDF of 4:  0.9999966727390528
Actual CDF of 4:     0.9986501019683699
```

As you can see, the empirical and analytical CDFs of the empirical and analytical normal distributions are actually very similar to each other. This happens because as we generate more and more samples, the Central Limit Theorem holds. Therefore, we can say that the empirically and analytically obtained CDFs of the random variables match.

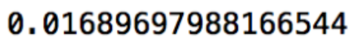
Part (b)

To complete this problem, I wrote a simple print statement utilizes the *norm_ dist_ cdf()* helper function:

```
    print(norm_dist_cdf(72, 256, 80) - norm_dist_cdf(72, 256, 66))
```

This function calculates the analytical or actual CDF according to a normal distribution, with the mean, standard deviation and x-value given.

For this problem in particular, I passed in a mean of 72, standard deviation of 16 (*i.e. variance of 256*), and x-values of 80 and 66 to the function. It then subtracts the two values to show the difference between the two analytical answers, which produces a value along the lines of the following:



```
0.01689697988166544
```