ΣΥΣΤΗΜΑΤΑ ΑΝΑΜΟΝΗΣ

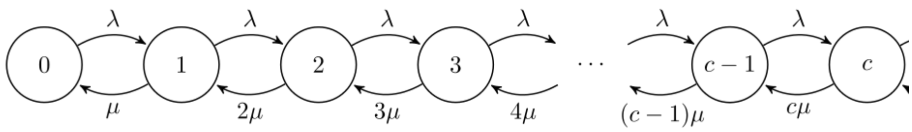
4η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ

Γεωργία Μπουσμπουκέα- el19059

<u>Ανάλυση και σχεδιασμός τηλεφωνικού κέντρου</u>

1.

Το διάγραμμα μεταβάσεων:



Οι πιθανότητες κατάστασης του συστήματος:

$$P_1 = \frac{\lambda}{\mu} \cdot P_0, \quad P_2 = \frac{\lambda}{2 \cdot \mu} \cdot P_1 = \frac{\lambda^2}{2 \cdot \mu^2} P_{0, \ldots}$$

$$\Rightarrow P_k = \frac{\lambda^k}{k! \cdot \mu^k} \cdot P_0 \;\Rightarrow P_k = \frac{p^k}{k!} P_0$$

Όμως οι πιθανότητες πρέπει να αθροίζονται στο 1, οπότε:

$$\sum_{k=0}^{k=c} P_k = 1 \Rightarrow P_0 + \sum_{k=1}^{k=c} \frac{p^k}{k!} \cdot P_0 = 1 \;\Rightarrow P_0 = \frac{1}{1 + \sum_1^c \frac{p^k}{k!}} = \frac{1}{\sum_0^c \frac{p^k}{k!}}$$

Η πιθανότητα να απορριφθεί ένα φορτίο είναι η πιθανότητα να φτάσει στην μέγιστη χωρητικότητά του, δηλαδή:

$$P_{blocking} = P_c = \frac{p^c}{c!} \cdot \sum_0^c \frac{p^k}{k!} = B(p,\, c)\; (Erlang - B\; Formula)$$

Άρα ο μέσος ρυθμός απωλειών του συστήματος είναι $\lambda \cdot P_{blocking}$

3.

Στην περίπτωση του erlang_factorial παρατηρούμε το αποτέλεσμα NaN, που οφείλεται σε stack overflow. Κάτι τέτοιο δεν συμβαίνει στην erlang_iterative, οπότε υπολογίζεται κανονικά η τιμή.

```
Calculation with erlang_factorial
NaN
0.024524
```

```
Calculation with erlang_iterative
0.024524
0.024524
```

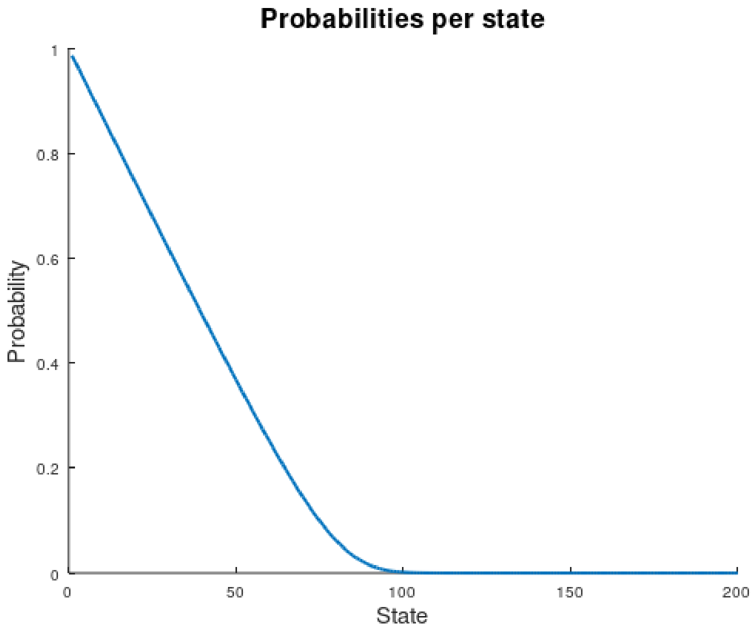
4.

a)

Χρησιμοποιώντας το μοντέλο του πιο απαιτητικού χρήστη για όλους τους εργαζόμενους, το συνολικό προσφερόμενο φορτίο είναι $p = 200 \cdot \frac{23}{60} = 76.667\ Erlangs$.

b)

Το προκύπτον διάγραμμα:



c)

Ύστερα από υπολογισμό προκύπτει πως η πιθανότητα απόρριψης κλήσης γίνεται μικρότερη του 1% όταν έχουμε τουλάχιστον 94 γραμμές.

Κώδικας:

```
addpath(pwd);
function res = erlang_factorial(p,c)
    factor = (p^c)/factorial(c);
    denom = 0;
    for i= 0:c
        denom += (p^i)/factorial(i);
    end
    res = factor/denom;
endfunction
display(erlang_factorial(50,30)); %the result is indeed the same
display(erlangb(50,30));

display("Calculation with erlang_factorial");
display(erlang_factorial(1024,1024));
display(erlangb(1024,1024));
```

```
addpath(pwd);
function res = erlang_iterative(p,n)
    i = 0;
    res = 1;
    for i=0:n
        res = p * res/(p*res + i);
    end
endfunction

display(erlang_iterative(50,30)); %the result is indeed the same
display(erlangb(50,30));

display("Calculation with erlang_iterative");
display(erlang_iterative(1024,1024));
display(erlangb(1024,1024));
```

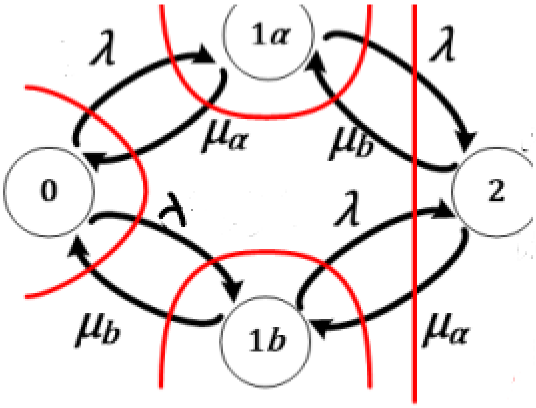
```
%4b
p = 200*23/60;
c = 1:200;
for lines=1:200  %the capacity of the system for each case
    block(i) = erlang_iterative(p,lines);
endfor
figure(1);
hold on;
title("Probabilities per state", "fontsize", 17)
xlabel("State", "fontsize", 15)
ylabel("Probability", "fontsize", 15)
plot(c, block, "linewidth", 1.6);


%4c
P=1; %starting from 0 lines, the probability to lose a client is 1
lines = 0;
while P>0.01
    P = erlang_iterative(p,lines);
    lines++;
endwhile
display(lines);
```

<u>Σύστημα εξυπηρέτησης με δύο ανόμοιους πελάτες</u>

1.

Το διάγραμμα μεταβάσεων:



(όπου μα=μ1, μβ= μ2)

$$\lambda \cdot P_0 = \mu_1 \cdot P_{1\alpha} + \mu_2 \cdot P_{1\beta}$$

$$(\mu_1 + \lambda) \cdot P_{1\alpha} = \lambda \cdot P_0 + \mu_2 \cdot P_2$$

$$(\mu_2 + \lambda) \cdot P_{1\beta} = \mu_1 \cdot P_2$$

$$(\mu_2 + \mu_1) \cdot P_2 = \lambda \cdot P_{1\alpha} + \lambda \cdot P_{1\beta}$$

Οπότε προκύπτει:

$$P_{1\alpha} = 0.86 \cdot P_0$$

$$P_{1\beta} = 0.78 \cdot P_0$$

$$P_2 = 1.37 \cdot P_0$$

Όμως :

$$\sum_{k=0}^{2} P_k = 1 \Rightarrow P_0 = 0.25 \Rightarrow P_{1\alpha} = 0.21, P_{1\beta} = 0.19, P_2 = 0.34$$

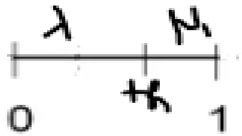
Όπως έχουμε πει, $P_{blocking} = P_2 = 0.34$

$$Μεσος\ αριθμός\ πελατών = E[n(t)] = \sum_{k=0}^{2} k \cdot P_k = 1.08\ πελατες$$

2.

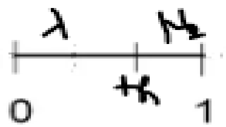
Για τον υπολογισμό των thresholds χρησιμοποιούμε τις παρακάτω συμβάσεις:

- Για την κατάσταση $\mathbf{P_{1\alpha}}$:



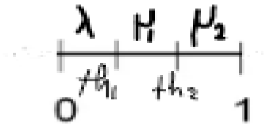
Οπότε $threshold-1a=\frac{\lambda}{\lambda+\mu_1}$

- Για την κατάσταση $\mathbf{P_{1\beta}}$:



Οπότε $threshold-1\beta=\frac{\lambda}{\lambda+\mu_2}$

- Για την κατάσταση P_2:



Οπότε $threshold-2-first=\frac{\lambda}{\lambda+\mu_1+\mu_2}$ και $threshold-2-second=\frac{\lambda+\mu_1}{\lambda+\mu_1+\mu_2}$

Στην προσομοίωση υπολογίζουμε κάθε 1000 επαναλήψεις τον μέσο αριθμό πελατών του συστήματος εκείνη την χρονική στιγμή. Θεωρούμε ότι φτάσαμε στην σύγκλιση όταν δύο τέτοιοι διαδοχικοί μέσοι όροι διαφέρουν λιγότερο από 0.00001.

Οι πιθανότητες που προκύπτουν είναι:

```
Command Window
0.2461
0.2134
0.2001
0.3403
```

Πράγματι είναι πολύ κοντά στις θεωρητικά υπολογιζόμενες. Η μικρή διαφορά οφείλεται στο κριτήριο σύγκλισης της προσομοίωσης, το οποίο το έχουμε ορίσει εμείς προσεγγιστικά.

Κώδικας:

```
clc;
clear all;
close all;

lambda = 1;
m1 = 0.8;
m2 = 0.4;

threshold_1a = lambda/(lambda+m1);
threshold_1b = lambda/(lambda+m2);
threshold_2_first = lambda/(lambda+m1+m2);
threshold_2_second = (lambda+m1)/(lambda+m1+m2);

current_state = 0;
arrivals = zeros(1,4);
total_arrivals = 0;
maximum_state_capacity = 2;
previous_mean_clients = 0;
delay_counter = 0;
time = 0;

while 1 > 0
  time = time + 1;

  if mod(time,1000) == 0
    for i=1:1:4
      P(i) = arrivals(i)/total_arrivals;
    endfor

    delay_counter = delay_counter + 1;

    mean_clients = 0*P(1) + 1*P(2) + 1*P(3) + 2*P(4);

    delay_table(delay_counter) = mean_clients;

```

```
    if abs(mean_clients - previous_mean_clients) < 0.00001
      break;
    endif
    previous_mean_clients = mean_clients;
  endif

  random_number = rand(1);

  if current_state == 0
      current_state = 1;
      arrivals(1) = arrivals(1) + 1;
      total_arrivals = total_arrivals + 1;
  elseif current_state == 1
    if random_number < threshold_1a
      current_state = 3;
      arrivals(2) = arrivals(2) + 1;
      total_arrivals = total_arrivals + 1;
    else
      current_state = 0;
    endif
  elseif current_state == 2
    if random_number < threshold_1b
      current_state = 3;
      arrivals(3) = arrivals(3) + 1;
      total_arrivals = total_arrivals + 1;
    else
      current_state = 0;
    endif
  else
      if random_number < threshold_2_first
        arrivals(4) = arrivals(4) + 1;
        total_arrivals = total_arrivals + 1;
      elseif random_number < threshold_2_second
        current_state = 2;
      else
        current_state = 1;
      endif
    endif

endwhile

display(P(1));
display(P(2));
display(P(3));
display(P(4));
```