

Міністерство освіти і науки України

Національний технічний університет України „КПІ імені Ігоря Сікорського ”

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

**Практична робота №3**

З дисципліни «Моделювання систем»

ПОБУДОВА ІМІТАЦІЙНОЇ МОДЕЛІ СИСТЕМИ З ВИКОРИСТАННЯМ ФОРМАЛІЗМУ МОДЕЛІ МАСОВОГО ОБСЛУГОВУВАННЯ

|  |  |
| --- | --- |
| Перевірила:  Стеценко Інна Вячеславівна | Виконав:  Група ІТ-94  Котяй Данило |

Київ 2022

**Завдання до практичної роботи**

1. Реалізувати універсальний алгоритм імітації моделі масового обслуговування з багатоканальним обслуговуванням, з вибором маршруту за пріоритетом або за заданою ймовірністю. **30 балів.**
2. Для наступного тексту задачі скласти формалізовану модель масового обслуговування та реалізувати її з використанням побудованого універсального алгоритму (**30 балів**):

У банку для автомобілістів є два віконця, кожне з яких обслуговується одним касиром і має окрему під'їзну смугу. Обидві смуги розташовані поруч. З попередніх спостережень відомо, що інтервали часу між прибуттям клієнтів у годину пік розподілені експоненційно з математичним очікуванням, рівним 0,5 од. часу. Через те, що банк буває переобтяжений тільки в годину пік, то аналізується тільки цей період. Тривалість обслуговування в обох касирів однакова і розподілена експоненційно з математичним очікуванням, рівним 0,3 од. часу. Відомо також, що при рівній довжині черг, а також при відсутності черг, клієнти віддають перевагу першій смузі. В усіх інших випадках клієнти вибирають більш коротку чергу. Після того, як клієнт в'їхав у банк, він не може залишити його, доки не буде обслугований. Проте він може перемінити чергу, якщо стоїть останнім і різниця в довжині черг при цьому складає не менше двох автомобілів. Через обмежене місце на кожній смузі може знаходитися не більш трьох автомобілів. У банку, таким чином, не може знаходитися більш восьми автомобілів, включаючи автомобілі двох клієнтів, що обслуговуються в поточний момент касиром. Якщо місце перед банком заповнено до границі, то клієнт, що прибув, вважається втраченим, тому що він відразу ж виїжджає. Початкові умови такі: 1) обидва касири зайняті, тривалість обслуговування для кожного касира нормально розподілена з математичним очікуванням, рівним 1 од. часу, і середньоквадратичним відхиленням, рівним 0,3 од. часу; 2) прибуття першого клієнта заплановано на момент часу 0,1 од. часу; 3) у кожній черзі очікують по два автомобіля.

Визначити такі величини: 1) середнє завантаження кожного касира;

2) середнє число клієнтів у банку; 3) середній інтервал часу між від'їздами клієнтів від вікон; 4) середній час перебування клієнта в банку;

5) середнє число клієнтів у кожній черзі; 6) відсоток клієнтів, яким відмовлено в обслуговуванні; 7) число змін під'їзних смуг.

1. Для наступного тексту задачі скласти формалізовану модель масового обслуговування та реалізувати її з використанням побудованого універсального алгоритму (**40 балів**):

У лікарню поступають хворі таких трьох типів: 1) хворі, що пройшли попереднє обстеження і направлені на лікування; 2) хворі, що

бажають потрапити в лікарню, але не пройшли повністю попереднє обстеження; 3) хворі, які тільки що поступили на попереднє обстеження. Чисельні характеристики типів хворих наведені в таблиці:

|  |  |  |
| --- | --- | --- |
| *Тип хворого* | *Відносна частота* | *Середній час реєстрації, хв* |
| 1 | 0,5 | 15 |
| 2 | 0,1 | 40 |
| 3 | 0,4 | 30 |

При надходженні в приймальне відділення хворий стає в чергу, якщо обидва чергових лікарі зайняті. Лікар, який звільнився, вибирає в першу чергу тих хворих, що вже пройшли попереднє обстеження. Після заповнення різноманітних форм у приймальне відділення хворі 1 типу ідуть прямо в палату, а хворі типів 2 і 3 направляються в лабораторію. Троє супровідних розводять хворих по палатах. Хворим не дозволяється направлятися в палату без супровідного. Якщо всі супровідні зайняті, хворі очікують їхнього звільнення в приймальному відділенні. Як тільки хворий доставлений у палату, він вважається таким, що завершив процес прийому до лікарні.

Хворі, що спрямовуються в лабораторію, не потребують супроводу. Після прибуття в лабораторію хворі стають у чергу в реєстратуру. Після реєстрації вони ідуть у кімнату очікування, де чекають виклику до одного з двох лаборантів. Після здачі аналізів хворі або повертаються в приймальне відділення (якщо їх приймають у лікарню), або залишають лікарню (якщо їм було призначено тільки попереднє обстеження). Після повернення в приймальне відділення хворий, що здав аналізи, розглядається як хворий типу 1.

У наступній таблиці приводяться дані по тривалості дій (хв):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Величина* | | | *Розподіл* | |
| Час між прибуттями відділення | в | приймальне | Експоненціальний математичним сподіванням 15 | з |
| Час слідування в палату | | | Рівномірне від 3 до 8 | |
| Час слідування з приймального відділення в лабораторію або з лабораторії в приймальне відділення | | | Рівномірне від 2 до 5 | |
| Час обслуговування лабораторії | в | реєстратуру | Ерланга з математичним сподіванням 4,5 і *k*=3 | |
| Час проведення аналізу в лабораторії | | | Ерланга з математичним сподіванням 4 і *k*=2 | |

Визначити час, проведений хворим у системі, тобто інтервал часу, починаючи з надходження і закінчуючи доставкою в палату (для хворих типу 1 і 2) або виходом із лабораторії (для хворих типу 3). Визначити також інтервал між прибуттями хворих у лабораторію.

**Хід роботи**

1. Реалізувати універсальний алгоритм імітації моделі масового обслуговування з багатоканальним обслуговуванням, з вибором маршруту за пріоритетом або за заданою ймовірністю.

В цьому завданні нам потрібно додати функціональність вибору маршруту за приорітетом або за заданою ймовірністю.

Лістинг коду за посиланням на GitHub:

<https://github.com/Lazyprog121/Model.Lab3>

def get\_nextElement(self):

isPriorityDefault = self.priorities == Element.defaultArray

isProbabilityDefault = self.probability == Element.defaultArray

isError = not isProbabilityDefault and not isPriorityDefault

if isError:

raise Exception('error in nextElement')

elif isPriorityDefault:

return np.random.choice(a=self.nextElement, p=self.probability)

elif isProbabilityDefault:

return self.get\_from\_priority()

elif isProbabilityDefault and isPriorityDefault:

return self.nextElement[0]

def get\_from\_priority(self):

priorities = self.priorities.copy()

min\_queue = float('inf')

min\_queue\_index = 0

for p in range(len(priorities)):

max\_priority = max(priorities)

if max\_priority == -1:

break

max\_priority\_index = priorities.index(max\_priority)

if 0 in self.nextElement[max\_priority\_index].state:

return self.nextElement[max\_priority\_index]

else:

if self.nextElement[max\_priority\_index].queue < min\_queue:

min\_queue = self.nextElement[max\_priority\_index].queue

min\_queue\_index = self.nextElement.index(self.nextElement[max\_priority\_index])

priorities[max\_priority\_index] = -1

return self.nextElement[min\_queue\_index]

Реалізовуємо багатоканальну модель, в якій задаємо ймовірності і послідовності наступних елементів. В цьому випадку ймовірності будуть 0.7 та 0.3

p1 = Process(maxqueue=3, delayMean=5, distribution='exp')

p1.probability = [0.7, 0.3]

c1 = Create(delayMean=5, name='Probability - Creator', distribution='exp')

p2 = Process(maxqueue=3, delayMean=5, distribution='exp')

p3 = Process(maxqueue=3, delayMean=5, distribution='exp')

c1.nextElement = [p1]

p1.nextElement = [p2, p3]

elements = [c1, p1, p2, p3]

model = Model(elements)

model.simulate(1000)

Перевіримо нашу модель.

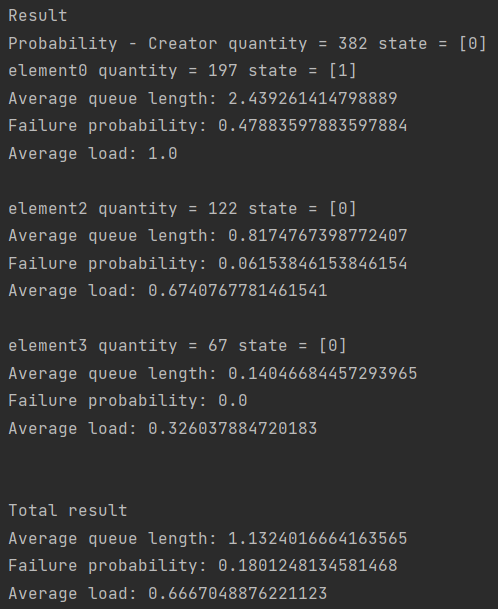


Рисунок 1 – модель з ймовірностями 0.7 та 0.3

Можемо зробити висновок, що ймовірності працюють, оскільки останній процес зміг обробити менше елементів, ніж попередній.

Тепер змінимо ймовірності на 0.3 та 0.7

p1.probability = [0.3, 0.7]

Запустимо модель.

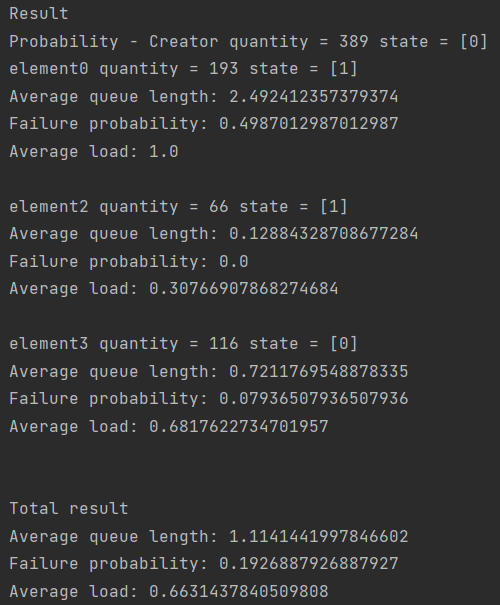


Рисунок 2 – модель з ймовірностями 0.3 та 0.7

Отже, можемо ще раз підтвердити, що модель працює, оскільки цього разу третій процес обробив більше елементів, ніж попередній.

2. Для наступного тексту задачі скласти формалізовану модель масового обслуговування та реалізувати її з використанням побудованого універсального алгоритму.

c1 = BankCreate(delayMean=0.5, name='Bank Creator', distribution='exp')

p1 = BankProcess(maxqueue=3, delayMean=0.3, name='Casa 1', distribution='exp')

p2 = BankProcess(maxqueue=3, delayMean=0.3, name='Casa 2', distribution='exp')

c1.nextElement = [p1, p2]

p1.state[0] = 1

p2.state[0] = 1

p1.tnext[0] = utility.normal(1, 0.3)

p2.tnext[0] = utility.normal(1, 0.3)

c1.tnext[0] = 0.1

p1.queue = 2

p2.queue = 2

element\_list = [c1, p1, p2]

bank = BankModel(element\_list, items\_to\_balance=[p1, p2])

bank.simulate(1000)

Перевіримо нашу модель.

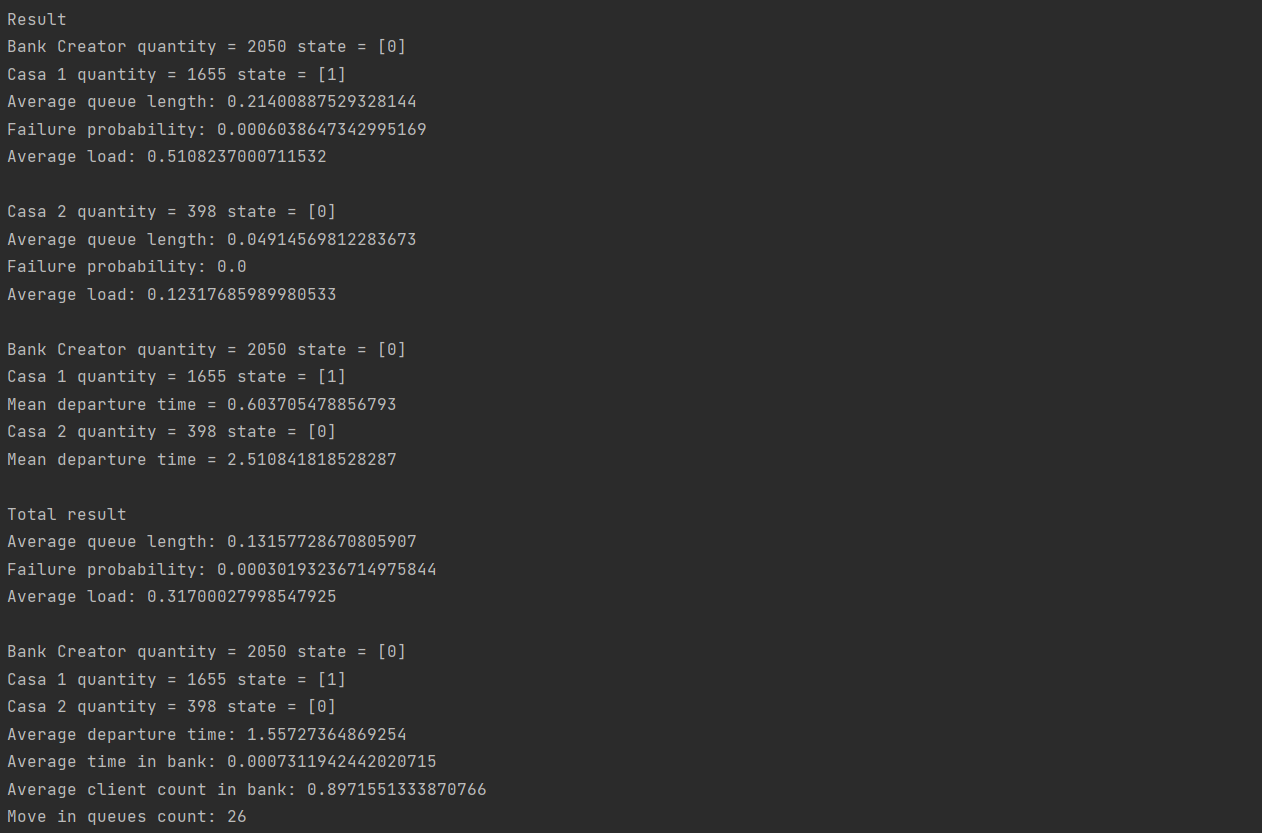
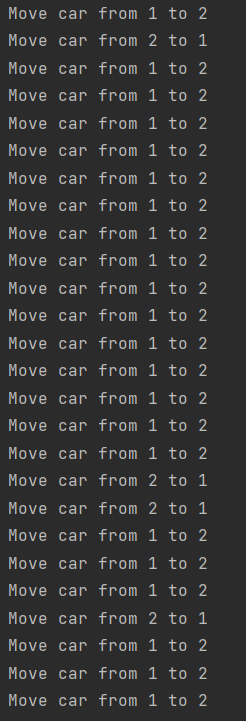


Рисунок 3 – модель масового обслуговування банк

Проаналізуємо нашу модель:

1) середнє завантаження кожного касира: 0.51 (касир 1), 0.12 (касир 2)

2) середнє число клієнтів у банку: 0.89

3) середній інтервал часу між від'їздами клієнтів від вікон: 1.55

4) середній час перебування клієнта в банку: 0.0007

5) середнє число клієнтів у кожній черзі: 0.21 (черга 1), 0.05 (черга 2)

6) відсоток клієнтів, яким відмовлено в обслуговуванні: 0.087%

7) число змін під'їзних смуг: 26

3. Для наступного тексту задачі скласти формалізовану модель масового обслуговування та реалізувати її з використанням побудованого універсального алгоритму.

c1 = HospitalCreate(delayMean=15.0, name='Hospital CREATOR', distribution='exp')

registration = HospitalProcess(

maxqueue=100,

threads=2,

name='Registration',

distribution='exp')

palata\_way = HospitalProcess(

maxqueue=100,

delayMean=3.0,

delayDev=8,

threads=3,

name='Way to the palata',

distribution='uniform')

lab\_way = HospitalProcess(

maxqueue=0,

delayMean=2.0,

delayDev=5,

threads=10,

name='Way to the lab registration',

distribution='uniform')

lab\_registration = HospitalProcess(

maxqueue=100,

delayMean=4.5,

delayDev=3,

threads=1,

name='Registration to lab',

distribution='erlang')

watching = HospitalProcess(

maxqueue=100,

delayMean=4.0,

delayDev=2,

threads=1,

name='Docktor checking',

distribution='erlang')

registration\_way = HospitalProcess(

maxqueue=0,

delayMean=2.0,

delayDev=5,

threads=10,

name='Way to the registration',

distribution='uniform')

exit1 = HospitalExit(name='EXIT1')

exit2 = HospitalExit(name='EXIT2')

c1.nextElement = [registration]

registration.nextElement = [palata\_way, lab\_way]

palata\_way.nextElement = [exit1]

lab\_way.nextElement = [lab\_registration]

lab\_registration.nextElement = [watching]

watching.nextElement = [exit2, registration\_way]

registration\_way.nextElement = [registration]

registration.priority\_patient\_types = [1]

registration.patient\_path = [[1], [2, 3]]

watching.patient\_path = [[3], [2]]

elements = [c1, registration, palata\_way, lab\_way, lab\_registration, watching, registration\_way, exit1, exit2]

model = HospitalModal(elements)

model.simulate(10000)

Перевіримо нашу модель.

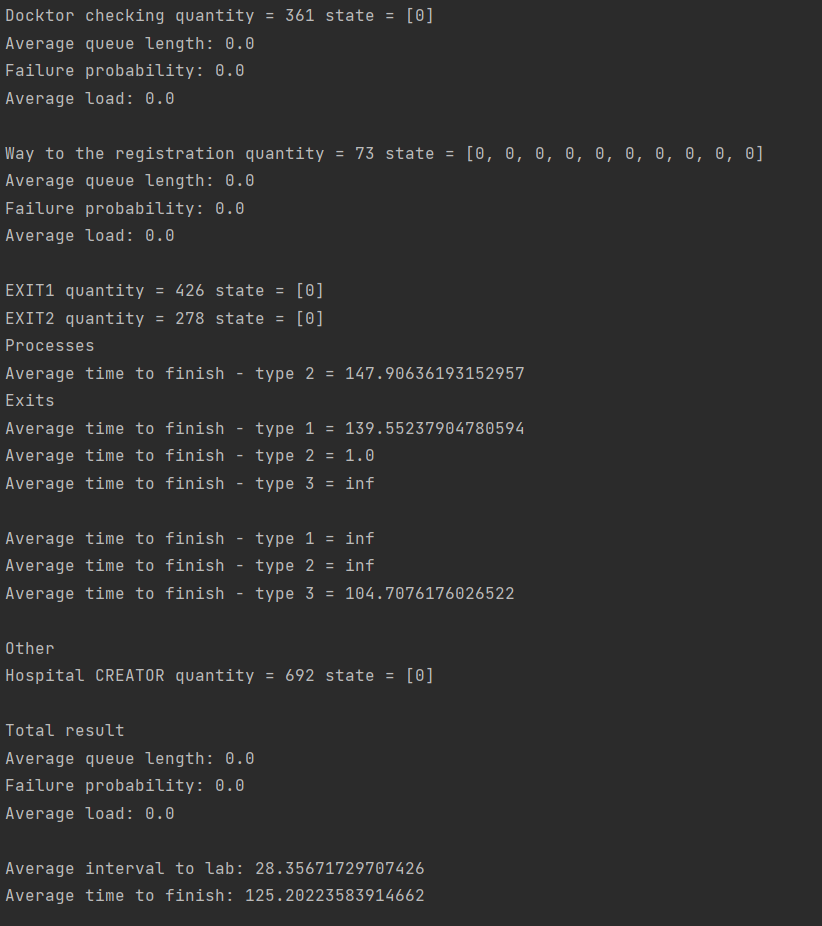


Рисунок 4 – модель масового обслуговування гостіталь

Проаналізуємо нашу модель:

1) час, проведений хворим у системі, тобто інтервал часу,

починаючи з надходження і закінчуючи доставкою в палату (для хворих

типу 1 і 2): 147.9

2) виходом із лабораторії (для хворих типу 3): 104.7

3) інтервал між прибуттями хворих у лабораторію: 28.3

**Висновок**

В цій лабораторній роботі ми реалізувати універсальний алгоритм імітації моделі масового обслуговування з багатоканальним обслуговуванням, з вибором маршруту за пріоритетом або за заданою ймовірністю. Використали цей універсальний алгоритм у системах масового обслуговування, таких як банк та госпіталь.