**Домашняя работа по предмету «Теория массового обслуживания»**

**Студент: Королев Дмитрий**

**Группа: БПИ-173**

**Ссылка:** <https://github.com/DeagleGross/Queueing-Theory-Simulation>

*Программный код:* [*https://github.com/DeagleGross/Queueing-Theory-Simulation/blob/master/Simulation.ipynb*](https://github.com/DeagleGross/Queueing-Theory-Simulation/blob/master/Simulation.ipynb)

Домашняя работа выложена на ресурс github.com. Вы можете просмотреть всю реализацию с небольшими комментариями там.

**Задание**

Для домашнего задания я придумал следующую задачу:

*В цехе находятся 2 станка.*

*В цех поступают детали в среднем через каждые 35 минут (здесь и далее подразумевается, что все времена распределены экспоненциально).*

*Обработка детали производится на одном из двух станков.*

*Первый обрабатывает деталь в среднем за 30 минут, при этом отвергает 10 % деталей, считая их бракованными.*

*Второй обрабатывает деталь в среднем за 50 минут, при этом отвергает 20 % деталей, считая их бракованными.*

*Если деталь была забракована в первый раз, то ее попробуют обработать еще раз после всех деталей, стоящих в очереди (будь то в очереди на первичную или вторичную обработку)*

*Детали, которые были отвергнуты 2 раза, считаются отходами и не обслуживаются цехом.*

*Детали, которые были отвергнуты на любом станке передаются на повторную обработку на второй станок.*

*Моделирование проводится на 500 деталей.*

*Каждый из станков имеет цену обработки (или попытки обработки) детали.*

*Для первого станка цена обработки составляет 5$. Для второго - 2$.*

*Цена готового изделия составляет 12 долларов.*

*Предпринимателю, который владеет цехом, поступило предложение об обмене второго станка. Новый станок имеет цену обработки в 3 $. Время обработки - 40 минут в среднем. Вероятность браковки - 10%.*

*Задача: выяснить какой из станков (старый 2-й или новый) будет более выгодный?*

*а) прибыль*

*б) прибыль \ сек*

Я принял решение реализовать симуляцию на языке Python.

**Тонкости реализации**

В стандартной библиотеке языка Python я нашел функцию, которая возвращает экспоненциально-распределенные величины в зависимости от переданной «лямбды» (функция random.expovariate(lambda)). Я задал такие начальные значения исходя из условия задачи:

arrival\_rate = 35 *# min*

first\_machine\_speed = 30 *# min*

second\_machine\_speed = 50 *# min*

third\_machine\_speed = 40 *# min*

И запускал вышеописанную функцию, передавая в качестве значения

В данном случае *rate* – частота прибытия деталей в цех (в мин) или частота обработки детали машиной (в мин).

Генерация времени осуществлялась таким алгоритмом:

1. Пусть *ti* – время, когда в цех поступил *i-ая* деталь.
2. Пока в цех не поступит 500 деталей (из условия задачи) или, иначе говоря, пока *i* не достигнет 500, будем генерировать *t­i* следующим образом:

Очевидно,