# Laboratory 2: Simulation of a single server queue (Grocery centre problem)

1. Introduction and Purpose of Experiment
2. Aim and Objectives
3. Experimental Procedure

An ATM booth has a single machine to withdraw cash. Customers arrive at the ATM at random times that are from 1 to 8 minutes apart. Each Inter-arrival time has the same probability of occurrence and service times vary from 1 to 6 minutes with the respective probabilities of time taken for service shown in Table 3 below:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Service(in minutes) | 1 | 2 | 3 | 4 | 5 | 6 |
| Probability | 0.10 | 0.20 | 0.30 | 0.25 | 0.10 | 0.05 |

1. Table 3

Simulate the system for arrival of 1000 customers starting with an empty ATM queue to determine the following:

i. Average waiting time of a customer

ii. Idle time of the ATM machine

iii. Average service time

iv. Average time between arrivals

Use random numbers between 1 to 1000 to determine inter arrival time, and random numbers between 1 to 100 to determine service time.

1. Algorithms

single\_server\_queue\_model():

1. for each request in queue:

2. AT[i] = AT[i-1] + IAT[i]

3. if (AT[i] >= SE[i-1])

4. SS[i] = AT[i-1]

5. else SS[i] = SE[i-1]

6. SE[i] = SS[i] + ST[i]

7. WAIT[i] = SS[i]-AT[i]

8. IDLE[i] = AT[i] – SE[i-1] >= 0 ? AT[i] – SE[i1] : 0

1. Presentation of Results

/\*

 \* To change this license header, choose License Headers in Project Properties.

 \* To change this template file, choose Tools | Templates

 \* and open the template in the editor.

 \*/

package lab02;

import java.util.ArrayList;

import java.util.Collections;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.OptionalDouble;

import java.util.Random;

import java.util.Scanner;

import java.util.stream.Collectors;

import java.util.stream.IntStream;

/\*\*

 \*

 \* @author shadowleaf

 \*/

public class Lab02 {

    /\*\*

     \* @param args the command line arguments

     \*/

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        System.out.print("Enter the Number of Requests : ");

        Integer N = input.nextInt();

        Random rand = new Random();

        // Taking it as input

//        List<Integer> IAT = new ArrayList<>();

//        List<Integer> ST = new ArrayList<>();

//        IAT.add(0);

//        ST.add(0);

//

//        System.out.print("Enter IAT's : ");

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

//            IAT.add(input.nextInt());

//        }

//

//        System.out.print("Enter ST's : ");

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

//            ST.add(input.nextInt());

//        }

        // Inter Arrival Times range from 1 - 8 mins

        List<Integer> IAT = IntStream.range(0, N + 1).mapToObj(i -> rand.nextInt(8) + 1).collect(Collectors.toList());

        // Service Time ranges from 1 - 6 mins

        List<Integer> ST = IntStream.range(0, N + 1).mapToObj(i -> rand.nextInt(6) + 1).collect(Collectors.toList());

        Map<String, List<Integer>> SIM\_TAB = new HashMap<>();

        SIM\_TAB.put("IAT", IAT);

        SIM\_TAB.put("ST", ST);

        SIM\_TAB.put("SS", new ArrayList<>(Collections.nCopies(N + 1, 0)));

        SIM\_TAB.put("AT", new ArrayList<>(Collections.nCopies(N + 1, 0)));

        SIM\_TAB.put("SE", new ArrayList<>(Collections.nCopies(N + 1, 0)));

        SIM\_TAB.put("WAIT", new ArrayList<>(Collections.nCopies(N + 1, 0)));

        SIM\_TAB.put("IDLE", new ArrayList<>(Collections.nCopies(N + 1, 0)));

        SIM\_TAB.get("AT").set(0, 0);

        for (int i = 1; i <= N; i++) {

            SIM\_TAB.get("AT").set(i, SIM\_TAB.get("AT").get(i - 1) + SIM\_TAB.get("IAT").get(i));

            if (SIM\_TAB.get("AT").get(i) >= SIM\_TAB.get("SE").get(i - 1)) {

                SIM\_TAB.get("SS").set(i, SIM\_TAB.get("AT").get(i));

            } else {

                SIM\_TAB.get("SS").set(i, SIM\_TAB.get("SE").get(i - 1));

            }

            SIM\_TAB.get("SE").set(i, SIM\_TAB.get("SS").get(i) + SIM\_TAB.get("ST").get(i));

            SIM\_TAB.get("WAIT").set(i, SIM\_TAB.get("SS").get(i) - SIM\_TAB.get("AT").get(i));

            SIM\_TAB.get("IDLE").set(i, SIM\_TAB.get("AT").get(i) - SIM\_TAB.get("SE").get(i-1) >= 0 ? SIM\_TAB.get("AT").get(i) - SIM\_TAB.get("SE").get(i-1) : 0);

        }

        System.out.println("REQNO\tIAT\tAT\tSS\tSE\tST\tWAIT\tIDLE");

        for (int i = 1; i <= N; i++) {

            String out = i + "\t"

                    + SIM\_TAB.get("IAT").get(i) + "\t"

                    + SIM\_TAB.get("AT").get(i) + "\t"

                    + SIM\_TAB.get("SS").get(i) + "\t"

                    + SIM\_TAB.get("SE").get(i) + "\t"

                    + SIM\_TAB.get("ST").get(i) + "\t"

                    + SIM\_TAB.get("WAIT").get(i) + "\t"

                    + SIM\_TAB.get("IDLE").get(i);

            System.out.println(out);

        }

        // Avg WAIT, Avg. Ser, Avg. IAT

        OptionalDouble avgWAIT = SIM\_TAB.get("WAIT").stream().mapToDouble(a -> a).average();

        OptionalDouble avgService = SIM\_TAB.get("ST").stream().mapToDouble(a -> a).average();

        OptionalDouble avgIAT = SIM\_TAB.get("IAT").stream().mapToDouble(e -> e).average();

        System.out.println("Average WAIT : " + avgWAIT.getAsDouble());

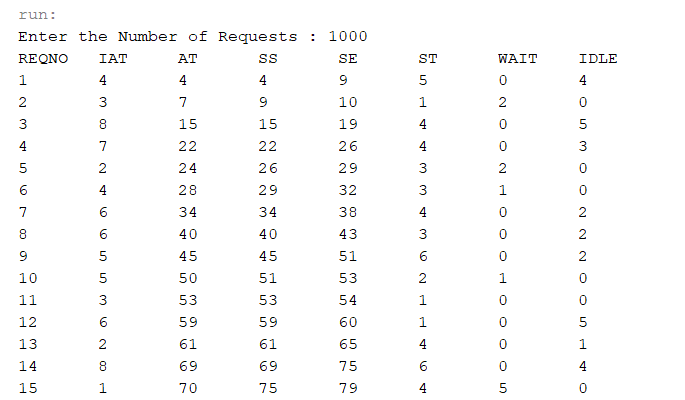
        System.out.println("Averate Service Time : " + avgService.getAsDouble());

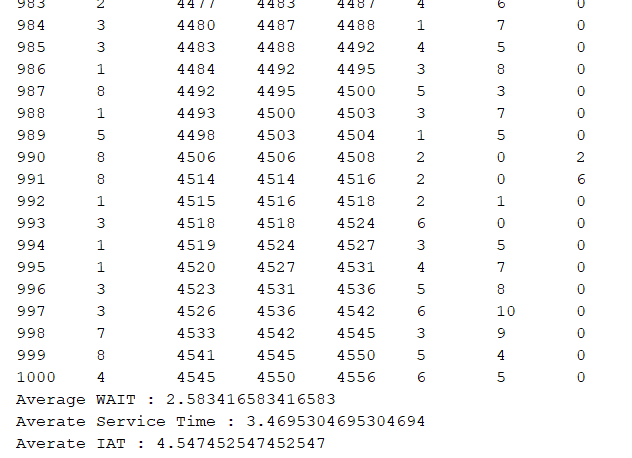
        System.out.println("Averate IAT : " + avgIAT.getAsDouble());

    }

}

1. Analysis and Discussions





1. Conclusions

Single Server Queue

This is the simplest queuing system as represented in the following figure. The central element of the system is a server, which provides service to the connected devices or items. Items request to the system to be served, if the server is idle. Then, it is served immediately, else it joins a waiting queue. After the task is completed by the server, the item departs.

