# Laboratory 3: Simulation of a two server Queue (Able Baker Problem)

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

An ATM booth has a two machine to withdraw cash. Customers arrive at the ATM at random times that are from 1 to 4 minutes apart with the respective probabilities of time taken for arrival shown in Table below. Service times of machine **A**ble vary from 2 to 5 minutes and service times of machine **Baker** vary from 3 to 6 minutes with the respective probabilities of time taken for service shown in Table below:

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. Total time in the system

**IAT Able Baker**

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
2. 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 lab03;

import java.util.Arrays;

import java.util.Random;

import java.util.Scanner;

/\*\*

 \*

 \* @author shadowleaf

 \*/

public class Lab03 {

    /\*\*

     \* @param arr

     \* @param e

     \* @param sta

     \* @param end

     \*/

    public static void fillArray(int arr[], int e, int sta, int end) {

        for (int i = sta ; i <= end ; i++) {

            arr[i] = e;

        }

    }

    public static int[] prepareProbDist(double prob[], int ass\_prob[]) {

        int[] cumulativeProb = new int[ass\_prob.length];

        int[] prob\_lookup = new int[101];

        cumulativeProb[0] = (int)(prob[0] \* 100);

        for (int i = 1 ; i < cumulativeProb.length ; i++) {

            cumulativeProb[i] = (int)(cumulativeProb[i-1] + prob[i] \* 100);

        }

        int start = 0;

        for (int i = 0 ; i < cumulativeProb.length ; i++) {

            int end = cumulativeProb[i];

            fillArray(prob\_lookup, ass\_prob[i], start, end);

            start = end;

        }

        return prob\_lookup;

    }

    /\*\*

     \*

     \* @param args

     \*/

    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();

        double[] IATPROB = {0.25, 0.40, 0.20, 0.15};

        int[] IAT\_ASS = {1, 2, 3, 4}; // IAT Associated

        int[] IAT\_LOOKUP = prepareProbDist(IATPROB, IAT\_ASS);

        double[] APROB = {0.30, 0.28, 0.25, 0.17};

        int[] AST\_ASS = {2, 3, 4, 5};

        int[] AST\_LOOKUP = prepareProbDist(APROB, AST\_ASS);

        double[] BPROB = {0.35, 0.25, 0.20, 0.20};

        int[] BST\_ASS = {3, 4, 5, 6};

        int[] BST\_LOOKUP = prepareProbDist(BPROB, BST\_ASS);

        // generate IAT, AST, BST

        int[] IAT = new int[N+1];

        int[] AST = new int[N+1];

        int[] BST = new int[N+1];

//        int[] IAT = new int[N+1];

        int[] ST = new int[N+1];

        int[] AT = new int[N+1];

        int[] ASS = new int[N+1];

        int[] ASE = new int[N+1];

        int[] BSS = new int[N+1];

        int[] BSE = new int[N+1];

        int[] TIS = new int[N+1];

        int[] WAIT = new int[N+1];

// Randomly directly generate the IAT and ST

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

//            IAT[i] = rand.nextInt(8) + 1;

//            ST[i] = rand.nextInt(6) + 1;

//        }

// Testing :

//        int[] ST = {0, 6, 1, 5, 5, 5};

//        int[] IAT = {0, 3, 2, 3, 1, 5};

        int idxAbleLastBusy = 0;

        int idxBakerLastBusy = 0;

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

            // generate random number from 0 to 100

            int randNum = rand.nextInt(101);

            // get a random IAT from the lookup table

            IAT[i] = IAT\_LOOKUP[randNum];

            AT[i] = AT[i-1] + IAT[i];

            boolean ableIsFree = ASE[idxAbleLastBusy] <= AT[i];

            boolean bakerIsFree = BSE[idxBakerLastBusy] <= AT[i];

            if (ableIsFree) {

                // get a random service time for Able

                ST[i] = AST\_LOOKUP[randNum];

                ASS[i] = AT[i]; ASE[i] = AT[i] + ST[i];

                TIS[i] = AT[i] - ASE[i]; WAIT[i] = ASS[i] - AT[i];

                idxAbleLastBusy = i;

            } else if (bakerIsFree) {

                // get a random serivce time for Baker

                ST[i] = BST\_LOOKUP[randNum];

                BSS[i] = AT[i]; BSE[i] = AT[i] + ST[i];

                TIS[i] = AT[i] - BSE[i]; WAIT[i] = BSS[i] - AT[i];

                idxBakerLastBusy = i;

            } else { // neither of them is free

                // check who gets free first

                boolean isAbleFreeFirst = ASE[idxAbleLastBusy] <= BSE[idxBakerLastBusy];

                if (isAbleFreeFirst) { // able is free first

                    ST[i] = AST\_LOOKUP[randNum];

                    ASS[i] = ASE[idxAbleLastBusy]; ASE[i] = ASE[idxAbleLastBusy] + ST[i];

                    TIS[i] = AT[i] - ASE[i]; WAIT[i] = ASS[i] - AT[i];

                    idxAbleLastBusy = i;

                } else { // baker is free first

                    ST[i] = BST\_LOOKUP[randNum];

                    BSS[i] = BSE[idxBakerLastBusy]; BSE[i] = BSE[idxBakerLastBusy] + ST[i];

                    TIS[i] = AT[i] - BSE[i]; WAIT[i] = BSS[i] - AT[i];

                    idxBakerLastBusy = i;

                }

            }

            // mat pucho ye bakwaas kahe kiye hum

            TIS[i] \*= -1;

        }

        System.out.printf("|%s\t|%s\t|%s\t|%s\t|%s\t|%s\t|%s\t|%s\t|%s\t|%s\t\n", "REQNO", "IAT", "ST", "AT", "ASS", "ASE", "BSS", "BSE", "TIS", "WAIT");

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

            System.out.printf("|%d\t|%d\t|%d\t|%d\t|%d\t|%d\t|%d\t|%d\t|%s\t|%s\t\n", i, IAT[i], ST[i], AT[i], ASS[i], ASE[i], BSS[i], BSE[i], TIS[i], WAIT[i]);

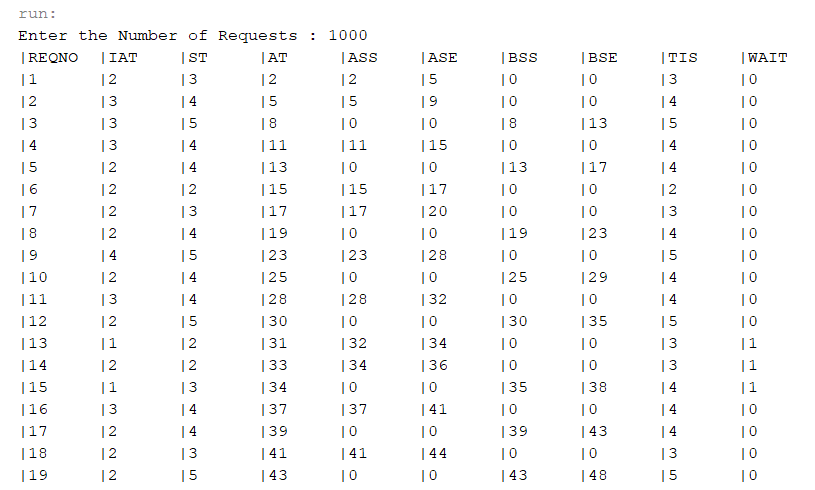
        }

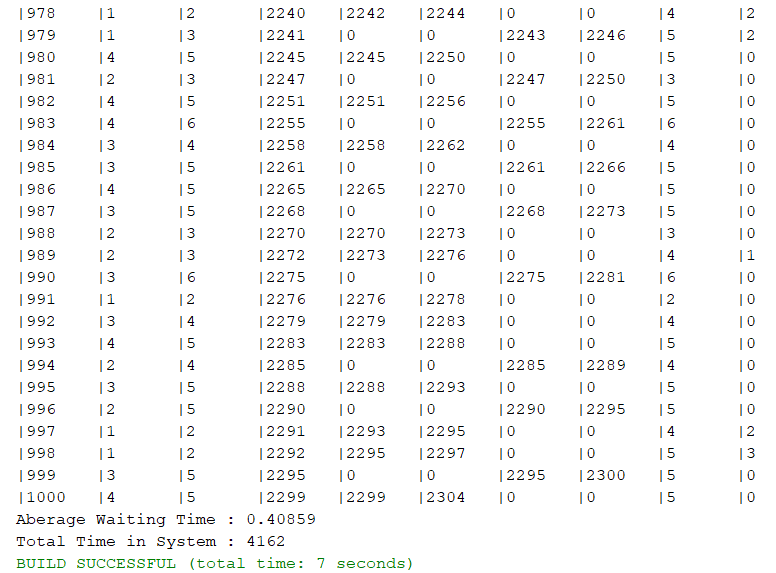
        System.out.printf("Aberage Waiting Time : %.5f\nTotal Time in System : %d\n", Arrays.stream(WAIT).mapToDouble(a -> a).average().getAsDouble(), Arrays.stream(TIS).sum());

    }

}

1. Analysis and Discussions





1. Conclusions