Chen Wang cwang784 Haodan Tan htan74

Assumption:

1. There are random passengers ranging from 1 to max 18.
2. Each passenger has a random destination from 2 to max 20.
3. Floor 1 is the ground floor and no waiting time to enter in the elevator.
4. Stopping time is 15 seconds for each floor which is fixed and irrelated to the numbers of passengers.

Summary:

We output four characteristic time results for each scenario including: 'average of total time', 'minimum of total time', 'maximum of the total time', and 'average time per passenger'. Scenario 1 has the largest 'average of total time' and 'average time per passenger', which are 158.15s and 91.52s. This meets our expectation since there is only one elevator in first situation. Scenario 2 has smaller results of 'average of total time' and 'average time per passenger' comparing to scenario 1, which are 125.05s and 70.58s. This meets our expectation since there are 2 elevators. Comparing scenario 1, more elevators should decrease the waiting time for each passenger. In the large number of repeated experiments, both scenario 1 and scenario 2 records the 'min of the total time' as 18s and 'max of total time' as 282s.

In scenario 3, we tested the 'maxFloor1' ranging from 7 to 15 and find the optimal results are 11 and 12 which have the smallest 'average of the total time' and 'average time per passenger'.
In scenario 4, we tested 6 < 'maxFloor1' < 'maxFloor2' < 16 and find the optimal results are 'maxFloor1 = 9, maxFloor2 = 15' and 'maxFloor1 = 8, maxFloor2 = 14'

Structural and Functional Parts:

We respectively built three files including main.c, elevator.c, and elevator.h:

- main.c: ‘test1()’, ‘test2()’, ‘test3()’,’test4()’ simulate the process 1000 times for each scenario. One loop for scenario 3 to test the situation for various maxFloor1 values. And double for loop to test the situation for various maxFloor1 and maxFloor2 values.
- elevator.c: Package all functions including ‘numOfpass()’ to generate the random number of passengers, ‘floorPerpass()’ to generate the random floor array, ‘computeTotalTime()’ and ‘computeAvgTime()’ to calculate the elevator running time, ‘eachTest1()’ , ‘eachTest2()’ , ‘eachTest3()’, ‘eachTest4()’ to simulate the process once for each scenario.
- elevator.h: header file to elevator.c

Haodan Tan (htan74)

```
int numOfpass(int upper, int lower);

int* floorPerpass(int totalpass);

int countDistinct(int* floor, int totalPass);

int countHighestFloor(int* floor, int totalPass);

 double* eachTest();

 void test1();

 double* eachTest2();

 void test2();
```

Chen Wang (cwang784)

```
int computeTotalTime(int highestFloor, int distinctFloor);

double computeAvgTime(double totalTime, int passenger);

void swap(int* xp, int* yp);

void sortFloor(int* floor, int passenger);

double waitTotal(int* floor, int passenger);

double* eachTest();

int* floorPerpass1(int totalpass, int maxFloor1);

int* floorPerpass2(int totalpass, int maxFloor1);

double* eachTest3(int maxFloor);

void test3(int maxFloor);

double* eachTest4(int maxFloor1, int maxFloor2);
```

For scenario 1:

1. We generate a random number of total passengers between 1 and 18.
2. We generate a random floor array to represent the destination for each passenger between 2 and 20 according to the random number of total passengers.

For scenario 2:

1. We generate a random number of total passengers between 1 and 18.
2. We generate a random number of elevator1 passengers between 0 and total passengers and assign the remaining passengers to elevator 2.
3. We generate two random floor arrays to represent the destination for each passenger between 2 and 20 according to the random number of elevator1 and elevator2 passengers.

For scenario 3 and scenario 4:

1. We generate a random number of total passengers between 1 and 18.
2. We generate a random floor array to represent the destination for each passenger between 2 and 20 according to the random number of total passengers.
3. We assign each element in the random floor array to a random elevator.

Summary:

We use two approaches to do the random process. Both works well for their assigned scenario and meet the probability expectation. In future work, we can use the approach of scenario 3 and 4 for both scenario 1 and 2 which can save some work and get the same probability expectation. But approach of scenario 2 cannot satisfy scenario 3 and 4 cause the probability expectation has been changed.

Calculation:

- The elevator running time:

$$t1 = 3 * (highest\ destination\ floor\ - 1)$$

$$t2 = 15 * total\ distinct\ floor\ numbers\ that\ have\ passengers\ left$$

$$t_{totalTime} = t_1 + t_2$$

- The total waiting time for per passenger:

$$t_{eachOne} = totaltime(denestion)$$

$$t_{waitTotal} = \sum_1^n t_{eachone}$$

- The average waiting time for per passenger:

$$t_{perPassengerTime} = \frac{t_{waitTotal}}{n}$$

Summary:

We think our approach works well. But we can do the calculation before we start coding in the future.