ECM2433 report

Jakub Davison

April 2022

1 Description of design decisions

The first decision I made when developing this program was the queue structure. I chose to go with two queues one for the left side and one for the right instead of having one large queue with a char or int variable determining if the car is coming from the left or right. The reasoning behind this was that it would save space as the number of entries would be the same in both options, however, with only having one queue i would have to store another variable. Not only did this help in space but also in run time. This is because if I only had one queue I would have to run an extra condition to determine whether a car is on the left or right. Both of the queues are designed to be a linked list with every entry containing: time spent in the queue as an integer (variable name waitTime), a char variable (finished) which allowed me to determine whether a car has passed through the traffic light or not, and a pointer to the next structure in the linked list. I chose a char for the finished variable as it takes up the least amount of space.

The next structure I used was for results. In this structure there is a variable for every necessary result I needed to obtain from this program. Every variable is an integer except for the average wait as I decided a more accurate result may needed to be displayed for extra accuracy. This was an assumption I made as there was no specification to what data type it should be in the instructions.

As I have provided a README file that clearly states that the user has to input 4 positive integers separated by a space, I did not program the main function defensively which would protect the user if the input they provide would be incorrect.

A further assumption I made was that the time taken for a car to pass is 0. This is because since the two lights cannot both be red at the same time, the car's travel time must be 0 as otherwise the car from the other direction could collide.

Instead of a boolean for the traffic light, I used an integer value. This is because the run time is quicker than a char and since I do not have to store multiple values of this variable I decided for the more optimum run time.

There are 3 structures created for each queue. The head (variable name headL/headR), temporary holder(variable name tempL/tempR), and a holder for the node that should be changed to finished (variable name finishedHolder /finishedHolderR). The use of a head node for each linked list is self-explanatory. The temporary holder is used to traverse the linked list and change any necessary values (mainly waitTime). The finished holder is used to select whichever node is next in queue and further on it's finished status will be changed.

The logic starts by adding to the wait time of every car that hasn't passed or which isn't the next to pass. It does this on both queues with the exception that the holder for finished is only selected from one of the queues depending on which light is on. As it was not specified I selected that the cars from the left side get the green light first. After this, as long as the time doesn't equal 0, a random number is selected to determine whether a car is added to one side, and then another random number is selected to determine whether a car is added to the other side. I use the rate which is supplied by the user as a maximum random number. This allows me to use it as rates as specified because as long as the random number is not 0 it will add a car to the respective queue. This means that the higher the rate, the more cars will be added. After this the finishedHolder is finished. Then the time variable gets amended. Once the time gets to 0 the light switches.

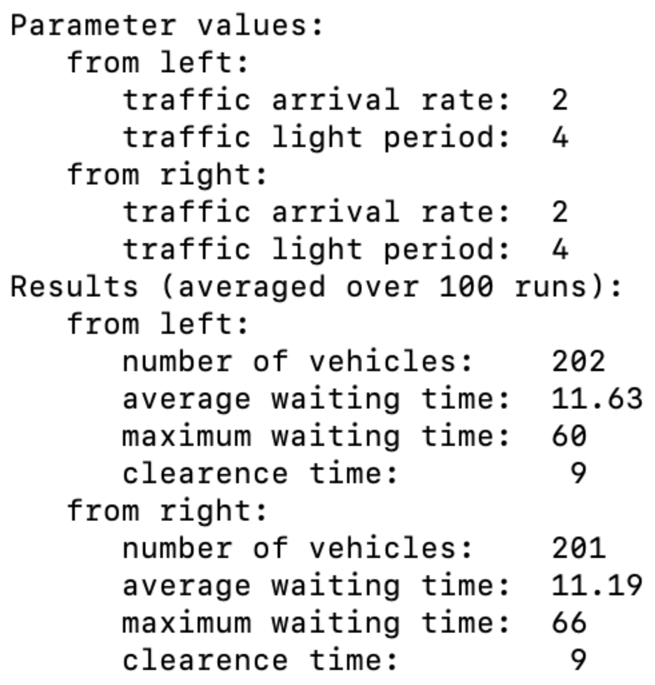
Once the 500 iterations are over, no new cars will be added but the same logic is used to continue allowing cars to pass. A right adn left variables are created here to count whether a side is empty. the time taken for it to be cleared (variable name clearTimeL/clearTimeR) is amended only if there are still cars that have not passed yet.

After this I used a while loop to go through both queues to access the data and do any necessary calculations to determine the variables which are the returned as a results structure.

2 Experiment description

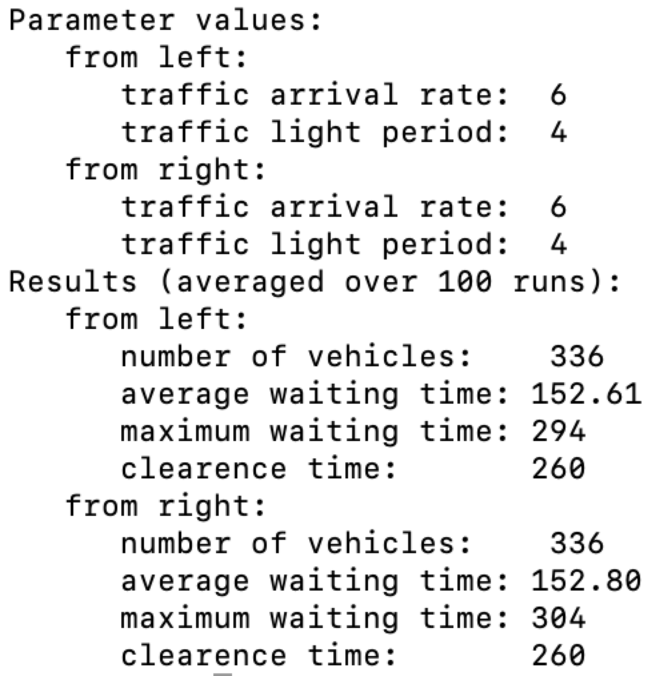
I have decided to maintain the length of time the light stays on for each side and change the rate at which cars spawn. I also decide to keep the rate at which vehicles arrive in each direction the same. I chose my input values in the first run to be 2 2 4 4. Where the rate at which cars come in both directions is 2 and the time a light is on in each direction is 4. In the second run my input was 6 6 4 4. The results can be seen below. It is the example output from both runs:

Test 1 results:



```
Parameter values:
   from left:
      traffic arrival rate:  2
      traffic light period:  4
   from right:
      traffic arrival rate:  2
      traffic light period:  4
Results (averaged over 100 runs):
   from left:
      number of vehicles:    202
      average waiting time:  11.63
      maximum waiting time:  60
      clearence time:         9
   from right:
      number of vehicles:    201
      average waiting time:  11.19
      maximum waiting time:  66
      clearence time:         9
```

Test 2 results:



```
Parameter values:
   from left:
      traffic arrival rate:  6
      traffic light period:  4
   from right:
      traffic arrival rate:  6
      traffic light period:  4
Results (averaged over 100 runs):
   from left:
      number of vehicles:     336
      average waiting time: 152.61
      maximum waiting time: 294
      clearence time:       260
   from right:
      number of vehicles:     336
      average waiting time: 152.80
      maximum waiting time: 304
      clearence time:       260
```

At first this alarming change in the average and maximum waiting time scared me, however, after closer examination it does make sense. Given the fact that the light is on for only 4 seconds/time units. This sudden change from a 50 percent chance of cars being added to a 83.3 percent chance has a massive impact on the backlog that will be created. Given the 500 iterations it no longer seems surprising that the waiting is so drastically changed. We can see how big of a backlog there is in the clearance time. To confirm that my results are not incorrect I ran a third test where my input was 4 4 4 4. Although the average wait time was large it is not surprising given the exponential nature of the matter at hand.