
Algorithm 1: FIFO

Input: input cars *Cars*

Output: clear queue *Queue*

```
1 if new car drives in the control area then  
2   |   Push the new car into the Queue;  
3 end
```

Algorithm 2: MS

Input: input cars *Cars*
Output: clear queue *Queue*

```
1 Queue  $\leftarrow$  [];  
2 NowDirection  $\leftarrow$  direction of the first car;  
3 if new car drives in the control area then  
4   car  $\leftarrow$  the new car ;  
5   SNindex  $\leftarrow$  0;  
6   WEindex  $\leftarrow$  0;  
7   if car in SN road then  
8     if car is the first car of SNQueue then  
9       Time[car]  $\leftarrow$  DistanceToIntersection[car]/ReferenceSpeed;  
10    else  
11      Time[car]  $\leftarrow$  Time[SNQueue.last]+delay;  
12      MaxTime  $\leftarrow$  DistanceToIntersection[car]/MinSpeed;  
13      MinTime  $\leftarrow$  DistanceToIntersection[car]/ReferenceSpeed;  
14      Time[car]  $\leftarrow$  max(Time[car],MinTime) ;  
15      Time[car]  $\leftarrow$  min(Time[car],MaxTime) ;  
16    end  
17    Push car into SNQueue;  
18  else  
19    Time[car]  $\leftarrow$  DistanceToIntersection[car]/ReferenceSpeed;  
20  end  
21  Time[car]  $\leftarrow$  Time[WEQueue.last]+delay;  
22  MaxTime  $\leftarrow$  DistanceToIntersection[car]/MinSpeed;  
23  MinTime  $\leftarrow$  DistanceToIntersection[car]/ReferenceSpeed;  
24  Time[car]  $\leftarrow$  max(Time[car],MinTime) ;  
25  Time[car]  $\leftarrow$  min(Time[car],MaxTime) ;  
26  Push car into WEQueue;  
27 end  
28 while Queue.length < (SNQueue.length+WEQueue.length) do  
29   if NowDirection==SN or WEindex $\geq$ WEQueue.length then  
30     Queue.push  $\leftarrow$  SNQueue[SNindex];  
31     SNindex++;  
32     if SNindex<SNQueue.length then  
33       if Time[SNQueue[SNindex]]-Time[SNQueue[SNindex-1]]>4*delay and  
34         WEindex<WEQueue.length then  
35           MaxTime  $\leftarrow$  DistanceToIntersection[WEQueue[WEindex]]/MinSpeed;  
36           MinTime  $\leftarrow$  DistanceToIntersection[WEQueue[WEindex]]/ReferenceSpeed;  
37           if MaxTime>Time[SNQueue[SNindex]]+2*delay and  
38             MinTime<Time[SNQueue[SNindex-1]]+2*delay then  
39             NowDirection=WE ;  
40           end  
41         end  
42       end  
43     end  
44   else  
45     if NowDirection==WE or SNindex $\geq$ SNQueue.length then  
46       Queue.push  $\leftarrow$  WEQueue[WEindex];  
47       WEindex++;  
48       if WEindex<WEQueue.length then  
49         if Time[WEQueue[WEindex]]-Time[WEQueue[WEindex-1]]>4*delay and  
50           SNindex<SNQueue.length then  
51             MaxTime  $\leftarrow$  DistanceToIntersection[SNQueue[SNindex]]/MinSpeed;  
52             MinTime  $\leftarrow$  DistanceToIntersection[SNQueue[SNindex]]/ReferenceSpeed;  
53             if MaxTime>Time[SNQueue[SNindex]]+2*delay and  
54               MinTime<Time[SNQueue[SNindex-1]]+2*delay then  
55               NowDirection=SN ;  
56             end  
57           end  
58         end  
59       end  
60     end  
61   end  
62 end
```

Algorithm 3: Time Calculation

Input: clear queue *Queue*
Output: Time to intersection *Time*

```
1 index ← 0;  
2 for car in Queue do  
3   if index == 0 then  
4     | Time[Queue[index]] ← DistanceToIntersection[car]/ReferenceSpeed;  
5   else  
6     | MaxTime ← DistanceToIntersection[car]/MinSpeed;  
7     | MinTime ← DistanceToIntersection[car]/ReferenceSpeed;  
8     if Queue[index-1].direction == Queue[index].direction then  
9       | Time[Queue[index]] ← Time[Queue[index-1]] + delay  
10      else  
11        | Time[Queue[index]] ← Time[Queue[index-1]] + delay*2  
12      end  
13      Time[Queue[index]] ← min(Time[Queue[index]], MaxTime);  
14      Time[Queue[index]] ← max(Time[Queue[index]], MinTime);  
15    end  
16 end
```
