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

Algorithm 3: Time Calculation

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

```
index← 0;
for car in Queue do
    if index==0 then
        Time[Queue[index]] ← DistanceToIntersection[car]/ReferenceSpeed;
    else
        MaxTime ← DistanceToIntersection[car]/MinSpeed;
        MinTime ← DistanceToIntersection[car]/ReferenceSpeed;
        if Queue[index-1].direction==Queue[index].direction then
            Time[Queue[index]] ← Time[Queue[index-1]]+delay
        else
            Time[Queue[index]] ← Time[Queue[[index-1]]+delay*2
        end
        Time[Queue[index]] ← min(Time[Queue[index]],MaxTime);
        Time[Queue[index]] ← max(Time[Queue[index]],MinTime);
    end
end
```