CSIE 5452, Fall 2019 | Homework 3

資工所 R08922123 王韻豪

1. Intersection Management (16pts)
   Because the whole intersection is modeled as one single conflict zone, there is a deadlock in this situation.
   We just need to give each car different priority and make sure that in each lane the priority order is descended from the start to the end, then there is no deadlock.
2. Level 3 : 在2020Level3的車輛可以量產，車輛已經可以在大部分情況下自行駕駛，但駕駛者需隨時準備控制車輛，自動駕駛輔助可以暫時免於操作，但當汽車偵測到系統無法處理時，立即回歸讓駕駛者接管其後續控制，法規在這部分我想2020可以制定出完整的架構與內容細節。

   Level 4 : 在2025Level4的車輛可以量產，駕駛者可在條件允許下讓車輛完整自駕，車可以按照設定之路線，執行包含轉彎、換車道與加速等工作，除了嚴苛氣候或道路模糊不清、意外，或是自動駕駛的路段已經結束，系統並提供駕駛者「足夠寬裕之轉換時間」，我想政府能在2025時候制定出自動駕駛適用的範圍，並提供合法的路線設定與基礎建設(輔助設施)。

   目前 Google 的 Waymo、Uber的自駕車都具備 L4 等級的能力，可設定目的地後自動將乘客載送到指定區域，在美國部分區域也已經有 L4 無人自駕車開始做接送乘客服務。

   Level 5 : 在2040Level5的車輛可以量產，「完全自動駕駛」的車輛，駕駛者不必在車內，車內也不會有「方向盤」設計，乘客任何時刻都不會控制到車輛。這部分我想需要很長一段時間才能達成，各個城市之間的建設都需要到達一定程度才能夠實現。

3.

```python
import sys
time = 0
back_edge = []
class vertex():

    def __init__(self, args):
        self.number = args
        self.edge = []
        self.ischeck = 'white'
        self.checktime = 0
        self.endtime = 0
        self.predecessor = None

    def add_connect(self, args):
        self.edge.append(args)

def dfs(vertex,vertex_list,predecessor=None):
    global time
    global back_edge
    time += 1
    vertex.checktime = time
    vertex.ischeck = 'gray'

    for j in vertex.edge :
        if j in vertex_list.keys() and vertex_list[j].ischeck == 'white':
            vertex_list[j].predecessor = vertex
            dfs(vertex_list[j],vertex_list,vertex)
        elif j in vertex_list.keys() and vertex_list[j].ischeck == 'gray':
            back_edge.append(dict(front = vertex.number, end = vertex_list[j].number))

    vertex.ischeck = 'black'
    time +=1
    vertex.endtime =time

if __name__ == '__main__':
    sys.setrecursionlimit(1000000)
    input_file = open(sys.argv[1],'r')
    total_amount = int(input_file.readline())
    total_edge = int(input_file.readline())
    vertex_list = {}

    for edge in input_file.readlines():
        vertex_front,vertex_rear = edge.strip().split()

        if vertex_front not in vertex_list.keys():
            temp = vertex(vertex_front)
            temp.add_connect(vertex_rear)
            vertex_list[vertex_front]=temp
        else:
            vertex_list[vertex_front].add_connect(vertex_rear)

    input_file.close()

    for i in vertex_list.keys():
        if vertex_list[i].ischeck == 'white' :
            dfs(vertex_list[i],vertex_list)
    if back_edge == []:
        print("There is no cycle in graph!")
    else:
        print("There is a cycle in graph!")
        for j in back_edge:
            print("The removal edge",j)
```