#2 为 Agent 构建最短路

Agent 类：

agent_id——编号，不用于计算，程序内部用 agent_seq_no 为读取的 agent 重新编号.
o/d_zone_id——该 agent 的起始 zone_id 与终到 zone_id.
path_id——路的编号，如从 zone-x 到 zone-y 可能有 4 条路，便以此用 0,1,2,3 代表.
agent_type——通过 settings.yml 定义，如 p-passenger，w-walk 等.
demand_period——通过 settings.yml 定义，如 period:AM，time_period: 0700_0800.
volume——交通量
以下是计算后得出的值：
distance——计算出的该 agent 所要完成目标的最短路.
travel_time——计算出的该 agent 经最短路所需时间.
node/link_sequence——计算出的该 agent 经最短路所经过 node/link_id.

此程序的输入与输出：

输入——node.csv，link.csv，agent.csv，demand.csv

agent_id	o_zone_id	d_zone_id	path_id	agent_type	demand_period	volume
1	1	2	0	p	AM	347.31

表示 id=1 的 agent 从 zone-1 到 zone-2 的流量为 347.31.
输出——agent_paths.csv

agent_id	o_zone_id	d_zone_id	path_id	distance	node_sequence	link_sequence
0	*1*	*1*				
275	1	2	0	3.06317	1;547;548;2	1;986;989
623	1	3	0	4.10747	1;547;549;3	1;987;994
1014	1	4	0	5.6817	1;547;549;550;4	1;987;996;998
1219	1	5	0	7.21787	1;547;549;551;5	1;987;997;1005

agent.csv 其实是简略化的输入，程序而后采用 int(volume)+1 重置了 volume，然后用新的 volume 值除以 settings.yml 中对应 agent 的 pce（换算系数），最终得到实际的车辆数，并将每一辆车作为一个新的 agent 来进行计算。
而 agent_paths.csv 也是简略化的输出，如图，其真正含义为：275-623 号 agent 为从 zone-1 到 zone-2 的车流，其最短路为 3.06，最短路对应的 node 顺序为...对应的 link 顺序为...
这就解释了在测试案例中，为何 agent 输入了 285959 个值，各区域间的 demand_volume 总和为 1137493 的情况下，程序最终输出的 agent 总共却有 1328195 个的问题——是因为程序在有 113 万个 demand_volume 的基础上，又因 int(volume)+1 的缘故，多了 20 万左右个估算值，才有 132 万 agent 这个数字.

母方法：

find_path_for_agents(G, column_pool, engine_type='c')——path.py
作用是为每一个 agent 的需求，使用 single_source_shortest_path()函数为其寻找最短路.

setup_agents(column_pool)——classes.py
用于初始化 agent，将每个输入的 agent 按其 volume 与 pce 按每辆车换算为新 agent.由于最短路程序输出的是 node-node 最短路，所以要根据数据源中输入的 o/d_zone_id，将其随机转化为 zone 中的任一 node_id.

```
pg.output_agent_paths(network,False)
```

‘output unique agent paths to a csv file’，输出时只将 od 不同的 agent 输出（相同 od 的 agent 只输出一个）.