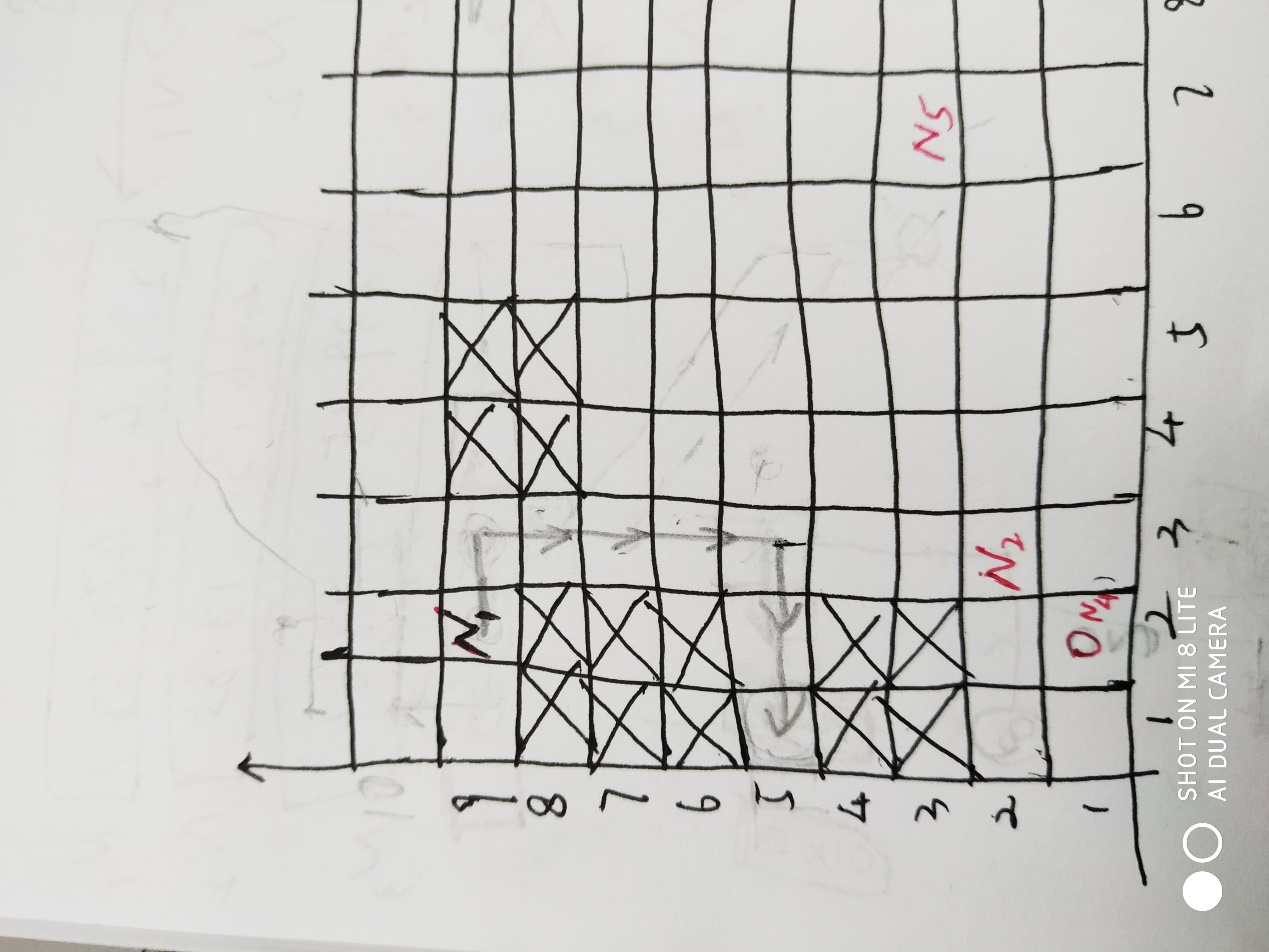
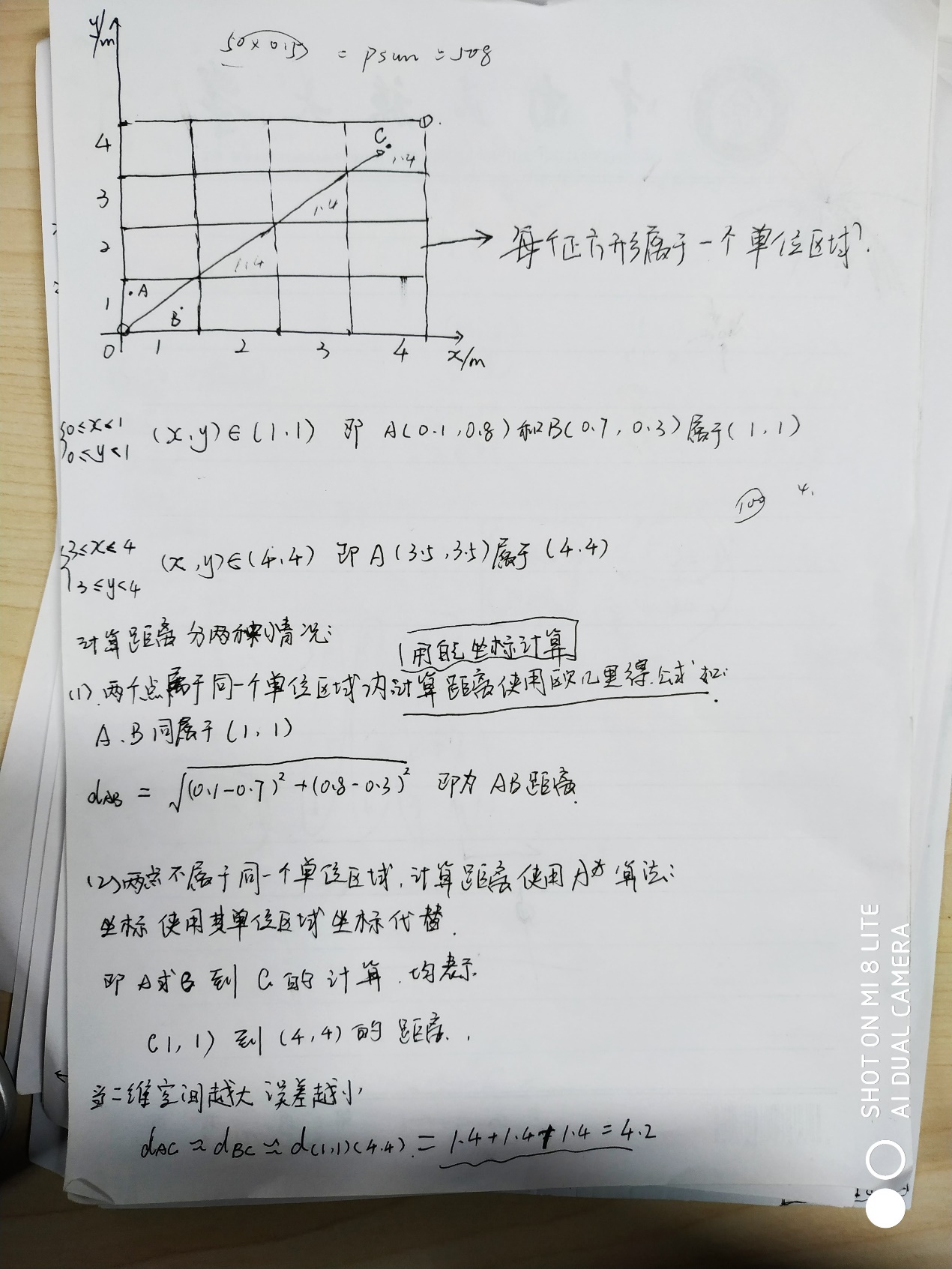
**1、A\*算法是在已知“起点”和“终点”的前提下找到最佳路径：**

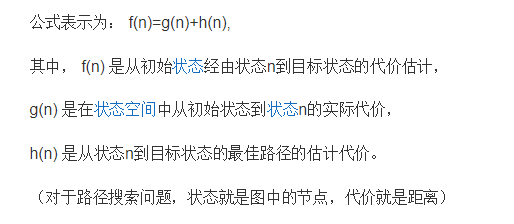
1. 首先，使用电单车节点运动的空间的边长作为A\*算法的计算空间的边长，然后将障碍区域，障碍节点添加进去🡪先创建这么一个空间
2. 障碍的大小为10m\*10m，总共20个，与创建电单车运动空间时的平面相同。
3. 将“起点坐标”和“终点坐标”映射到A\*算法的计算空间中【类似于二维平面，不过是用n个1m\*1m作为一个小网格来划分整个空间的】
4. 整个空间是用一个二维数组保存，存在障碍的小网格对应的数组值为1，不存在障碍的小网格对应的数组值为0。如图：存在X的为障碍



1. 在A\*算法中计算两点之间距离分为两种形式：

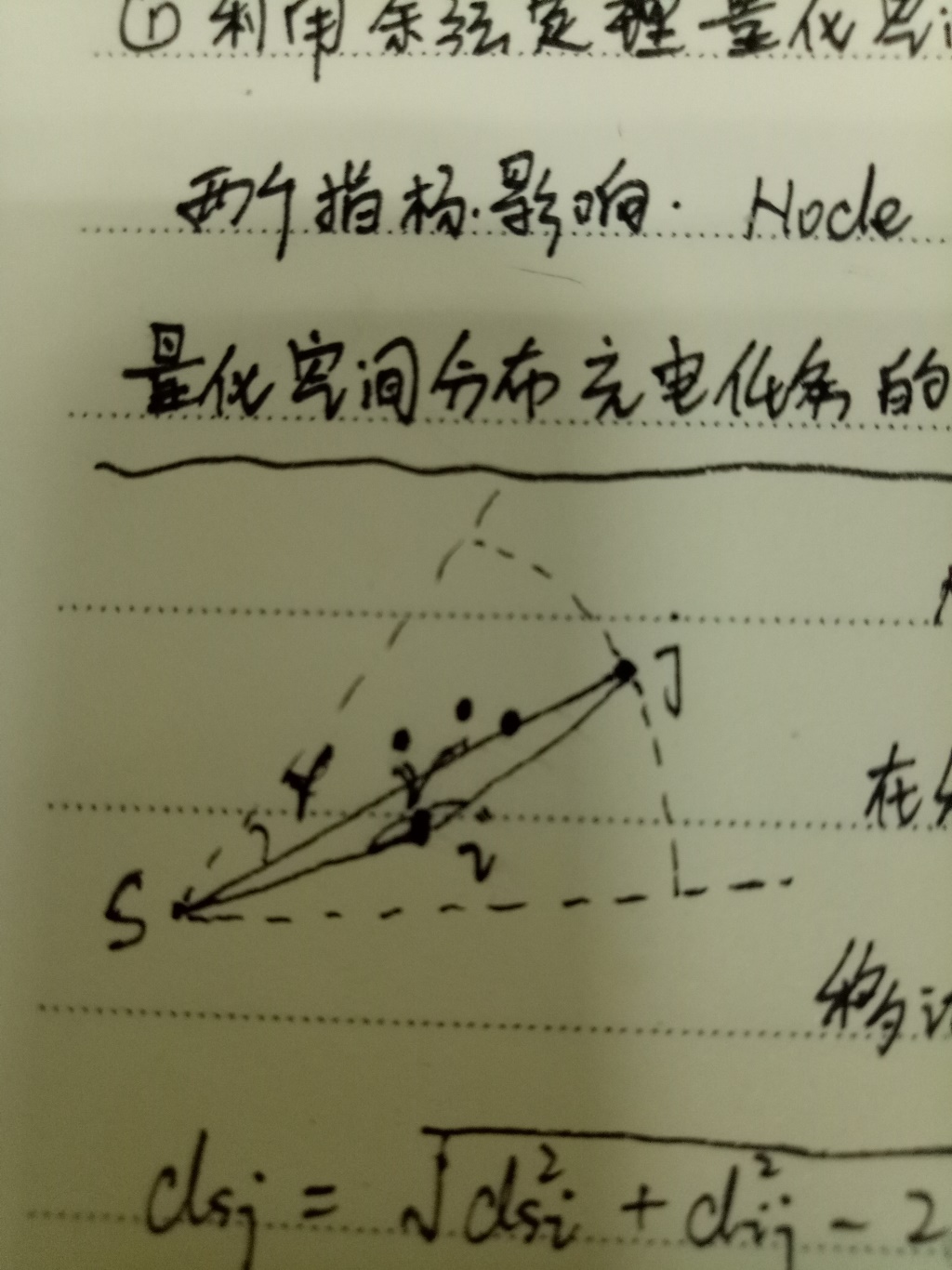


1. 在同一个小网格内：如图片中A(起点)，B(终点)，直接利用A，B两点坐标点入欧几里得距离公式，求AB最短路径。
2. 不在同一个小网格内：如A(起点)，C(终点)，
3. 首先需要取整处理，A坐标为(x1,y1)🡪A(int(x1), int(y1)),int整型取的是数值的下限，对应B也做相应的处理。【计算空间相对来说】
4. 在A\*算法中存在评价函数f (n) = g(n) + h(n), f(n)每次计算都取局部最小值(局部最优)，计算f(n)值的过程中，已经保证节点是不会穿过障碍区域的！！！计算过程每次只走一个子网格！！需要通过当前起点位置的点移动到终点为止。



**2、插入算法解释：**

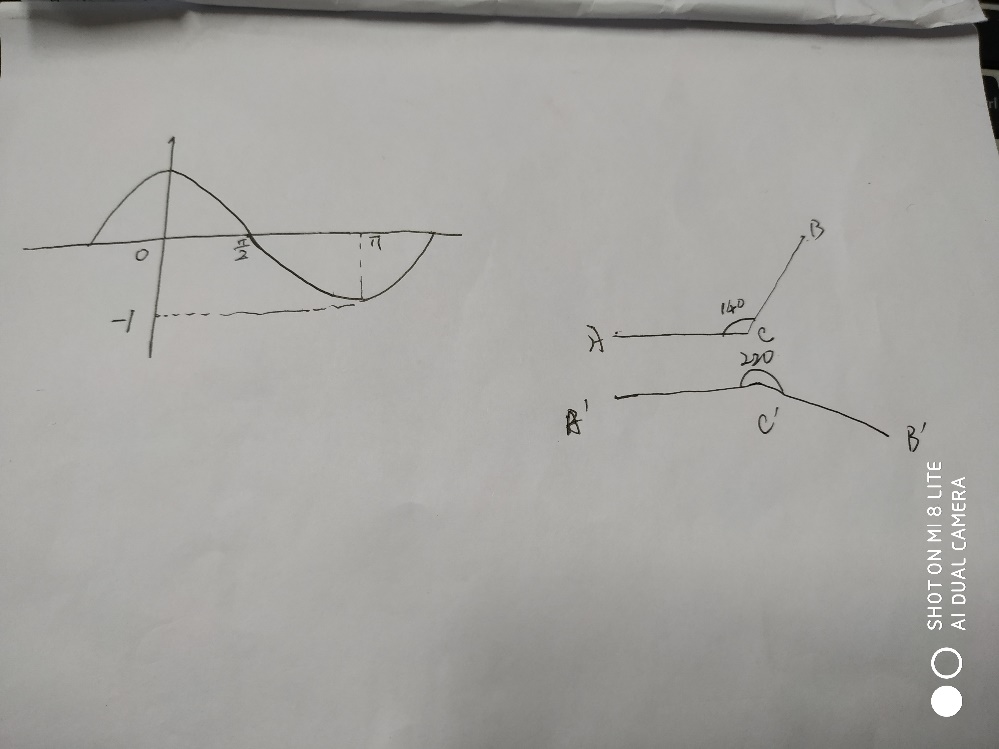
情况描述：当MCV选定电单车作为目标充电节点后，在移动至的路途上优先查询判断ALERT缓冲池中是否存在已经发送了ALERT且未被添加到充电回路的节点，若存在，则判断该节点是否满足公式(1),若满足公式(1)则先将该节点插入充电回路，若ALERT缓冲池中不存在发送过ALERT的节点，则判断Request缓冲池中是否存在已经发送了Request且为别添加进入充电回路的节点，若存在则判断该节点是否满足公式(1),若满足公式(1),则将该节点先插入充电回路。【每次运行插入算法仅时，允许至多1个中间节点被插入】由于MCV在工作时的能量消耗主要集中于移动能耗以及充电能耗，而充电能耗要远小于移动能耗，所以在考虑电单车选择性插入时主要考虑移动距离。



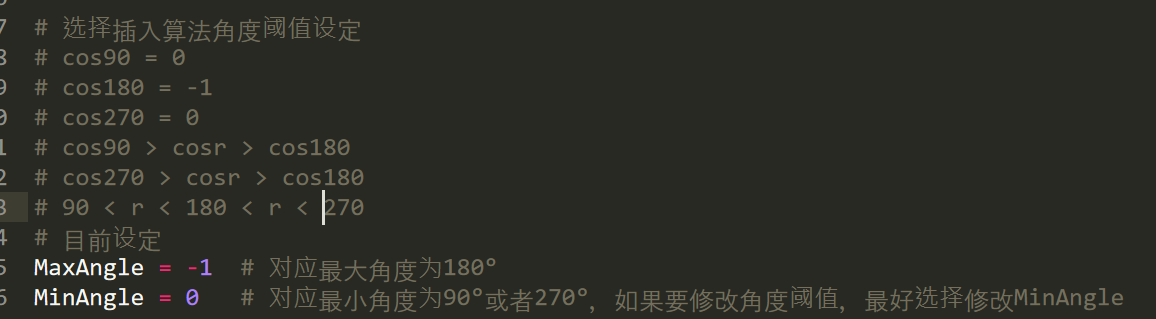
说明：当MCV从S处出发要行驶向j处为之服务，中途遇到i发出请求，判断是否为i服务的标准是判断si与ij之间的夹角是否接近于180度，因为

(1)

因此如果原始是选择从电单车i-j进行充电，中途遇到k，则构建i-k以及k-j的真实路径，可以得到路径长度，原始路径长度i-j的也可以得到，构建三角形ikj，已知三边长度，可以得到任意两边的夹角，判断该夹角是否大于某个阈值角度，如果大于就可以插入，反之就不能插入。



图中Cos(A,B) = Cos(A’,B’)的，但是AB和A’B’的夹角根据参考系不一样，它们的夹角可以相等，可以互补，对于它们的Cos值都是一样的



Algorithm 按需充电(以第一种出发机制为例【动态缓冲池】)

1：数据初始化,输入节点序列NodeList

2：while(len(NodeList) != 1)

3： if NodeRequestFlag is False

4: 电单车开始运行

5： 改变电单车节点的坐标和剩余能量等信息

6： 将发送了Request的节点序号按顺序保存到Request\_list中

7： if ((len(Request\_list) >= RequestThreshold) or ((len(Request\_list) >= (len(NodeList) -1))) and ( i == len(len(NodeList) - 1))

8： 修改距离矩阵

9: NodeRequestFlag = True

10： end if

11： end if

12： while (NodeRequestFlag is True):

13： if Request\_list的长度 >= 缓冲池阈值RequestThreshold:

14： 从Request\_list中提取前RequestThreshold个节点构成R\_list序列,用于充电

15： end if

16： if l Request\_list的长度 < 缓冲池阈值RequestThreshold:

17： 从Request\_list中提出全部节点构成R\_list序列,用于充电

18： end if

19： 依次取出R\_list中的节点

20： 调用判断决策算法，判断取出的当前点是否可以被充电

21： if 可以被调度

22： 将该节点从Nodelist中剔除，len(NodeList)会减一

23： 当前点被选择进行充电

24： 其它节点继续运动，修改坐标位置，剩余能量等信息

25： 将发送了Request的节点添加到Request\_list中

26： 修改距离矩阵

27： end if

28： else if 不可调度

29： 跳过当前节点，选择下一节点

30： end if

31： if R\_list的长度 == 1:

32： 修改RequestThreshold的值

33： if Request\_list的长度 == 0:

34： NodeRequestFlag = False

35： break

36： end if

37： if Request\_list的长度!= 0:

38： if Request\_list的长度 >= 缓冲池阈值RequestThreshold:

39： 用目前Request\_list中的RequestThreshold个节点，构建R\_list

40： end if

41： if Request\_list的长度 < 缓冲池阈值RequestThreshold::

42： if 未充电节点个数len(NodeList) – 1 == Request\_list的长度:

43： 用目前Request\_list中的所有节点，构建R\_list

44： end if

45： else if未充电节点个数len(NodeList) – 1 != Request\_list的长度:

46： NodeRequestFlag = False

47： break

48： end if

49： end if

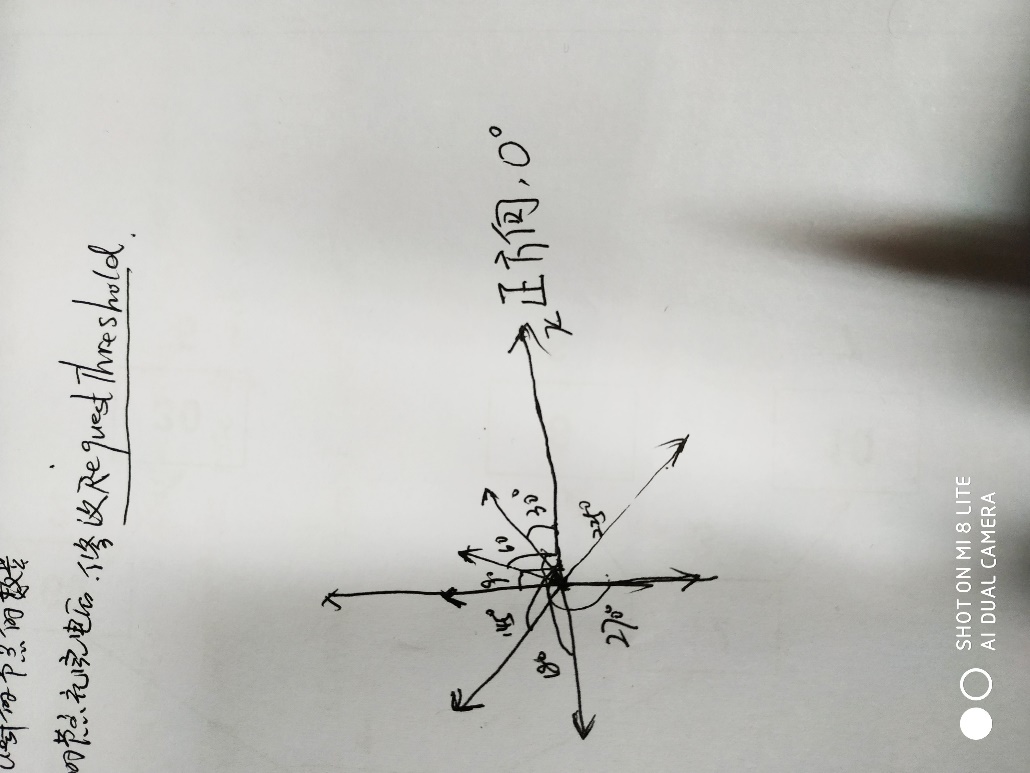
50： end if

51： end if

52： end while

53：end while

3、关于共享电单车在网络中移动的方向：



初始化电单车移动方向(或者角度)，以x轴的正方向为0°，射线的起点为当前电单车节点，随机产生任意一个方向角【也许是30°或者是225°或者是其它角度】🡪小车起始运动方向就是这么来的

运动过程中，如果电单车遇到障碍，它也会通过方向角来改变自己的移动方向的：

1. 首先如果遇到障碍，在还没重复一圈时，AlphaValue = AlphaValue + 30
2. 在遇到障碍后，重复了一圈后，AlphaValue = AlphaValue + 1【更细化】