代码实现

报告中先介绍准确率超过96%的代码实现，再分别介绍通过93,90,70的代码中对于代码速度的优化

头文件和自定义常数如下



头文件便于使用c++标准库中的数据结构和函数

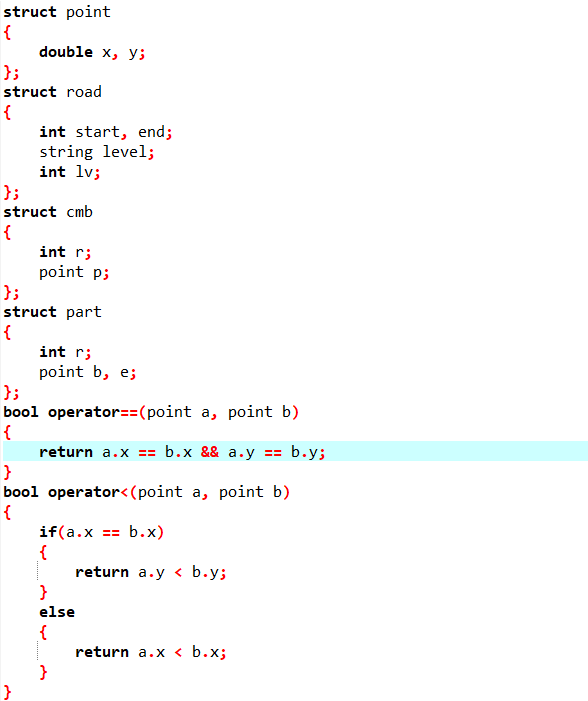
PI是圆周率

dis是网格边长大小，代码把整个地图分成了每个格子大小为60m \* 60m的网格

sigma是计算发射概率时使用的常数，影响发射概率在总概率中所占的比重

beta是计算转移概率时使用的常数，影响转移概率在总概率中所占的比重

自定义结构体如下



结构体point用于表示点，记录了点的xy坐标

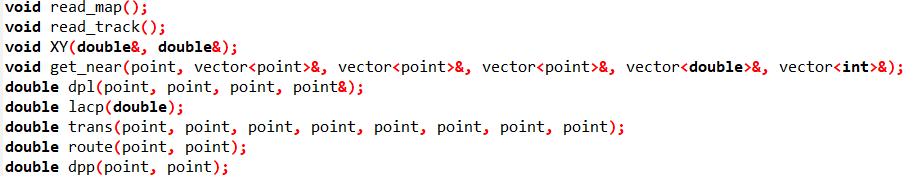
结构体road用于保存路（很长一条的，有很多GPS定位点组成的）的部分信息，包括起始点编号start，终止点编号end，道路等级（字符串形式）level，道路等级（数字形式）lv

结构体cmb，用于表示可到达点，p表示可到达的点，r到达该点通过的路的编号

结构体part，用于保存路段（很小一段，仅有两个GPS点组成的，一条路可以视为很多条路段拼接而成），包括起始点，终止点，路段所属路的编号。

此外，为了方便map和后续其它使用的方便，为point重载了两个运算符

自定义函数如下



函数read\_map用于读取地图信息（即路段信息）

函数read\_track用于读取需要匹配的轨迹，并匹配轨迹，最后将结果输出

函数XY用于经纬度向直角坐标的转换

函数get\_near用于抓取轨迹中的GPS点附近的路，并传回一些相关信息， GPS点到路上的投影点，GPS点与路的距离（以发射概率的形式传输），GPS所在路的始顶点和末顶点

函数dpl用于计算点到线段的距离

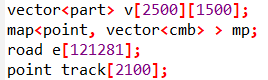
函数lacp用于将gps点到路段的距离转化为发射概率的数值

函数trans用于计算转移概率

函数route用于计算路上两点之间在路网上的距离（即通过路网移动最少多少距离能从一个点到达另一个点）

函数dpp用于计算两点间的距离

所用数据结构



vector<part> v[2500][1500]用于记录每个网格内经过的路段。这是一个二维数组，每个元素都是一个向量，当有路段经过某网格时便把它加入到该网格对应的向量中去

map<point, vector<cmb> > mp是邻接表储存路段信息的容器，记录了每个点可以到达的点，如果点a可以通过路c到点b，则在向量mp[a]中添上由b和c组成的cmb类型变量

road e[121281]用于储存部分路的信息

point track[2100]用于储存当前正在匹配的轨迹GPS点序列

下面介绍各个函数的细节

main函数

其中调用了read\_map和read\_track，读取地图，读取轨迹，并输出匹配结果

XY函数

由于地图所在的范围很小，所以xy坐标关于经纬度的变化近似可以认为是线性的，设定好两个一次函数即可

read\_map函数

该函数先读入路的条数，之后读取每条路的信息。读取每条路的信息时，当读取了两个在同一条路中且相邻的GPS点时，就得到了一个路段，并把路段信息加入到它所经过的所有格子里面，同时为前一个点所能到达的点中添加上后一个点和这条路的编号的信息。

read\_track函数

该函数先读入轨迹的条数，之后依次读入每条轨迹。对于每一条轨迹的读入和处理，首先把轨迹的所有GPS点读入track中，top保存GPS点的个数，之后定义了很多二维向量，第一个数对应GPS点的是第几个（从0开始），第二个数对应每个点可能匹配到的路段的序号（不是路的编号，只是一个用于标记的序号），这些二维向量中，r储存路的编号，bosi储存每个匹配路最有可能来自于之前的哪条，b储存发射概率，delta储存发射概率与转移概率之积，x储存匹配时gps点在路段上的投影点，head储存路段的起始点，tail储存路段的终止点。之后，调用get\_near函数获得head、tail、x、b、r，随后进行隐马尔科夫模型的计算，包括调用trans计算转移概率，深入计算，回溯等过程。将计算的结果储存在cand中，最后输出结果。

get\_near函数

该函数在gps点所在网格以及其周围的八个网格中寻找经过了这些网格的路段，重复的不取，调用dpl计算点到路段的距离并获得投影点的坐标，最后筛选出发射概率不低于最大发射概率除以2.9的路段作为最终的待匹配的路段

dpl函数

用数学公式计算点到直线的距离，如果是端点离轨迹点最近，则使用这个端点到轨迹点的距离。

lacp函数

在距离大于200m的返回发射概率为0，其余时候利用论文上的公式返回对应的值

trans函数

计算公式是利用两个轨迹点间的距离和两个投影点的路段距离的差的绝对值作为评判标准，套用论文上的公式，如果差值过大则直接返回一个非常小的转移概率。其中，调用了函数route获得两个点的路段距离

route函数

使用广度优先搜索（6层）寻找可以从一个点到达另一个点的路径

dpp函数

使用数学公式计算两点间的距离

70的优化

1. 剔除了转移概率的计算
2. 读入轨迹是如果得到了距离很近的gps点，则认为这两个点所在的路段相同，当做一个点处理
3. 选取路段时考虑前进方向与路段方向的夹角，如果夹角大于90则直接不予选取
4. 输入的优化，加快了速度

90的优化

1. 计算转移概率时如果是前后两个匹配的路段所属路相同则直接设为1，如果前匹配点所属路段的终止点编号等于后匹配点所属路段的起始点编号则直接返回0.86，否则再计算，且此时的计算广度优先搜索的层数仅设置为1层
2. 计算转移概率时，如果前一个点的总概率除以beta已经小于当前点的总概率，则无需再计算，因为转移概率最大不超过beta分之一
3. 寻找附近路段时，如果多个路段同属一条路，则只选取一个发射概率最大的加入待匹配路段

93的优化

1. 广度优先搜索的层数降至了3层

修改后准确率取得突破的几个点

1. 计算点到路段的距离时，不能只使用垂线距离，要考虑路段上离轨迹点最近点是端点的情况，这一处修正之后，准确率可以达到70以上
2. 由于路段的情况复杂，如果将方向考虑纳入选择临近路段时，有可能忽视正确的路段，删去方向的限制之后准确率达到85以上
3. 转移概率的层数，当广度优先搜索的层数逐层增加时，概率也会增加，从1到4有着明显增加，4到6有着缓慢增加，6之后增加极度缓慢，调至6后可以使得准确率达到93以上
4. 参数的选择和调整。使用手动调整时，很有可能找到的是一个极值点，在一个比较大的范围内先逐一尝试之后再逼近，可以接近最大值点，把参数从之前调到的极大值点修改到接近最大值点（应该是）的地方准确率可以达到97以上