# 需求分析

一个天大新校区的导航软件，能够针对单行道、机动车道、人行道的输入，给出两点间的最短路线。

# 概要和详细设计

程序功能：

本程序实现了一个导航软件，能够读入图中的边和点，然后根据用户输入，或是恰好在某个位置，或是离某个已知点的方向和距离，以及用户想到达的点，最后给出导航路线。

设计概要：

很明显，这个问题需要用最短路dijkstra算法求解，单行道其实就是单向边，双行道就是双向边，而机动车道和人行道的区分只要Dijkstra出两个实例,dj[0],dj[1],加边的时候若是人行道(sidewalk)就加到dj[0],若是机动车道(lane)就加到dj[1]即可。

为了与实际生活更加贴近，我在边结构体(Edge)中多定义了一个量(dir)用于标记边的方向，这样在最后最短路回溯答案时，可以根据边的方向更好的指路。

关于导航：

我分了两种情况讨论:

1：使用者恰好位于某个已经在地图上标记出的点:

利用dijkstra算法，从使用者所在点出发跑最短路，在最短路过程中记录每个点的前驱边的编号在p数组中。然后利用p数组从终点向起点回溯，回溯过程中经过的点和边就是导航的路线啦。然后注意下顺序，因为我们是从终点向起点回溯的，需要把中间经过的边先记下来，反个序才可以。

2：使用者只知道自己距离某个已标记点的方向和距离：

设计背景：要考虑这种情况是因为实际生活中我们很难非常碰巧就在某个已标记点，而把自己的当前位置强行等同于某个已标记点，再去导航，会有许多问题（尤其是你对周围道路并不熟悉）。

这个问题比前一个复杂，但我们可以尽量将他转化为前一个问题。

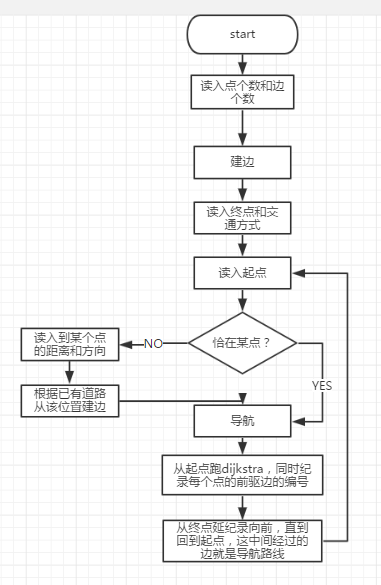
首先我们将使用者当前位置作为一个新的点加入原图，并根据原图的信息为这个点建边，这样这个点就变成了一个当前地图上的已知点了，然后就可以用前一个问题的方法解决了。

而建边过程就是找两条边即可:

一条是同向边，从已标记点的边中找出方向输入方向相同的，那么就从新加点向该边终点连一条边；

一条是反向边（可能不存在）：枚举所有边，直到找到一条方向与给定方向相反且终点是已标记点的边，如果找到这样的边，就从新加点向已给定点建一条边。

# 程序结构



# 代码解释

Line 10：定义宏 startidx=n+1，意为始终以n+1作为导航时才加入图中的点（也就是使用者当前位置的点始终默认为第n+1个点）

Line 15 to 17 and 31 to 71:最短路算法dijkstra

Line 18 to 25:D数组将方向（char\*）转化为int,R数组将交通方式（char）转化为Int

Line 27,28:rid将路名映射成int,road存储每个int对应的路名（也可以说rid是将路名映射成下标）

Line 72 to 89:guide(int s,int t)函数，从s向t导航，dijkstra(int s)是最短路算法，d数组记录s到下标对应点的最短路长度，p数组则纪录边编号。所有的边都是存储在edges数组中的，所以只需要编号即可。先从终点t开始逆向延p数组回溯到起点s,把中间经过点存入v数组，再将v数组逆向存入g数组，这样g数组从0开始就是每个要经过的点了，导航过程就是依次输出g数组每个点的前驱边的起点，终点，方向和距离。

Line 91 to 111:notAt(int s,int t,int dist,char cdir)函数，处理不是恰好位于已经标记的位置的情况，由于要加入新的边，会影响到原来的地图，所以先用oldEdges存下改之前的图，导航结束后再改回去。

NotAt函数4个参数的意义分别是：

t:终点，而使用者当前位于距离s点dir方向dist距离处。

我们新加一个点start标记使用者当前位置,然后建立与他相关的边。

首先从s出发找到与dir同向的边，那么start到该边终点e.to的距离就是e.dist-dist;

显然start是位于道路中间的某个点，他也可能存在反向能回到s的边，于是我们遍历图中所有的边，若找到一条边e，他的终点(e.to)是s，并且他的方向恰与cdir相反，那么从start到s就有一条权为dist方向与cdir相反的边。

然后再从start到t跑一遍guide函数就好啦。

Line 149:我们规定对于where are you?的问题回答的指令如下：

1：形如 200 east to changcheng

2：形如 at changcheng

2对应恰好在已标记点，1对应使用者知道自己距离某个已标记点的距离和方向。

# 测试

注释掉的为输出结果

（第一组）

6 9

v0 v1 100000 north lane

v1 v2 5 east lane

v2 v3 50 east lane

v0 v4 30 east lane

v0 v5 100 north lane

v4 v5 60 north lane

v0 v2 10 east lane

v3 v5 10 north lane

v4 v3 20 south lane

v3

sidewalk

2 west to v1

//not accessable,没有建sidewalk的边，根本没有路

v5

lane

at v0

/\*from v0 to v4 in the east 30 meters away

from v4 to v3 in the south 20 meters away

from v3 to v5 in the north 10 meters away

v4到v5有长为60的路，但不是最短的\*/

v3

lane

3 east to v1

/\*from your position to v2 in the east 2 meters away

from v2 to v3 in the east 50 meters away\*/

(第二组)

3 4

v0 v1 3 east sidewalk

v1 v0 3 west sidewalk

v1 v2 10 east sidewalk

v2 v1 10 west sidewalk

v2 lane at v1

//not accessable

v2 sidewalk 2 east to v1

//from your position to v2 in the east 8 meters away