邻接表数据结构：

用来存储图的数据信息，上个样例：

4 5（4为顶点，5为边数）

1 4 9 第一边（顶点1到顶点4距离为9）

2 4 6 第二边

1 2 5 第三边

4 3 8 第四边

1 3 7 第五边

首先用3个数组存储顶点间的关系，

for(int i = 1;i <= 边数;i++)//这里的i是第几边的意思，有用的。

{

输入数据

u[i] = ....; v[i] = ....;w[i] = .....;

}

用first数组记录起始顶点在第几边，first数组长度跟顶点数相同（顶点的编号是递增加1的，1、2、3.....这样子），比如说first[2] = 2就是说明编号为2的顶点的第一条（上面输入的2 4 6）出边在第二边。要先first数组全部设为-1；

用代码就是这样实现

for(int i = 1;i <= n;i++)

first[i] = -1;

for(int i = 1;i <= 边数;i++)//这里的i是第几边的意思，有用的。

{

输入数据

u[i] = ....; v[i] = ....;w[i] = .....;

.................

first[u[i]] = i;//如果前面有相同的起始顶点，会覆盖掉，原来的数值会给next[i]

//遍历时候顺序会和输入的顺序相反

}

然后再来个next数组，接着first数组，记录第二条、第三条.....的边数信息。next的长度与边数相同。用代码就是这样实现

for(int i = 1;i <= n;i++)

first[i] = -1;

for(int i = 1;i <= 边数;i++)//这里的i是第几边的意思，有用的。

{

输入数据

u[i] = ....; v[i] = ....;w[i] = .....;

next[i] = first[u[i]];

first[u[i]] = i;

}

讲的可能不清楚，把上面的输入数据拿下来

4 5（4为顶点，5为边数）

1 4 9 第一边（顶点1到顶点4距离为9）

2 4 6 第二边

1 2 5 第三边

4 3 8 第四边

1 3 7 第五边

First:5 4 -1 2;

Next:-1 -1 1 -1 3;

遍历就是

k = first[1];

While(k != -1)

{

输出u[k],v[k],w[k];

k = next[k];//-1就相当于没有，结束循环

}

这个邻接表实现来自于《啊哈算法》思想就是把每次输入记为号数，第一边，第二边.....输入要有起始顶点，结束顶点，边长。就是按边数来存储。然后first数组就是记录各个顶点的出边，没有为-1；然后next根据first数组里面的值看下有没有剩下的边，next的下标就是表示输入时的第几号边，这一号边下一个相同起始顶点的边是几号，没有就为-1；

个人感觉实现起来很复杂，太多数组了，关系一层套一层。五个数组，花费的空间也很大，

但是实现的基础只用数组，就各个支持数组的语言都可以实现这个版本。邻接表用于顶点数小于边数的平方的图，会比邻接矩阵快很多。但是实现真的太复杂了，感觉好绕。