# 普通最短路

以下M代表边数，N代表点数

## Dijkstra算法：

不能有负数价值边，使用优先队列(堆)优化时间复杂度M\*logN，单原点算法。

数组dist表示源点到各个顶点的最短距离，堆que表示需要以此为中转点更新其他店的点的集合，堆每个节点有2个值，点编号I,和到原点距离dis

用邻接表或者链式前向星存图的数据。设原点是s，初始，dist除了 （dis[s] = 0）。其余全是INF，类似广度优先，把原点入堆，每次弹出堆顶元素,把它叫now，遍历当前点now的所有可到达点next，若dist[now]+w（now,next）小于dist[next]本来值，就更新并且入队等待下次遍历。

* 存在重边怎么办？

在邻接表建立的时候(一般是输入时)，就过滤掉，取最小的

* 有无法到达的边怎么办？

不需要考虑，有的话dist[i]会等于INF

* 对于有向图也适用
* 对于含有负边的图不适用此算法
* 自环需要考虑，输入时候排除掉
* 迪杰斯特拉也可以像bf或者spfa那样记录入队次数，判断图的环路，但是图不能有负边，会导致错误。

## bellman-ford算法

时间复杂度 N\*M

dist数组和Dijkstra算法中表示的意思一样

1,.初始化：所有顶点的最短距离值是INF,原点到自身是0

2.迭代求解：反复对边集E中的每条边进行[松弛操作](https://baike.baidu.com/item/%E6%9D%BE%E5%BC%9B%E6%93%8D%E4%BD%9C)，使得顶点集V中的每个顶点v的最短距离估计值逐步逼近其最短距离；（dotNum-1次,更新一定能求得最短路）

3.检验负权回路：判断边集E中的每一条边的两个端点是否收敛。如果存在未收敛的顶点，则算法返回false，表明问题无解；否则算法返回true，并且从源点可达的顶点v的最短距离保存在 d[v]中。

其实就是检验 每条边 的起点最短路dist[b]加上当前边价值val是不是小于末点的dist[e],小于的话说明存在负回路，不小于就不存在

* 存在重边怎么办？

在建立边时候，就过滤掉，取最小的

* 有无法到达的边怎么办？

不需要考虑，有的话dist[i]会等于INF

* 对于有向图也适用
* 对含负边的图也适用
* 可以输出存在哪些负环
* 存图的时候简单地一条条边存更好，而不是用邻接表
* 自环需要考虑，输入时候排除掉

## SPFA算法

队列优化的bf算法(bf算法优化的一种)，时间复杂度不稳定，M\*K,k是大常数，一般认为普通稀疏图和迪杰斯特拉相当，密集图和bf一样很慢。

建立一个队列，初始时队列里只有起始点，建立一个dist记录起始点到所有点的最短路径,这里同迪杰斯特拉算法和bf算法的dist,,特殊的SPFA算法还需要一个updateTimes记录每个点入队次数，需要vis记录在不在队列中。然后执行松弛操作:用队列里有的点作为起始点去刷新到所有点的最短路，如果刷新成功且被刷新点不在队列中则把该点加入到队列最后。重复执行直到队列为空。

判断有无负环：  
　　如果某个点进入队列的次数超过N次则存在负环（SPFA无法处理带负环的图）

* 存在重边怎么办？

在邻接表建立的时候(一般是输入时)，就过滤掉，取最小的

* 有无法到达的边怎么办？

不需要考虑，有的话dist[i]会等于INF

* 对于有向图也适用
* 对含负边的图也适用
* 不稳定的算法，不适用于稠密图，总体效率不如迪杰斯特拉，但大大强于bf
* 自环需要考虑，输入时候排除掉

## 弗洛伊德算法：

# 其他最短路问题：

## 恰好经过k条边的多源最短路：

邻接矩阵的意义是两点经过一条边最短路，对邻接矩阵，做类似佛洛依德算法的变换，迭代k,dis[i][j]=min(dis[i][k]+ dis[k][j]),可以得到了恰好经过一条边的最短路。把这个操作看成乘法，就可以用矩阵快速幂求出恰好经过k条边的多源最短路了。时间复杂度N^3\*logk