# 普通最短路

以下E代表边数，V代表点数,dist[i]代表原点到编号为i的点的最短路，定义**松弛操作**:

给两个点u和v,dist[v]=min(dist[u],dis[u,v]); dis[u,v]代表点u到v的边的长度。

## Dijkstra算法：

**算法思路：**

给的一个图，把图中的点分为两部分，已经确定了最短路的点(用集合S表示)，和没有确定最短路的点(用集合Q表示)，dist表示源点到各个顶点的最短距离,初始dist[s]=0。S只有原点s

定义对顶点u的**松弛操作**：遍历全部和u相连的其他点v，令dist[v]=min(dist[u]+w(u,v), dist[v])

在Q集合选取所有与S集合有边相连的点中，dist值最小的点u，对u周围的点全都进行**松弛操作,**之后把这个点u加入S集合，这里保证能确定dist[u]一定是s到u的最短路，因为我们是选取Q集合dist值最小的点，Q集其他点dist一定大于u，没有负权边的情况下，意味着你怎么松弛，也不能使dist[u]更小。

反复进行上述步骤V-1次，就能找到所有点最短路。若果暴力的找Q集合dist值最小的点，时间复杂度是o(V^2+E) ,堆优化是o(ElgV) ，斐波那契堆是o(VlogV+E)

**代码思路：**

使用优先队列(堆)优化时间复杂度ElgV，单原点算法，不能有负数价值边。

数组dist表示源点到各个顶点的最短距离，堆que每个节点有2个值，点编号i,和到原点距离dist,堆顶是Q集合中dist值最小的那个点

用邻接表或者静态邻接表存图。设原点是s，初始:dist除了 （dist[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。

迪杰斯特拉算法有个优化是：对于每次的当前点now， 若now.dis> dist[now],那么说明now.dis代表的路径肯定不是最短的路，直接跳过这层搜索即可。由于now.dis不可能小于dist[now]，有些地方写成if(now.dis!=dist[now.i]) continue;

* 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

* 有无法到达的点怎么办？

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 V\*E，dist数组和Dijkstra算法中表示的意思一样

**代码思路：**

对所有边进行一次松弛操作，这样反复重复V-1次一定能得到所有点最短路。

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" \t "_blank)，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)

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，存在负环说明图没有最短路，终止程序。

**性能分析和优化：**

密集图和bf一样很慢，总体效率肯定不如迪杰斯特拉，但大大强于bf，稀疏图很快，均摊复杂度能达到O(E)，当然它是很容易被卡的，比如网格图，立方体图,菊花图都能拉高SPFA的复杂度，但有几种优化可供参考：

1. 改用栈而非队列
2. 用双端队列，每次将入队结点距离和队首比较，如果更大则插入至队尾
3. 用双端队列，每次将入队结点距离和队内距离平均值比较，如果更大则插入至队尾。

* 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
* 自环需要考虑，输入时候排除掉

## Johnson算法(全源最短路)

Johnson就是改进版的V次Dijkstra，解决了Dijkstra不允许有负边的问题，一种显而易见的想法是把所有边加上一个固定值，然后求出最短路后减去路径段数乘以固定值，但该方法是错的，如下图(以边集数组给出)：(1,2,1), (1,3,2), (3,2,-4) 从1到2的最短路应该是1->3->2

但同时加4后就不是最短路了，因为从1到2的走法中，路径种段数不同，会让每种走法加的长度不同，当然就会使得原本是最短路的路变得不是最短路。

正确思路：添加0点，它到其他点都有边且权值为0，然后用bellman-ford或者SPFA求出0点到其余点最短路，到第i个点最短路长度为h[i]

假设任意边u->v的权值为w,重定义每条边的权值为ww(u,v)=w(u,v)+h[u]-h[v]

//**证明所有边权ww(u,v)非负**：

由于最短路性质可知，h[u]+ w(u,v)>= h[v] => h[u]- h[v]- w(u,v)>=0

此时我们证明了重定义后图的所有边权ww(u,v)非负，所以可以用V次迪杰斯特拉。最后，对于一条路径path,假设点为 则迪杰斯特拉求得最短路长度为：

我们知道就是原本要求的最短路，设第i次迪杰斯特拉求得到点j的最短路为dist[i][j]，则答案ans[i][j]= dist[i][j]

时间复杂度为O(VElgV)由于是稀疏图，所以一定强于弗洛伊德的O()

## 弗洛伊德 (全源最短路)：

设dis[i][j]代表点i到j的最短路，dis[i][j]==INF表示i到j不存在路

逐对的枚举两个点，称之为a和b,之后再枚举点,称之为c。

每次进行松弛操作：dis[a][b]=min(dis[a][b],dist[a][c]+ dist[c][b])这里若dist[a][c]或dist[c][b]是INF，说明不存在路，当然就不进行松弛操作

弗洛伊德算法是三重循环嵌套，时间复杂度是V^3。它时间复杂度高一般只是适用V<=300的情况，但是优势是能求出任意两点最短。且时间常数小，在稠密图中比进行V次Dijkstra强

# 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

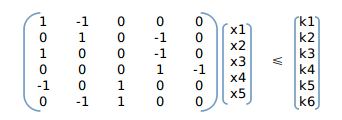
## 差分约束

差分约束系统（system of difference constraints），是求解关于一组变数的特殊不等式组之方法。如果一个系统由n个变量和m个约束条件组成，其中每个约束条件形如xj-xi<=bk(i,j∈[1,n],k∈[1,m]),则称其为差分约束系统(system of difference constraints)。亦即，差分约束系统是求解关于一组变量的特殊不等式组的方法。

通俗一点地说，差分约束系统就是一些不等式的组，而我们的目标是通过给定的约束不等式组求出最大值或者最小值或者差分约束系统是否有解。



把x1 x2 x3…写成矩阵



定义左边的矩阵为A，中间的列向量为 x, 右边的列向量为k。

我们观察一下A可以发现它和我们在离散数学中讲图的时候讲了一个关联矩阵很像。

矩阵A的特点是每行只有一个1和一个-1，其余列都是0。

而关联矩阵的特点是每列只有一个1和一个-1，其余行为0.

即A的转置矩阵就是一个关联矩阵。

那么我们想想，是否可以通过求解图的最短路径来求得不等式组的一个可行解呢？

结论是可以的。

对于每一个差分约束系统，我们根据矩阵A来建立图G（V， E），而G的关联矩阵就是A的转置矩阵，边权存在二维数组W中。假设我们现在要求单源最短路，将各个定点的最短距离存到对应的**dis[v]**中。最后我们要得到的结果一定满足： dis[v] <= dis[u] + w[u][v]. 变形一下得到： dis[v] - dis[u] <= w[u][v].   另dis[v] 为 Xv， dis[u]为Xu，有Xv - Xu <= w[u][v]. 可以看到当前单源最短路问题与差分约束系统问题的等价性。

故可以通过求解单源最短路来求解差分约束问题。当图中存在负环的时候，不等式组无解，否则有解，并且xi的解为dis[i]。若dis[i] = INF，则xi的值为任意值。

**结论是：小于的差分约束等价于最短路**