# 多源最短路

## floyd算法

**int** dist[400][400];

**void** **Floyd**(**int** n)

{

**for** (**int** k **=** 1; k **<=** n; **++**k)

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

**for** (**int** j **=** 1; j **<=** n; **++**j)

dist[i][j] **=** min(dist[i][j], dist[i][k] **+** dist[k][j]);

}

# 单源最短路

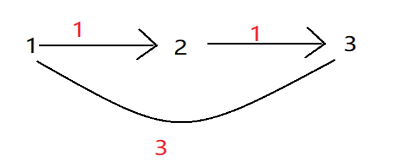
## Bellman-Ford算法

假设**没有负权环**，那这条路径上的点的总个数一定**不大于n**。

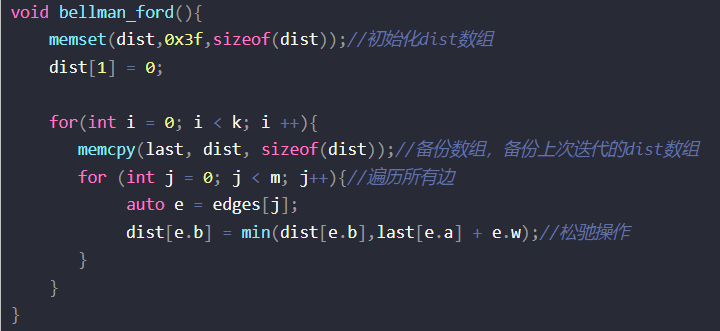
**最短路上的点的总个数一定不大于n，把所有边松弛n-1遍，得到答案。时间复杂度O（mn）m为边数，n为点数**

**负权环的判断：在n-1次松弛之后，再进行一次松弛，如果有dist更新，就有负权环**

**串联问题（对路径数目有要求时发生的问题）**

遍历完第一条边dist[2] = 1,遍历完第二条边dist[3] = 3,遍历第三条边，由于1→2的dist已经确定，在掌握这个信息的前提下，发生串联，dist[3]可以直接松驰，更新为dist[3] = 2

怎么保证不发生串联呢？我们保证更新的时候只用上一次循环的结果就行。所以我们先备份一下。备份之后**backup数组存的就是上一次循环的结果**，我们用上一次循环的结果来更新距离。所以我们这样写dist[b]=min(dist[b],backup[a]+w)来更新距离



k是题目要求最多能经过的路径数

与迪杰斯特拉算法的区别：

1. 迪杰斯特拉算法是借助贪心思想，每次选取一个未处理的最近的结点，去对与他相连接的边进行松弛操作；贝尔曼福特算法是**直接对所有边进行N-1遍松弛操**作。
2. 迪杰斯特拉算法要求边的权值不能是负数；贝尔曼福特算法边的权值可以为负数，并可**检测负权回**路。

链接：[一文弄懂Bellman-Ford(贝尔曼福特算法)](https://blog.csdn.net/Yaoyao2024/article/details/129824109?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522AF3B9065-1EBD-4DCF-A350-5F46FB11B08E%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=AF3B9065-1EBD-4DCF-A350-5F46FB11B08E&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-129824109-null-null.142%5ev100%5econtrol&utm_term=Bellman-Ford&spm=1018.2226.3001.4187)

## SPFA算法（队列优化的ford算法）

-SPFA是如何做到“只更新可能更新的点”的？

1. **只让当前点能到达的点入队**
2. 如果一个点**已经在队列**里，便**不重复入队**
3. 如果一条边**未被更新**，那么它的终点不入队（dist[n]未被更新，n通向的点不会跟着被更新）

随机数据下期望时间复杂度是O(m+nlogn)

**负权环的判断：当一个顶点进队超过n-1次，就代表存在负权环**

**判断2：若再进行一次遍历，还能得到s到某些节点更短的路径的话，则说明存在负环路。**

## Dijkstra算法

Dijkstra算法的流程就是，不断取出**离顶点最近**而**没有被访问过**的点，松弛它和它能到达的所有点。

如何取出离顶点最近的点？如果暴力寻找，那就是朴素的Dijkstra算法，时间复杂度是O(nn)但我们可以采取**堆优化**。具体而言，我们可以用一个**优先队列**（或手写堆，那样更快）来维护所有节点。这样可以实现在O((n+m)logn)的时间内跑完最短路[[1]](https://zhuanlan.zhihu.com/p/96621396#ref_1)。

**正确性证明**

假设每次从未确定最短距离的节点中选择距离最短的节点 u时，dist[u]是从源节点 s 到 u 的最短距离。

**证明**：  
假设在某一步选择了节点 u 进行松弛，此时 dist[u] 为从 s到 u的最短路径距离。如果存在一条更短的路径，假设路径为 s→x→u，则在算法选择 u前应已先选择 x并更新 u，因为 dist[x]≤dist[u]。因此 dist[u] 不可能有更短的路径，选中 u 是正确的。

该算法无法处理有负边的图。

## 打印路径

我们之前只是求出了最短路径长，如果我们要打印具体路径呢？这听起来是一个比较困难的任务，但其实很简单，我们只需要用一个pre[]数组存储每个点的**父节点**即可。（单源最短路的起点是固定的，所以每条路有且仅有一个祖先节点，一步步溯源上去的路径是唯一的。相反，这里不能存**子节点**，因为从源点下去，有很多条最短路径）

每当更新一个点的dist时，顺便更新一下它的pre。这种方法对SPFA和Dij都适用

原博客：[算法学习笔记(6)：最短路问题](https://zhuanlan.zhihu.com/p/96621396)