最短路问题及其求解算法

辛济远

2017 年 12 月 24 日

1 最短路问题

最短路问题是图论中的一个经典算法问题，旨在寻找图中任意两节点之间的最短路径。根据其起点和终点进行分类，可将最短路问题分为单源最短路问题和多源最短问题。其中单源最短路问题指在确定起点后，求解起点到其他点之间的最短路问题。多源最短路问题即求解任意两点之间的最短路，即全图最短路。也可以按照边权的正负进行划分，分为在有负边权的图中求解最短路和在无负边权的图中求解最短路问题。

最短路的求解可以简单通过搜索进行实现。求解最短路问题的经典算法包括 Dijkstra 算法、Floyd 算法、Bellman-Ford 算法、SPFA 算法以及启发式搜索 A* 算法。

2 Dijkstra 算法

Dijkstra 算法主要用于求解求单源、无负权的最短路。时间复杂度为 $O(V * V + E)$。

Dijkstra 算法从某种程度上和最小生成树算法 Prim 类似（毕竟是两个人一起想出来的），均采用了贪心策略。不同之处在于，Dijkstra 算法在计算路程时，对于每一个端点，计算端点到原点的距离，而不是计算端点到已选点集合的距离。

算法特点： Dijkstra 算法使用了广度优先搜索解决赋权有向图或者无向图的单源最短路径问题，算法最终得到一个最短路径树。该算法常用于路由算法或者作为其他图算法的一个子模块。

算法描述： Dijkstra 算法采用的是一种贪心的策略，声明一个数组 dis 来保存源点到各个顶点的最短距离和一个保存已经找到了最短路径的顶点的集合：T。初始时，原点 s 的路径权重被赋为 0 (dis[s] = 0)。

若对于顶点 s 存在能直接到达的边 (s,m)，则把 dis[m] 设为 w(s, m), 同时把所有其他（s 不能直接到达的）顶点的路径长度设为无穷大。初始时，集合 T 只有顶点 s。然后，从 dis 数组选择最小值，则该值就是源点 s 到该值对应的顶点的最短路径，并且把该点加入到 T 中，此时完成一个顶点，然后，我们需要看看新加入的顶点是否可以到达其他顶点并且看看通过该顶点到达其他点的路径长度是否比源点直接到达短，如果是，那么就替换这些顶点在 dis 中的值。

然后，又从 dis 中找出最小值，重复上述动作，直到 T 中包含了图的所有顶点。

下面以图 1 为例，演示一下 Dijkstra 算法的处理流程。(设以 A 为原点) 因为 Dijkstra 算法和 Prim 算法类似，故不再重复。

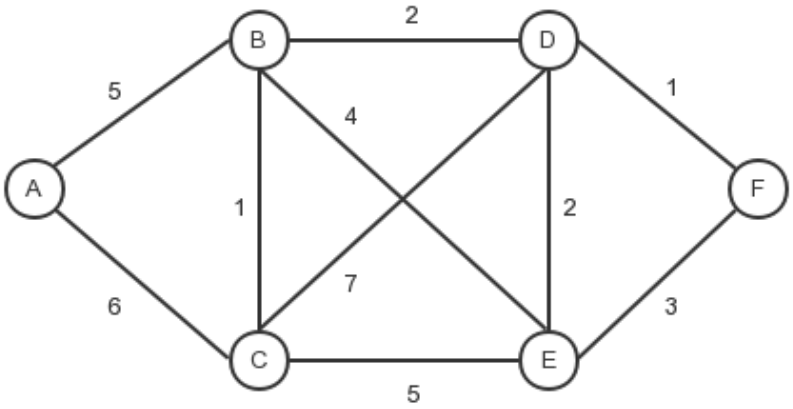


图 1: 无负权图

3 SPFA 算法

SPFA 算法主要适用于单源最短路问题，可以存在负权。时间复杂度为 $O(ke)$ 其中 k 为所有顶点进队的平均次数，可以证明 k 一般小于等于 2。

想要理解 SPFA 算法，需要理解一个非常重要的操作“松弛”。上图展示了一个简单的松弛操作。

图 2: 松弛操作

以 A 为源点，第一次计算距离时，得到 $dis_C = 12, dis_B = 4$。当我们对 B 进行处理时，可以发现路线 A->B->C 的距离小于 A->C 的距离，故我们用 A->B->C 的距离替代原先 A->C 的距离。这样就完成了一次松弛操作。

算法思想： 我们用数组 d 记录每个结点的最短路径估计值，用邻接表来存储图 G。采取的方法是动态逼近法：设立队列用来保存待优化的结点，优化时每次取出队首结点 u，并且用 u 点当前的最短路径估计值对离开 u 点所指向的结点 v 进行松弛操作，如果 v 点的最短路径估计值有所调整，且 v 点不在当前的队列中，就将 v 点放入队尾。这样不断从队列中取出结点来进行松弛操作，直至队列空为止。

建立一个队列，初始时队列里只有起始点，再建立一个表格记录起始点到所有点的最短路径（该表格的初始值要赋为极大值，该点到他本身的路径赋为 0）。然后执行松弛操作，用队列里有的点作

表 1: Dijkstra 算法

i	数组	A	B	C	D	E	F
1	T	√					
	dis	0	∞	∞	∞	∞	∞
2	T	√					
	dis	0	5	6	∞	∞	∞
3	T	√	√				
	dis	0	5	6	∞	∞	∞
4	T	√	√				
	dis	0	5	6	7	9	∞
5	T	√	√	√			
	dis	0	5	6	7	9	∞
6	T	√	√	√	√		
	dis	0	5	6	7	9	∞
7	T	√	√	√	√		
	dis	0	5	6	7	9	8
8	T	√	√	√	√	√	√
	dis	0	5	6	7	9	8

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）

4 Floyd 算法

4.1 计算最短路权值

Floyd 算法主要用于解决多源最短路问题，可以存在负权。时间复杂度为 $O(V^3)$。

算法思想： Floyd 算法采用了动态规划的思想。每次在图上选取一对顶点 i, j，并在图上选取另一个不同于 i, j 的顶点 k，考察顶点 k 是否可以对 i, j 之间的路径进行松弛，如果可以则松弛。根据以上描述，可以得出递推公式

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{如果 } k = 0 \\ min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}) & \text{如果 } k \geq 1 \end{cases}$$

其中 $d_{ij}^{(k)}$ 表示从顶点 i 到顶点 j 且中间点编号不大于 k（$k = 0$ 即不经过中间点）的最短路权值。

Floyd 算法使用邻接矩阵 $\boldsymbol{D}$ 存储任意两点之间的权值，按 k 的增序逐个对顶点对进行松弛。

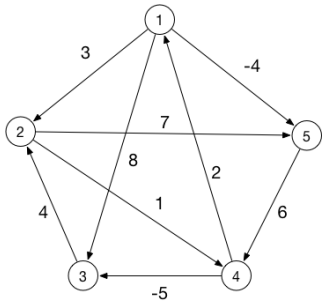
4.2 构造最短路径

我们通过引入先驱矩阵 $\boldsymbol{\Pi}$ 来记录最短路的路径。最短路径 $\pi_{ij}^{(k)}$ 的递推公式为

$$\pi_{ij}^{(0)} = \begin{cases} NULL & \text{如果 } i = j \text{ 或 } w_{ij} = \infty \\ i & \text{如果 } i \neq j \text{ 且 } w_{ij} < \infty \end{cases}$$

$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)} & \text{如果 } d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \\ k & \text{如果 } d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \end{cases}$$

我们用一张图来演示这个过程。



$$D^{(0)} = \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix} \quad \Pi^{(0)} = \begin{bmatrix} NULL & 1 & 1 & NULL & 1 \\ NULL & NULL & NULL & 2 & 2 \\ NULL & 3 & NULL & NULL & NULL \\ 4 & NULL & 4 & NULL & NULL \\ NULL & NULL & NULL & 5 & NULL \end{bmatrix}$$

$$D^{(1)} = \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix} \quad \Pi^{(1)} = \begin{bmatrix} NULL & 1 & 1 & NULL & 1 \\ NULL & NULL & NULL & 2 & 2 \\ NULL & 3 & NULL & NULL & NULL \\ 4 & 1 & 4 & NULL & 1 \\ NULL & NULL & NULL & 5 & NULL \end{bmatrix}$$

$$D^{(2)} = \begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix} \quad \Pi^{(2)} = \begin{bmatrix} NULL & 1 & 1 & 2 & 1 \\ NULL & NULL & NULL & 2 & 2 \\ NULL & 3 & NULL & 2 & 2 \\ 4 & 1 & 4 & NULL & 1 \\ NULL & NULL & NULL & 5 & NULL \end{bmatrix}$$

$$D^{(3)} = \begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix} \quad \Pi^{(3)} = \begin{bmatrix} NULL & 1 & 1 & 2 & 1 \\ NULL & NULL & NULL & 2 & 2 \\ NULL & 3 & NULL & 2 & 2 \\ 4 & 3 & 4 & NULL & 1 \\ NULL & NULL & NULL & 5 & NULL \end{bmatrix}$$

$$D^{(4)} = \begin{bmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix} \quad \Pi^{(4)} = \begin{bmatrix} NULL & 1 & 4 & 2 & 1 \\ 4 & NULL & 4 & 2 & 1 \\ 4 & 3 & NULL & 2 & 1 \\ 4 & 3 & 4 & NULL & 1 \\ 4 & 3 & 4 & 5 & NULL \end{bmatrix}$$

$$D^{(5)} = \begin{bmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix} \quad \Pi^{(5)} = \begin{bmatrix} NULL & 3 & 4 & 5 & 1 \\ 4 & NULL & 4 & 2 & 1 \\ 4 & 3 & NULL & 2 & 1 \\ 4 & 3 & 4 & NULL & 1 \\ 4 & 3 & 4 & 5 & NULL \end{bmatrix}$$