Floyd算法

主讲人：邓哲也

大纲

- 多源最短路径问题
- Floyd算法思想
- Floyd算法时间复杂度
- 有向图的闭包

多源最短路径问题

- 给定一个有向图 G=(V,E)，求出 G 中每对顶点间的最短路径。
- 解决这个问题的方法一般有两种：

1. 轮流以每个顶点为源点，重复执行单源最短路径算法（如：Dijkstra 算法、Bellman-Ford 算法、SPFA 算法）n 次。总的时间复杂度是 $O(n^3)$ 或 $O(n^2m)$ 或 $O(knm)$.
2. 采用 Floyd 算法。 Floyd 算法的时间复杂度也是 $O(n^3)$，但 Floyd 算法形式更直接，代码也极其简洁。

Floyd算法思想

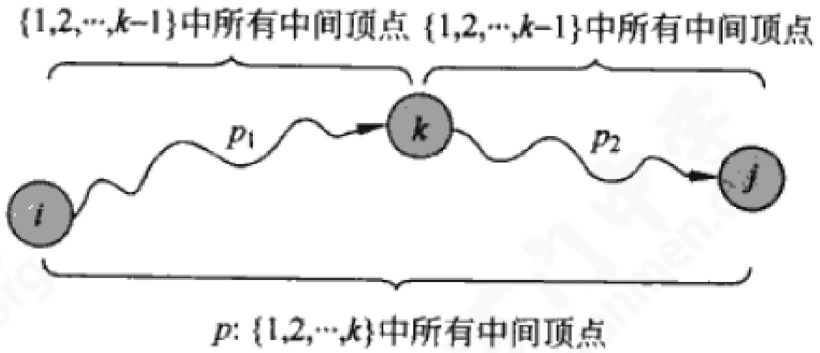
- 在 Floyd 算法中，我们利用最短路径结构中的另一个特征。
- 该算法考虑最短路径上的中间节点，其中简单路径 $p = \langle v_1, v_2, \cdots, v_r \rangle$ 上的中间节点是除 v_1 和 v_r 以外 p 上的任何一个顶点。
- 设 G 的顶点集合为 $V = \{1, 2, \cdots, n\}$。
- 接下来我们将考察中间节点最大值为 k 的所有最短路。

Floyd算法思想

- 对任意一对顶点 i，j，考察从 i 到 j 且中间节点皆属于集合{1, 2, .., k}的所有路径，p 是其中的一条最短路径。
- 如果 k 不是路径 p 的中间顶点，则 p 的所有中间节点都在集合{1, 2, …, k-1} 中。因此从 i 到 j 且满足所有中间节点皆属于集合{1, 2, …, k-1} 的一条最短路径，也同样是从 i 到 j 且满足所有中间顶点皆属于集合 {1, 2,…,k} 的一条最短路径。

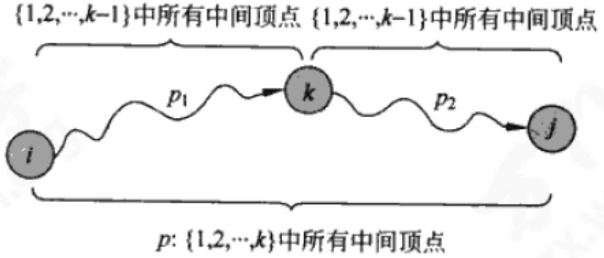
Floyd算法思想

- 如果 k 是路径 p 的中间节点，则 p 可以分解为 i -> k -> j。
- p1 是 i 到 k 的一条最短路径，满足中间节点都属于集合{1, 2, …, k}.
- 因为 k 不是路径 p1 上的中间顶点，所以 p1 是从 i 到 k 的一条最短路径，且其所有中间顶点都属于集合{1, 2, …, k-1}。
- 类似的，p2 也是从 k 到 j 的一条最短路径。



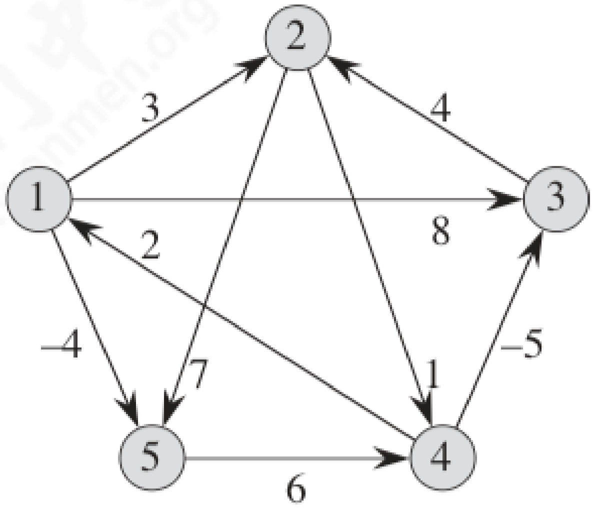
Floyd算法思想

- 基于上述观察，我们可以得出一个最短路径计算的递归公式。
- 令 $d_{ij}^{(k)}$ 为从 i 到 j，且满足所有中间顶点皆属于集合 $\{1, 2, \cdots, k\}$ 的一条最短路径的权值。
- 当 k = 0 时，从顶点 i 到顶点 j 的路径中没有中间节点。
- 也即 $d_{ij}^{(0)} = w_{ij}$
- k ≠ 0 时， $d_{ij}^{(k)} = min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$



Floyd算法思想

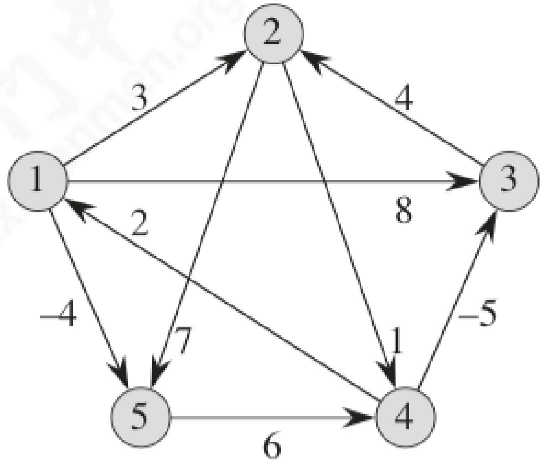
- 接下来我们就以这张图为例，看看 Floyd 算法是如何工作的：



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

Floyd算法思想

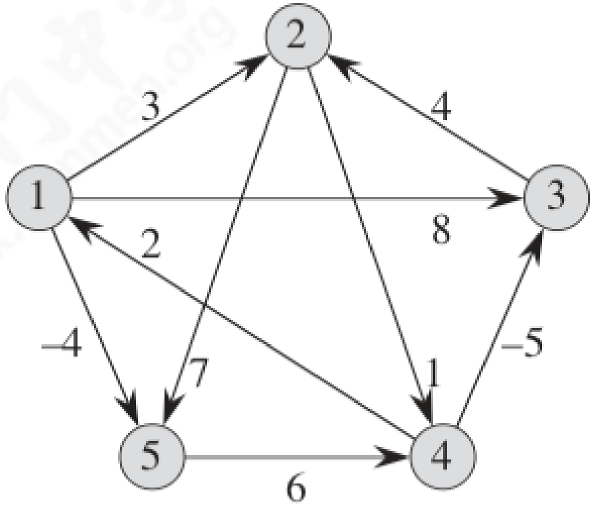
- 第一轮迭代：



$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \boxed{5} & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Floyd算法思想

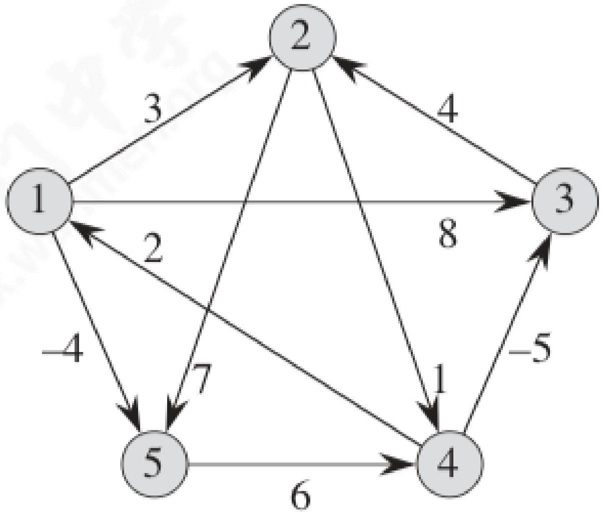
- 第二轮迭代：



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

Floyd算法思想

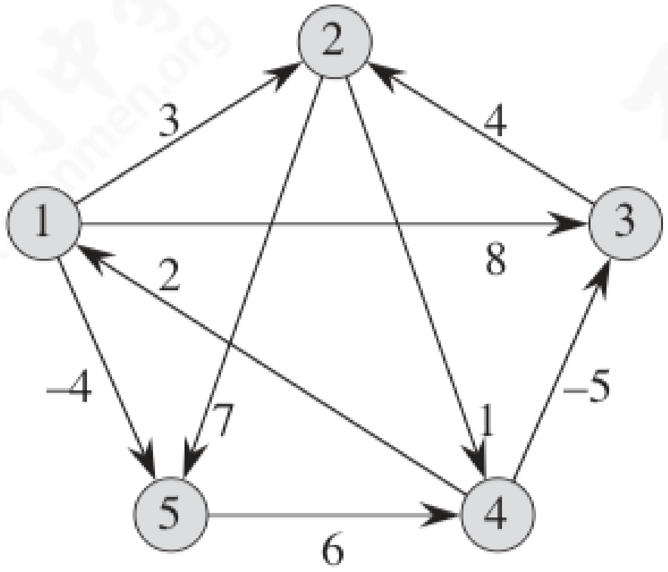
- 第三轮迭代：



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

Floyd算法思想

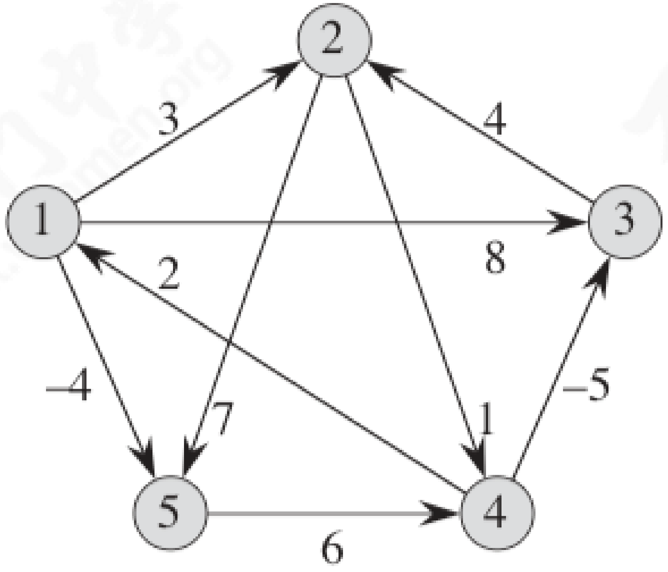
- 第四轮迭代：



$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ \boxed{3} & 0 & \boxed{-4} & 1 & -1 \\ \boxed{7} & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ \boxed{8} & \boxed{5} & \boxed{1} & 6 & 0 \end{pmatrix}$$

Floyd算法思想

- 第五轮迭代，算法结束。



$$D^{(5)} = \begin{pmatrix} 0 & 1 & \boxed{-3} & \boxed{2} & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

Floyd算法复杂度

- 最外层枚举所有 k， 内层枚举每一对(i， j)。
- 一共三层循环，故时间复杂度为$O(n^3)$.

有向图的传递闭包

- 已知一有向图 G=(V,E)，我们希望确定对所有顶点对(i,j)，求出 G 中是否存在一条从 i 到 j 的路径。
- 当 k = 0 时，$d_{ij}^{(0)} = w_{ij}$
- 当 k ≠ 0 时，$d_{ij}^{(k)} = d_{ij}^{(k-1)} \ or \ (d_{ik}^{(k-1)} \ and \ d_{kj}^{(k-1)})$

下节课再见