Dijkstra算法

主讲人：邓哲也

大纲

- 最短路径问题
- Dijkstra算法思想
- Dijkstra算法实现
- Dijkstra算法时间复杂度
- Dijkstra算法正确性

最短路径问题

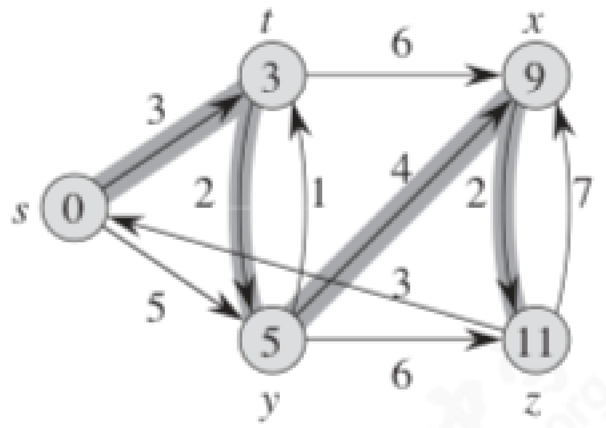
- 最短路径问题（Shortest Path Problem）是有向图和无向图中的一个典型问题。
- 最短路问题要求解的是如果从图中某一顶点到达另一顶点的路径可能不止一条，如何找到这条路径，使得沿着这条路径各边上的权值总和（即从源点到终点的距离）达到最小，这条路径就称为最短路径。

最短路径问题

- 根据边权的范围以及问题求解的需要，最短路问题可以分为以下 4 种情形，分别用不同的算法求解。
- 单源最短路径（固定一个顶点为原点，求源点到其他每个顶点的最短路径）
    - 1. 边权非负：Dijkstra算法。
    - 2. 边权允许为负：Bellman-Ford算法
    - 3. Bellman-Ford算法的改进版本：SPFA算法
- 多源最短路径（计算每个点对之间的最短路）
    - 4. 求所有顶点之间的最短路径：Floyd算法

Dijkstra算法思想

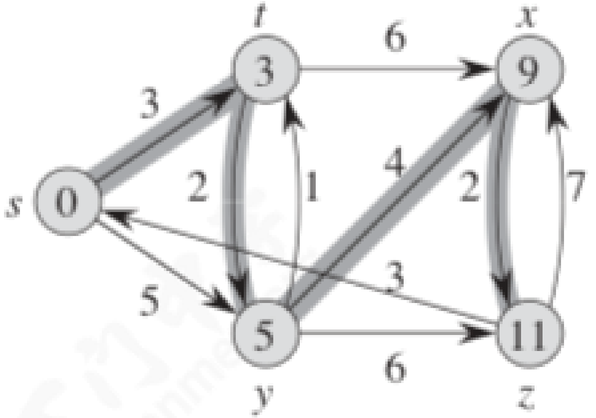
- 问题的提出：给定一个带权有向图 G 和源点 v0，求 v0 到 G 中其他每个顶点的最短路径。限定各边上的权值非负。
- 如右图，s 为源点。
- s 到 t 的最短路径距离为 3，路径为 s -> t
- s 到 y 的最短路径距离为 5，路径为 s -> y 或 s -> t -> y
- s 到 x 的最短路径距离为 9，路径为 s -> t -> x 或 s -> t -> y -> x
- s 到 z 的最短路径距离为 11，路径为 s -> y -> z 或 s -> t -> y -> x -> z



Dijkstra算法思想

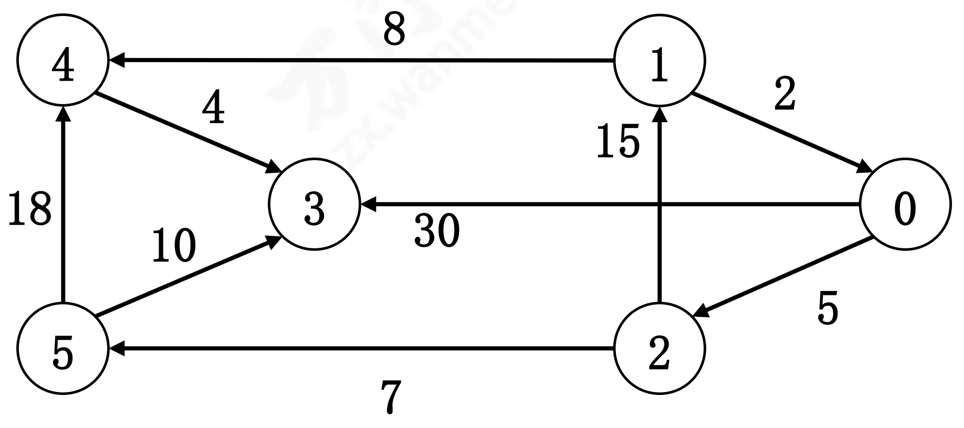
- 为求得这些最短路径，Dijkstra 提出按路径长度的递增次序，逐步产生最短路径的算法。

- 首先求出长度最短的一条最短路径，再参照它求出长度次短的一条最短路径，以此类推，直到从源点 v0 到其他各个顶点的最短路径全部被求出为止。



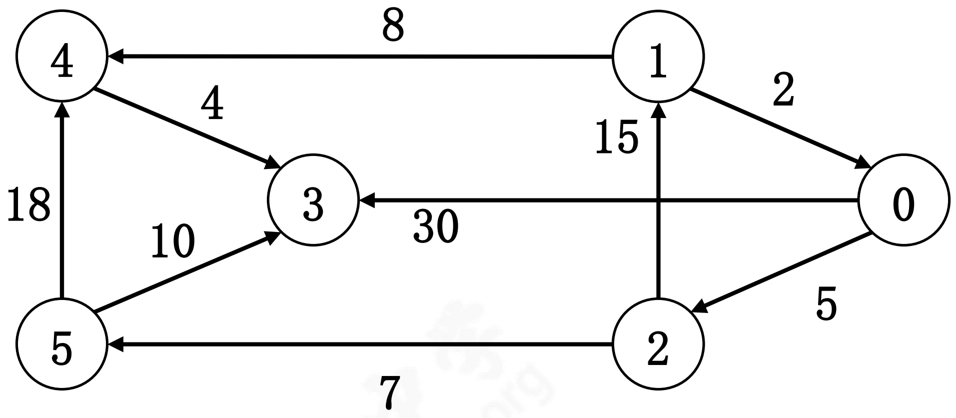
Dijkstra算法思想

- 我们来以这张图为例，完整的描述一遍 Dijkstra 算法的思想。



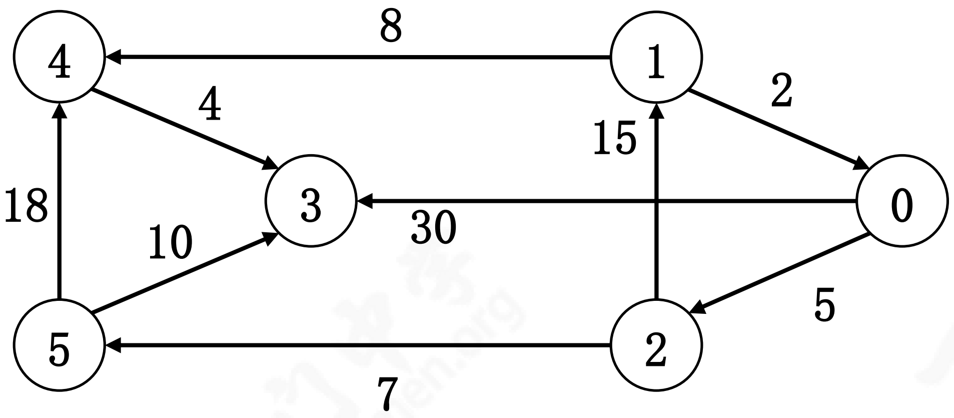
Dijkstra算法思想

- 首先求出长度最短的一条最短路径：0 -> 2，长度为 5（2 号点）.
- 顶点 2 的最短路求出来之后，0 到其他顶点的最短路径长度有可能要改变，比如 0 -> 1 从原来的 ∞ 变成了 20， 0 -> 5 从原来的 ∞ 变成了 12. 这样次短的最短路就是 12（5 号点）.



Dijkstra算法思想

- 5 的最短路求出后，0 到其他点的最短路还会发生改变。比如 0 -> 3 从原来的 30 变为了 22，0 -> 4 也从 ∞ 变为了 28。这时第三短的最短路就确定了，是 20（1 号点）.
- 之后再依次确定第四短的最短路是 22（3 号点）.
- 第五短的最短路是 28 （4 号点）



Dijkstra算法实现方法

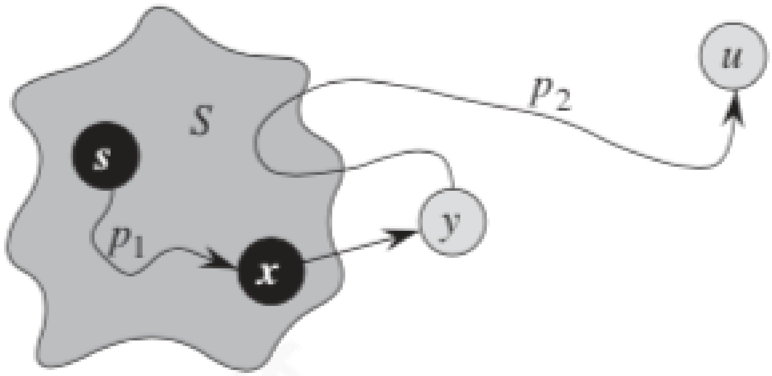
1. 设置两个顶点的集合 S 和 T
   a） S 中存放已找到最短路径的顶点，初始时，集合 S 只有一个顶点，即源点 v0.
   b） T 中存放当前还未找到最短路径的顶点
2. 在 T 集合中选取当前最短的一条最短路径（v0, ..., vk)，从而将 vk 加入到顶点集合 S 中，并且修改源点 v0 到 T 中各顶点的最短路径长度。
3. 重复步骤 2，直到所有的顶点都被加入集合 S 中。

Dijkstra算法时间复杂度分析

- 在 Dijkstra 算法中，最主要的工作是求源点到其他 n - 1 个顶点的最短路径及长度，要把其他 n - 1 个顶点加入到集合 S 中来。
- 每加入一个顶点，首先要在 n - 1 个顶点中判断每个顶点是否属于集合 T， 且最终要在 dist 数组中找元素值最小的顶点
- 然后要对其他 n - 1 个顶点，要判断每个顶点是否属于集合 T 以及是否需要修改 dist 数组元素值。
- 所以时间复杂度为 O(n^2)

Dijkstra算法的正确性

- 可以证明，v0 到 T 中顶点 vk 的最短路径，要么是从 v0 到 vk 的直接路径，要么是从 v0 经 S 中某个顶点 vi 再到 vk 的路径。



思考

- 比较 Dijkstra 算法的思想与 Prim 算法的思想之间的相似之处。

下节课再见