Dijkstra算法堆优化

主讲人：邓哲也

大纲

- Dijkstra算法回顾
- Dijkstra算法堆优化
- Dijkstra算法堆优化实现
- Dijkstra算法堆优化时间复杂度

Dijkstra算法回顾

- 回顾一下在 Dijkstra 算法里，重复做以下 3 步工作：
  1. 在数组 dist[i] 里查找 S[i] != 1，并且 dist[i] 最小的顶点 u。
  2. 将 S[u] 改为 1，表示顶点 u 已经加入进来了。
  3. 修改 T 集合中每个顶点 v_k 的 dist 及 path 数组值。

- 在第一步中，我们是怎么找 dist[i] 最小的顶点 u 的？
- O(n) 枚举。
- 可以做的更好吗？

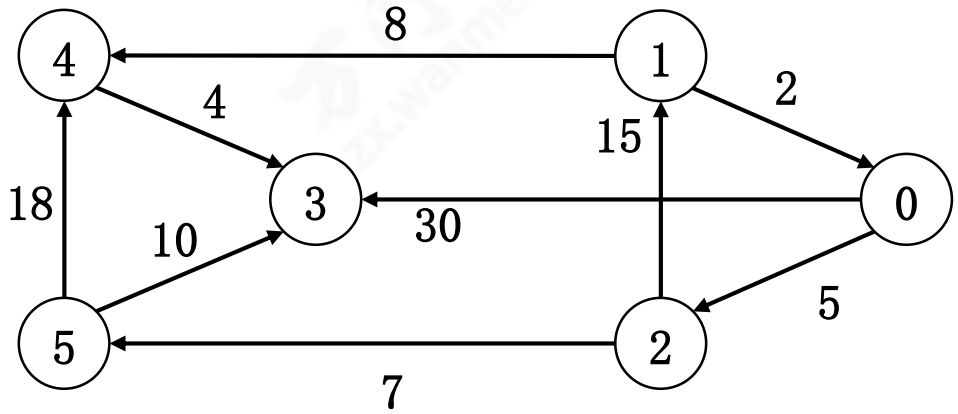
Dijkstra算法堆优化

- 回顾一下在 Dijkstra 算法里，重复做以下 3 步工作：
  1. 在数组 dist[i] 里查找 S[i] != 1，并且 dist[i] 最小的顶点 u。
  2. 将 S[u] 改为 1，表示顶点 u 已经加入进来了。
  3. 修改 T 集合中每个顶点 v_k 的 dist 及 path 数组值。

- 用最小堆来维护 T 集合中的 dist[i]。
- 执行第一步时，堆顶元素就是需要的 u。
- 第二步，直接弹出堆顶元素即可。
- 第三步，不变。

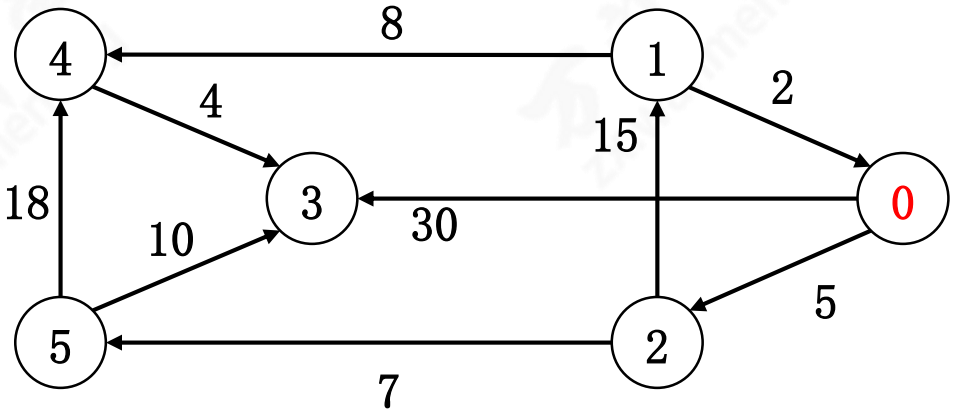
Dijkstra算法堆优化实现

- 让我们回到这张图，看看堆优化算法是怎么运行的。



Dijkstra算法堆优化实现

- 初始状态时，堆中有所有除了 0 以外的点的（dist，u）二元组。

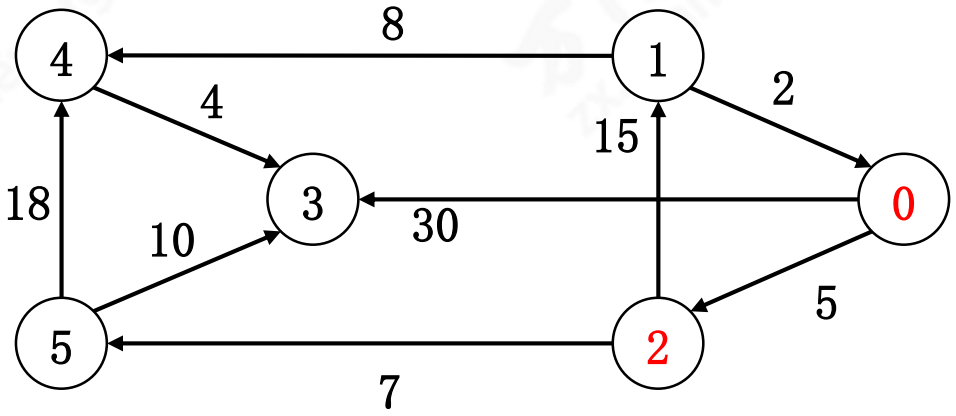


最小堆中元素：
（5，2）
（30，3）
（∞，1）
（∞，4）
（∞，5）

	0	1	2	3	4	5
S	1	0	0	0	0	0
dist	0	∞	5	30	∞	∞
path	-1	-1	0	0	-1	-1

Dijkstra算法堆优化实现

- 求出顶点 2 的最短路径，对 dist 和 path 进行修改，同时更新最小堆。

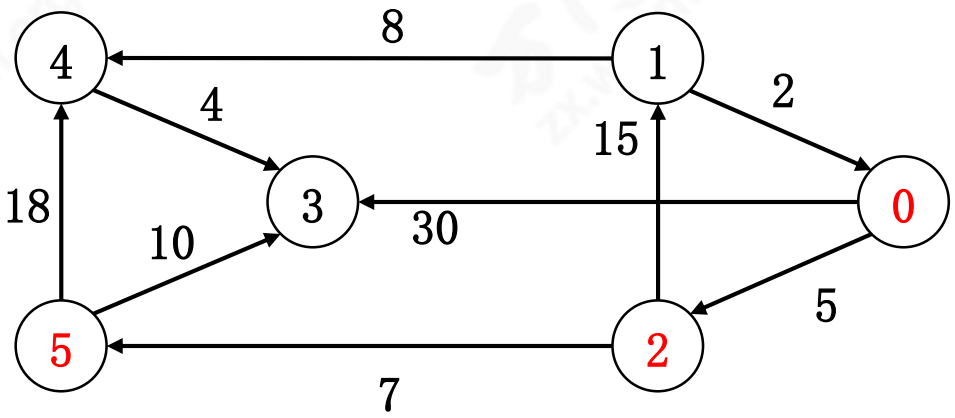


最小堆中元素：
（12，5）
（20，1）
（30，3）
（∞，4）

	0	1	2	3	4	5
S	1	0	1	0	0	0
dist	0	20	5	30	∞	12
path	-1	2	0	0	-1	2

Dijkstra算法堆优化实现

- 求出顶点 5 的最短路径，对 dist 和 path 进行修改，同时更新最小堆。

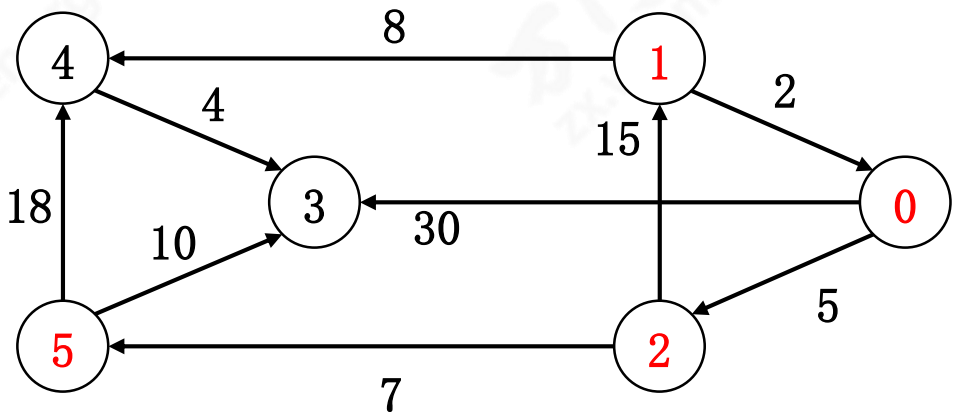


最小堆中元素：
（20，1）
（22，3）
（30，4）

	0	1	2	3	4	5
S	1	0	1	0	0	1
dist	0	20	5	22	30	12
path	-1	2	0	5	5	2

Dijkstra算法堆优化实现

- 求出顶点 1 的最短路径，对 dist 和 path 进行修改，同时更新最小堆。

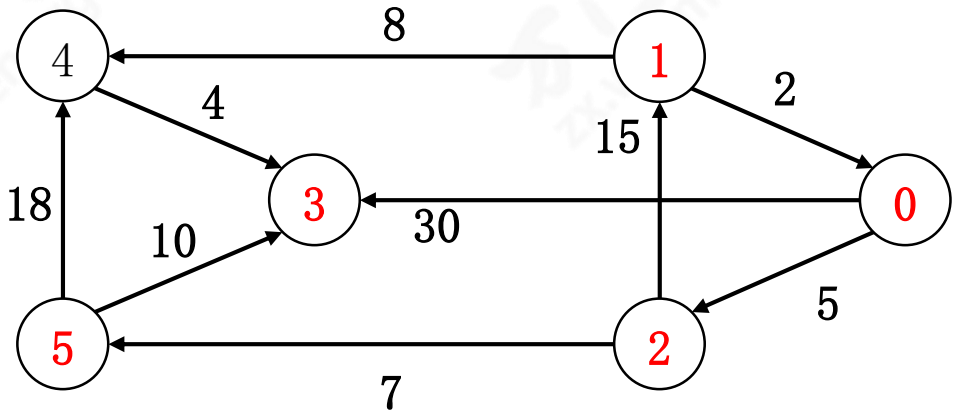


最小堆中元素：
（22，3）
（28，4）

	0	1	2	3	4	5
S	1	1	1	0	0	1
dist	0	20	5	22	28	12
path	-1	2	0	5	1	2

Dijkstra算法堆优化实现

- 求出顶点 3 的最短路径，对 dist 和 path 进行修改，同时更新最小堆。

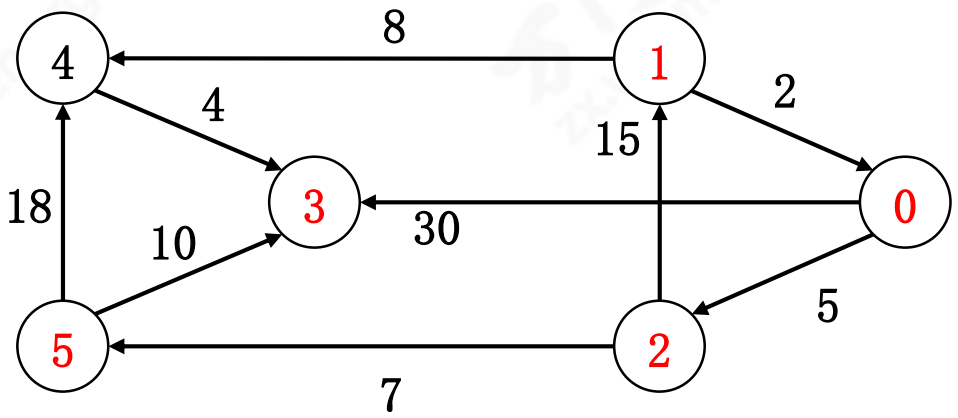


最小堆中元素：
（22，3）
（28，4）

	0	1	2	3	4	5
S	1	1	1	1	0	1
dist	0	20	5	22	28	12
path	-1	2	0	5	1	2

Dijkstra算法堆优化实现

- 求出顶点 3 的最短路径，对 dist 和 path 进行修改，同时更新最小堆。

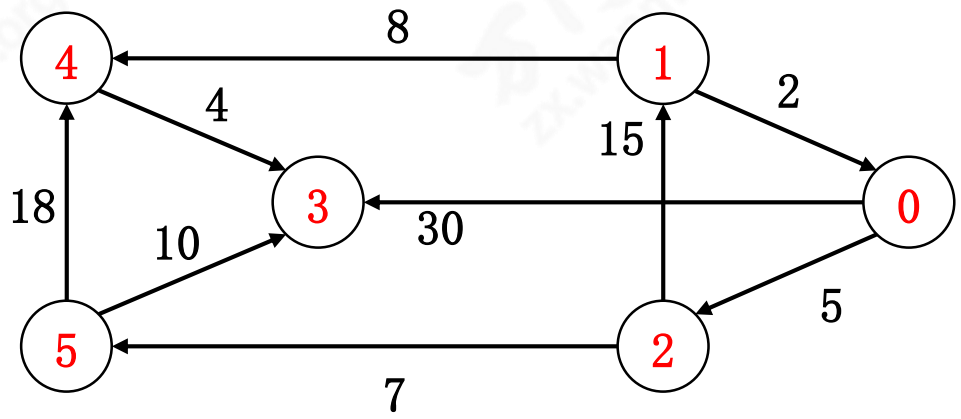


最小堆中元素：
（28，4）

	0	1	2	3	4	5
S	1	1	1	1	0	1
dist	0	20	5	22	28	12
path	-1	2	0	5	1	2

Dijkstra算法堆优化实现

- 求出顶点 4 的最短路径，对 dist 和 path 进行修改，同时更新最小堆。

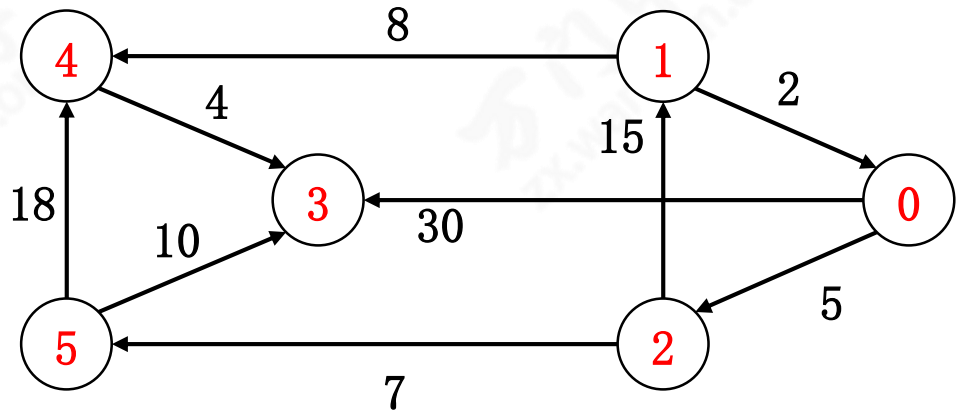


最小堆中元素：

	0	1	2	3	4	5
S	1	1	1	1	1	1
dist	0	20	5	22	28	12
path	-1	2	0	5	1	2

Dijkstra算法堆优化实现

- 至此，0 到其他各顶点的最短路径长度都求完了。



最小堆中元素：

	0	1	2	3	4	5
S	1	1	1	1	0	1
dist	0	20	5	22	28	12
path	-1	2	0	5	1	2

Dijkstra算法堆优化时间复杂度

- 回顾一下在 Dijkstra 算法里，重复做以下 3 步工作：
  1. 在数组 dist[i] 里查找 S[i] != 1，并且 dist[i] 最小的顶点 u。
  2. 将 S[u] 改为 1，表示顶点 u 已经加入进来了。
  3. 修改 T 集合中每个顶点 v_k 的 dist 及 path 数组值。

- 执行第一步时，堆顶元素就是需要的 u。
  - 时间复杂度O(1)
- 第二步，直接弹出堆顶元素即可。
  - 时间复杂度O(log n)

Dijkstra算法堆优化时间复杂度

- 回顾一下在 Dijkstra 算法里，重复做以下 3 步工作：
  1. 在数组 dist[i] 里查找 S[i] != 1，并且 dist[i] 最小的顶点 u。
  2. 将 S[u] 改为 1，表示顶点 u 已经加入进来了。
  3. 修改 T 集合中每个顶点 v_k 的 dist 及 path 数组值。
- 第三步，不变。
  - 注意更新 dist 时，需要把新的 dist 插入堆，一次需要O(log n)。
  - 注意到每个点只会被选入 S 一次，每次会更新这个点所有的出边。因此每一条边只会被更新一次。
  - m 条边总共产生的时间复杂度是O(m log n).

Dijkstra算法堆优化时间复杂度

- 回顾一下在 Dijkstra 算法里，重复做以下 3 步工作：
  1. 在数组 dist[i] 里查找 S[i] != 1，并且 dist[i] 最小的顶点 u。
  2. 将 S[u] 改为 1，表示顶点 u 已经加入进来了。
  3. 修改 T 集合中每个顶点 v_k 的 dist 及 path 数组值。

- 因此总的时间复杂度为 O((n + m) log n)，非常优秀。

下节课再见