Bellmen-Ford算法实现

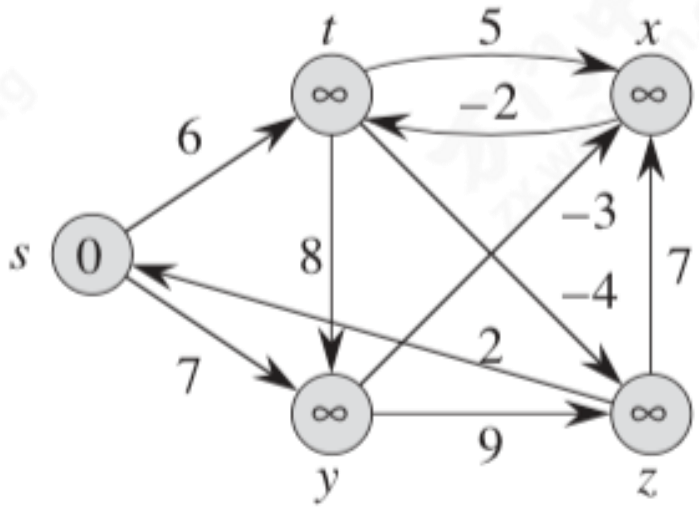
主讲人：邓哲也

Bellman-Ford算法实现

- Bellman-Ford算法在实现时，需要用到以下两个数组：
- （1）使用同一个数组 dist[n] 来存放一系列的 $dist^k[n]$。算法结束时 dist[n] 中存的就是 $dist^{n-1}[u]$.
- （2）path[n] 数组的含义同 dijkstra 算法中的 path 数组。

Bellman-Ford算法实现

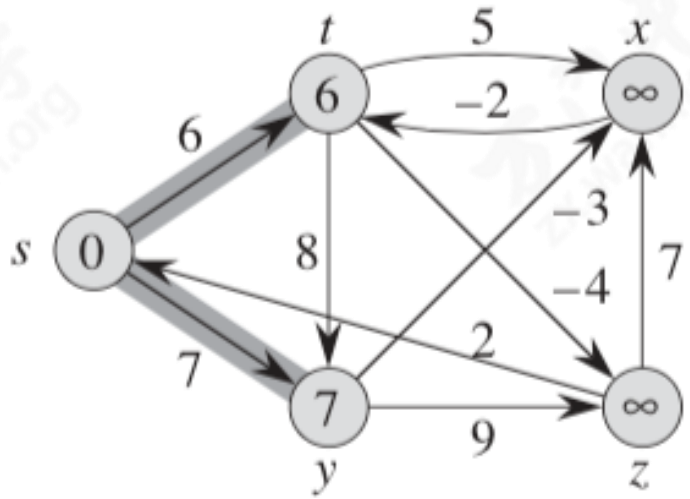
下面就利用这张图，来看看 bellman-ford 算法的执行过程吧！



	s	t	x	y	z
dist	0	∞	∞	∞	∞
path	-1	-1	-1	-1	-1

Bellman-Ford算法实现

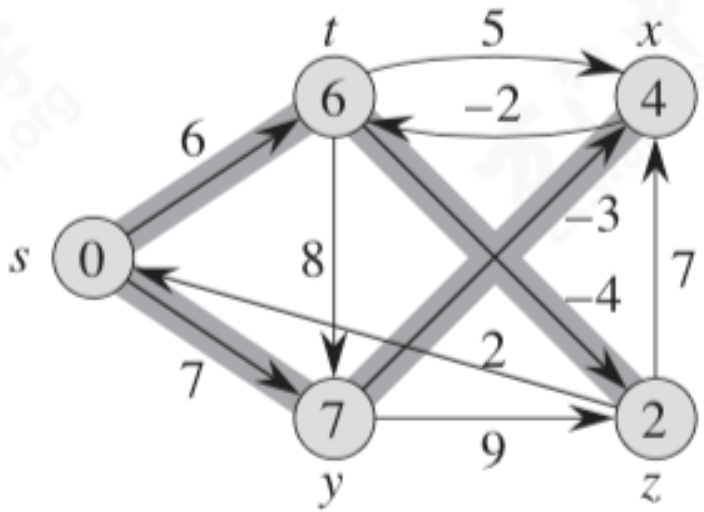
- 第一轮迭代，dist[u] = edge[v_0][u]



	s	t	x	y	z
dist	0	6	∞	7	∞
path	-1	s	-1	s	-1

Bellman-Ford算法实现

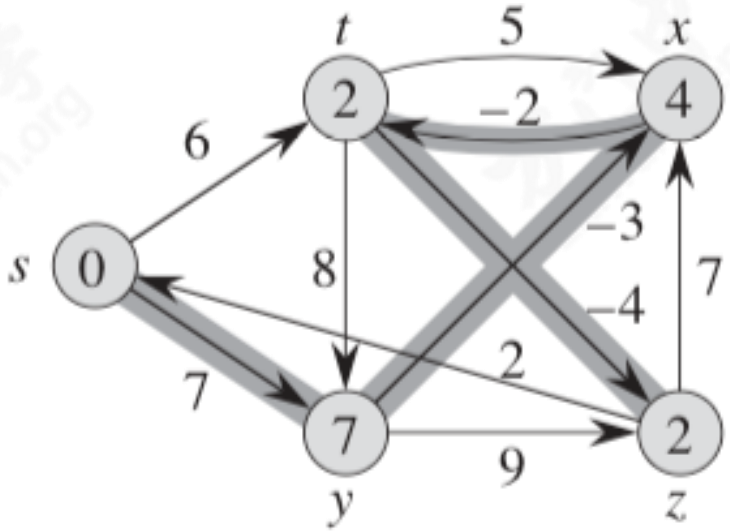
- 第二轮迭代 $dist^2[u] = \min\{dist^1[u], dist^1[j] + edge[j][u]\}$



	s	t	x	y	z
dist	0	6	4	7	2
path	-1	s	y	s	t

Bellman-Ford算法实现

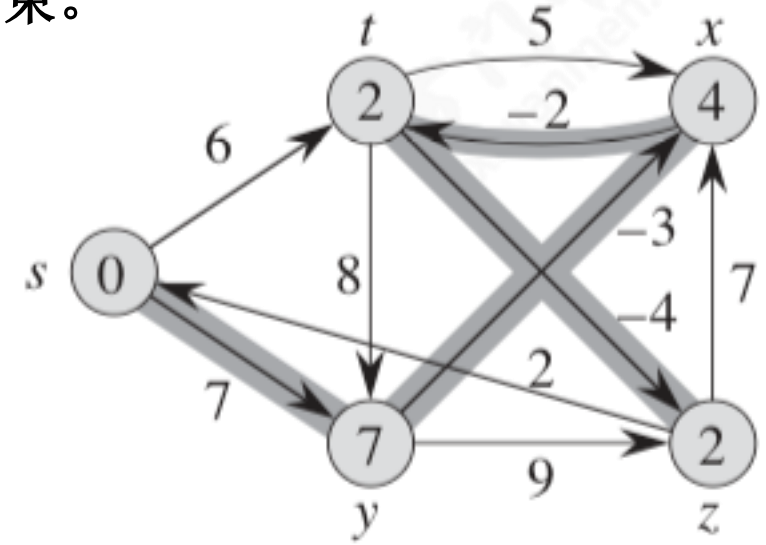
- 第三轮迭代 $dist^3[u] = \min\{dist^2[u], dist^2[j] + edge[j][u]\}$



	s	t	x	y	z
dist	0	2	4	7	2
path	-1	x	y	s	t

Bellman-Ford算法实现

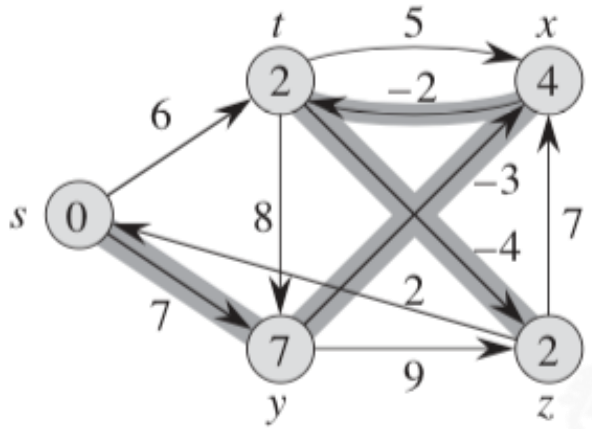
- 第四轮迭代 $dist^4[u] = \min\{dist^3[u], dist^3[j] + edge[j][u]\}$
- 算法结束。



	s	t	x	y	z
dist	0	2	4	7	-2
path	-1	x	y	s	t

Bellman-Ford算法实现

- path 数组的变化与 dijkstra 算法类似。通过倒向追踪可以得到源点到某个点的最短路径。
- 比如 z，沿着 path[z] 倒向追踪，z -> t -> x -> y -> s。
- 故 s 到 z 的最短路为 s -> y -> x -> t -> z.



	s	t	x	y	z
dist	0	2	4	7	-2
path	-1	x	y	s	t

下节课再见