

## 本节内容

# 最短路径

## Dijkstra算法

王道考研/CSKAOYAN.COM

## 迪杰斯特拉



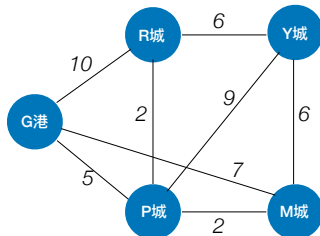
艾兹格·W·迪杰斯特拉  
Edsger Wybe Dijkstra  
(1930~2002)

👑  
1972年图灵奖得主

- 提出“goto 有害理论”——操作系统，虚拟存储技术
- 信号量机制PV原语——操作系统，进程同步
- 银行家算法——操作系统，死锁
- 解决哲学家进餐问题——操作系统，死锁
- Dijkstra最短路径算法——数据结构大题、小题

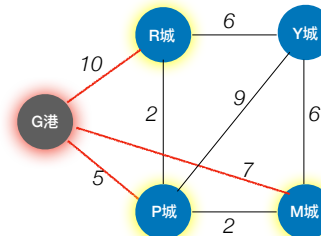
王道考研/CSKAOYAN.COM

## BFS算法的局限性



王道考研/CSKAOYAN.COM

## BFS算法的局限性



带权路径长度——当图是带权图时，一条路径上所有边的权值之和，称为该路径的带权路径长度

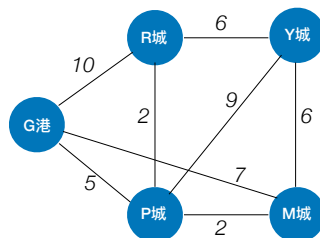
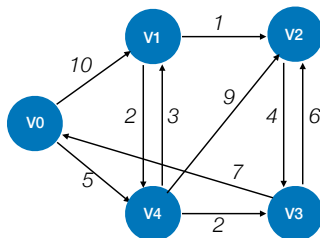
BFS算法求单源最短路径只适用于无权图，或所有边的权值都相同的图

最短路径问题



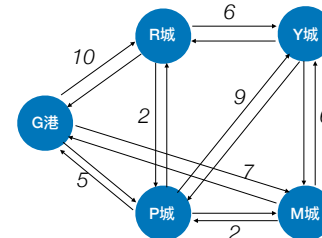
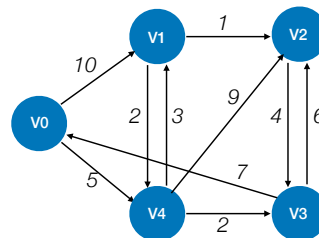
王道考研/CSKAOYAN.COM

## Dijkstra算法



王道考研/CSKAOYAN.COM

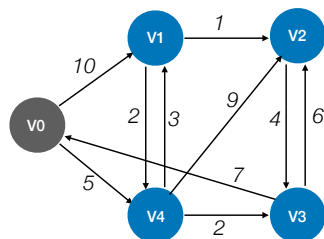
## Dijkstra算法



王道考研/CSKAOYAN.COM

## Dijkstra算法

初始：从V<sub>0</sub>开始，初始化三个数组信息如下



标记各顶点是否已找到最短路径

V0	V1	V2	V3	V4
✓	✗	✗	✗	✗

最短路径长度

final[5]				
0	10	∞	∞	5

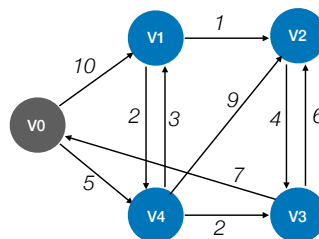
路径上的前驱

path[5]				
-1	0	-1	-1	0

王道考研/CSKAOYAN.COM

## Dijkstra算法

第1轮：循环遍历所有结点，找到还没确定最短路径，且dist 最小的顶点V<sub>i</sub>，令final[i]=ture。



标记各顶点是否已找到最短路径

V0	V1	V2	V3	V4
✓	✗	✗	✗	✗

最短路径长度

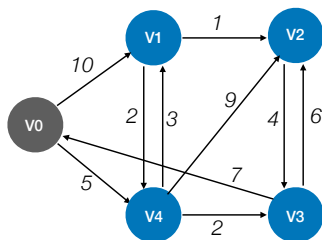
final[5]				
0	10	∞	∞	5

路径上的前驱

path[5]				
-1	0	-1	-1	0

王道考研/CSKAOYAN.COM

## Dijkstra算法



第1轮：循环遍历所有结点，找到还没确定最短路径，且dist 最小的顶点 $V_i$ ，令 $final[i]=ture$ 。

标记各顶点是否已找到最短路径

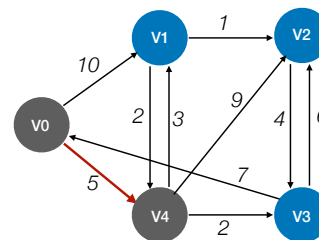
最短路径长度

路径上的前驱

	V0	V1	V2	V3	V4
$final[5]$	✓	✗	✗	✗	✗
$dist[5]$	0	10	∞	∞	5
$path[5]$	-1	0	-1	-1	0

王道考研/CSKAOYAN.COM

## Dijkstra算法



第1轮：循环遍历所有结点，找到还没确定最短路径，且dist 最小的顶点 $V_i$ ，令 $final[i]=ture$ 。

标记各顶点是否已找到最短路径

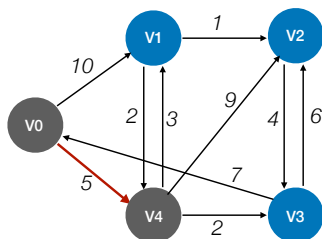
最短路径长度

路径上的前驱

	V0	V1	V2	V3	V4
$final[5]$	✓	✗	✗	✗	✓
$dist[5]$	0	10	∞	∞	5
$path[5]$	-1	0	-1	-1	0

王道考研/CSKAOYAN.COM

## Dijkstra算法



第1轮：循环遍历所有结点，找到还没确定最短路径，且dist 最小的顶点 $V_i$ ，令 $final[i]=ture$ 。

标记各顶点是否已找到最短路径

最短路径长度

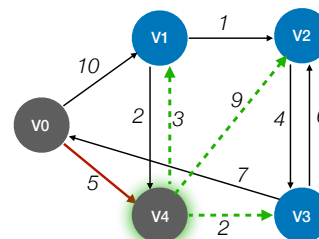
路径上的前驱

	V0	V1	V2	V3	V4
$final[5]$	✓	✗	✗	✗	✓
$dist[5]$	0	10	∞	∞	5
$path[5]$	-1	0	-1	-1	0

检查所有邻接自  $V_i$  的顶点，若其  $final$  值为false，则更新  $dist$  和  $path$  信息

王道考研/CSKAOYAN.COM

## Dijkstra算法



第1轮：循环遍历所有结点，找到还没确定最短路径，且dist 最小的顶点 $V_i$ ，令 $final[i]=ture$ 。

标记各顶点是否已找到最短路径

最短路径长度

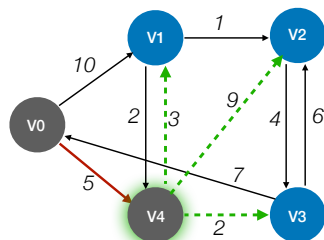
路径上的前驱

	V0	V1	V2	V3	V4
$final[5]$	✓	✗	✗	✗	✓
$dist[5]$	0	10	∞	∞	5
$path[5]$	-1	0	-1	-1	0

检查所有邻接自  $V_i$  的顶点，若其  $final$  值为false，则更新  $dist$  和  $path$  信息

王道考研/CSKAOYAN.COM

## Dijkstra算法



第1轮：循环遍历所有结点，找到还没确定最短路径，且dist 最小的顶点 $V_i$ ，令 $final[i]=ture$ 。

标记各顶点是否已找到最短路径

最短路径长度

路径上的前驱

	V0	V1	V2	V3	V4
$final[5]$	✓	✗	✗	✗	✓

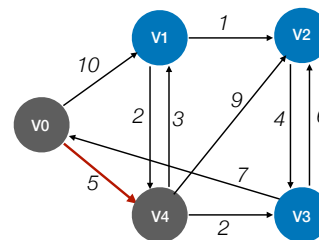
	V0	V1	V2	V3	V4
$dist[5]$	0	8	14	7	5

	V0	V1	V2	V3	V4
$path[5]$	-1	4	4	4	0

检查所有邻接自  $V_i$  的顶点，若其  $final$  值为false，则更新  $dist$  和  $path$  信息

王道考研/CSKAOYAN.COM

## Dijkstra算法



第2轮：循环遍历所有结点，找到还没确定最短路径，且dist 最小的顶点 $V_i$ ，令 $final[i]=ture$ 。

标记各顶点是否已找到最短路径

最短路径长度

路径上的前驱

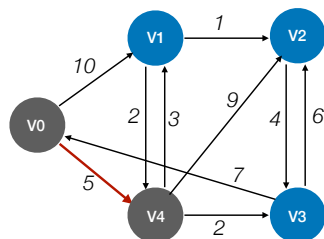
	V0	V1	V2	V3	V4
$final[5]$	✓	✗	✗	✗	✓

	V0	V1	V2	V3	V4
$dist[5]$	0	8	14	7	5

	V0	V1	V2	V3	V4
$path[5]$	-1	4	4	4	0

王道考研/CSKAOYAN.COM

## Dijkstra算法



第2轮：循环遍历所有结点，找到还没确定最短路径，且dist 最小的顶点 $V_i$ ，令 $final[i]=ture$ 。

标记各顶点是否已找到最短路径

最短路径长度

路径上的前驱

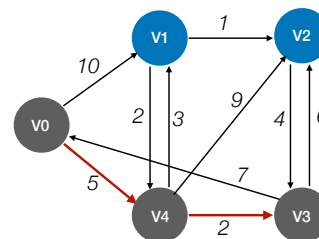
	V0	V1	V2	V3	V4
$final[5]$	✓	✗	✗	✗	✓

	V0	V1	V2	V3	V4
$dist[5]$	0	8	14	7	5

	V0	V1	V2	V3	V4
$path[5]$	-1	4	4	4	0

王道考研/CSKAOYAN.COM

## Dijkstra算法



第2轮：循环遍历所有结点，找到还没确定最短路径，且dist 最小的顶点 $V_i$ ，令 $final[i]=ture$ 。

标记各顶点是否已找到最短路径

最短路径长度

路径上的前驱

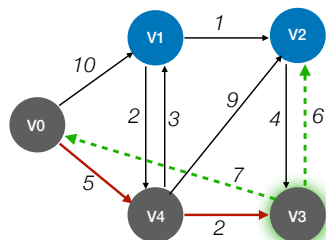
	V0	V1	V2	V3	V4
$final[5]$	✓	✗	✗	✓	✓

	V0	V1	V2	V3	V4
$dist[5]$	0	8	14	7	5

	V0	V1	V2	V3	V4
$path[5]$	-1	4	4	4	0

王道考研/CSKAOYAN.COM

## Dijkstra算法



第2轮：循环遍历所有结点，找到还没确定最短路径，且dist 最小的顶点 $V_i$ ，令 $final[i]=ture$ 。

标记各顶点是否已找到最短路径

最短路径长度

路径上的前驱

	V0	V1	V2	V3	V4
$final[5]$	✓	✗	✗	✓	✓

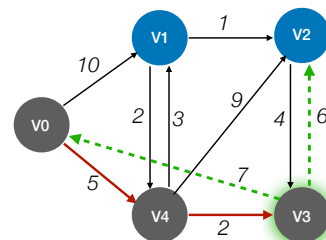
	V0	V1	V2	V3	V4
$dist[5]$	0	8	14	7	5

	V0	V1	V2	V3	V4
$path[5]$	-1	4	4	4	0

检查所有邻接自  $V_i$  的顶点，若其  $final$  值为false，则更新  $dist$  和  $path$  信息

王道考研/CSKAOYAN.COM

## Dijkstra算法



第2轮：循环遍历所有结点，找到还没确定最短路径，且dist 最小的顶点 $V_i$ ，令 $final[i]=ture$ 。

标记各顶点是否已找到最短路径

最短路径长度

路径上的前驱

	V0	V1	V2	V3	V4
$final[5]$	✓	✗	✗	✓	✓

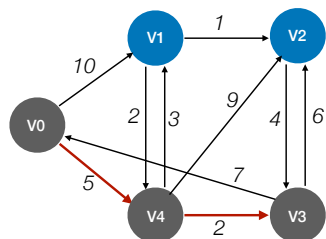
	V0	V1	V2	V3	V4
$dist[5]$	0	8	13	7	5

	V0	V1	V2	V3	V4
$path[5]$	-1	4	3	4	0

检查所有邻接自  $V_i$  的顶点，若其  $final$  值为false，则更新  $dist$  和  $path$  信息

王道考研/CSKAOYAN.COM

## Dijkstra算法



第3轮：循环遍历所有结点，找到还没确定最短路径，且dist 最小的顶点 $V_i$ ，令 $final[i]=ture$ 。

标记各顶点是否已找到最短路径

最短路径长度

路径上的前驱

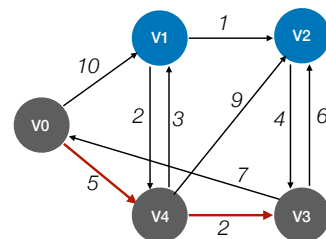
	V0	V1	V2	V3	V4
$final[5]$	✓	✗	✗	✓	✓

	V0	V1	V2	V3	V4
$dist[5]$	0	8	13	7	5

	V0	V1	V2	V3	V4
$path[5]$	-1	4	3	4	0

王道考研/CSKAOYAN.COM

## Dijkstra算法



第3轮：循环遍历所有结点，找到还没确定最短路径，且dist 最小的顶点 $V_i$ ，令 $final[i]=ture$ 。

标记各顶点是否已找到最短路径

最短路径长度

路径上的前驱

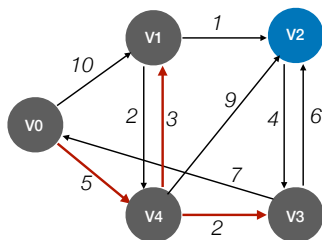
	V0	V1	V2	V3	V4
$final[5]$	✓	✗	✗	✓	✓

	V0	V1	V2	V3	V4
$dist[5]$	0	8	13	7	5

	V0	V1	V2	V3	V4
$path[5]$	-1	4	3	4	0

王道考研/CSKAOYAN.COM

## Dijkstra算法



第3轮：循环遍历所有结点，找到还没确定最短路径，且dist 最小的顶点 $V_i$ ，令 $final[i]=ture$ 。

标记各顶点是否已找到最短路径

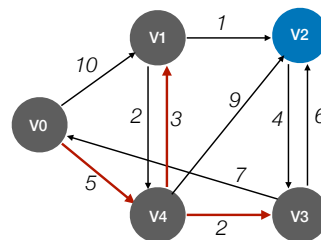
最短路径长度

路径上的前驱

	V0	V1	V2	V3	V4
$final[5]$	✓	✓	✗	✓	✓
$dist[5]$	0	8	13	7	5
$path[5]$	-1	4	3	4	0

王道考研/CSKAOYAN.COM

## Dijkstra算法



第3轮：循环遍历所有结点，找到还没确定最短路径，且dist 最小的顶点 $V_i$ ，令 $final[i]=ture$ 。

标记各顶点是否已找到最短路径

最短路径长度

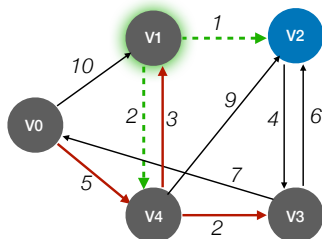
路径上的前驱

	V0	V1	V2	V3	V4
$final[5]$	✓	✓	✗	✓	✓
$dist[5]$	0	8	13	7	5
$path[5]$	-1	4	3	4	0

检查所有邻接自  $V_i$  的顶点，若其  $final$  值为false，则更新  $dist$  和  $path$  信息

王道考研/CSKAOYAN.COM

## Dijkstra算法



第3轮：循环遍历所有结点，找到还没确定最短路径，且dist 最小的顶点 $V_i$ ，令 $final[i]=ture$ 。

标记各顶点是否已找到最短路径

最短路径长度

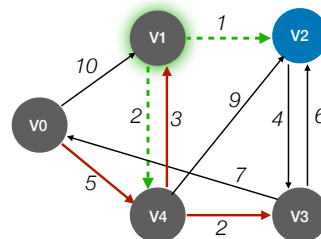
路径上的前驱

	V0	V1	V2	V3	V4
$final[5]$	✓	✓	✗	✓	✓
$dist[5]$	0	8	13	7	5
$path[5]$	-1	4	3	4	0

检查所有邻接自  $V_i$  的顶点，若其  $final$  值为false，则更新  $dist$  和  $path$  信息

王道考研/CSKAOYAN.COM

## Dijkstra算法



第3轮：循环遍历所有结点，找到还没确定最短路径，且dist 最小的顶点 $V_i$ ，令 $final[i]=ture$ 。

标记各顶点是否已找到最短路径

最短路径长度

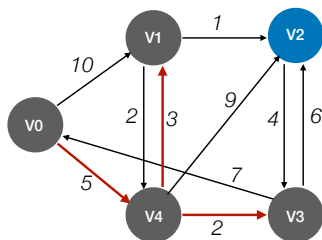
路径上的前驱

	V0	V1	V2	V3	V4
$final[5]$	✓	✓	✗	✓	✓
$dist[5]$	0	8	9	7	5
$path[5]$	-1	4	1	4	0

检查所有邻接自  $V_i$  的顶点，若其  $final$  值为false，则更新  $dist$  和  $path$  信息

王道考研/CSKAOYAN.COM

## Dijkstra算法



第4轮：循环遍历所有结点，找到还没确定最短路径，且dist 最小的顶点 $V_i$ ，令 $final[i]=ture$ 。

标记各顶点是否已找到最短路径

	V0	V1	V2	V3	V4
$final[5]$	✓	✓	✗	✓	✓

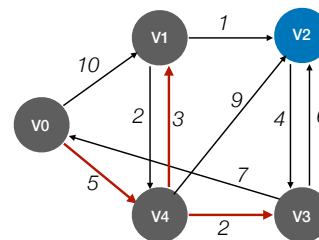
最短路径长度

$dist[5]$	0	8	9	7	5
-----------	---	---	---	---	---

路径上的前驱

$path[5]$	-1	4	1	4	0
-----------	----	---	---	---	---

## Dijkstra算法



第4轮：循环遍历所有结点，找到还没确定最短路径，且dist 最小的顶点 $V_i$ ，令 $final[i]=ture$ 。

标记各顶点是否已找到最短路径

	V0	V1	V2	V3	V4
$final[5]$	✓	✓	✗	✓	✓

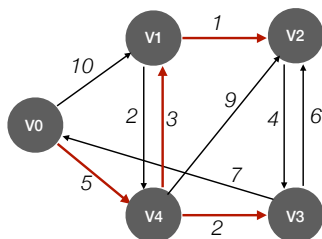
最短路径长度

$dist[5]$	0	8	9	7	5
-----------	---	---	---	---	---

路径上的前驱

$path[5]$	-1	4	1	4	0
-----------	----	---	---	---	---

## Dijkstra算法



第4轮：循环遍历所有结点，找到还没确定最短路径，且dist 最小的顶点 $V_i$ ，令 $final[i]=ture$ 。

标记各顶点是否已找到最短路径

	V0	V1	V2	V3	V4
$final[5]$	✓	✓	✓	✓	✓

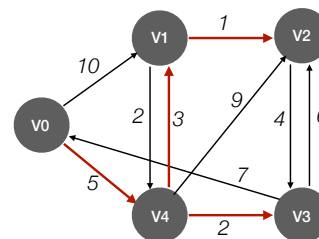
最短路径长度

$dist[5]$	0	8	9	7	5
-----------	---	---	---	---	---

路径上的前驱

$path[5]$	-1	4	1	4	0
-----------	----	---	---	---	---

## Dijkstra算法



第4轮：循环遍历所有结点，找到还没确定最短路径，且dist 最小的顶点 $V_i$ ，令 $final[i]=ture$ 。

标记各顶点是否已找到最短路径

	V0	V1	V2	V3	V4
$final[5]$	✓	✓	✓	✓	✓

最短路径长度

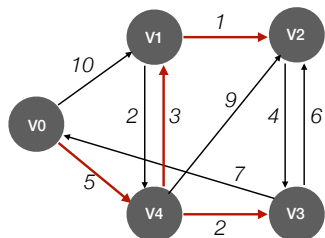
$dist[5]$	0	8	9	7	5
-----------	---	---	---	---	---

路径上的前驱

$path[5]$	-1	4	1	4	0
-----------	----	---	---	---	---

检查所有邻接自  $V_i$  的顶点，若其  $final$  值为false，则更新  $dist$  和  $path$  信息

### 如何使用数组信息?



标记各顶点是否已找到最短路径

最短路径长度

路径上的前驱

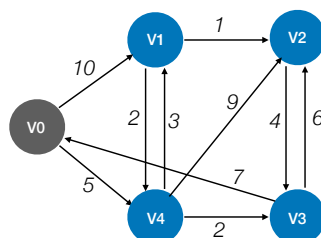
	V0	V1	V2	V3	V4
final[5]	✓	✓	✓	✓	✓
dist[5]	0	8	9	7	5
path[5]	-1	4	1	4	0

V0到V2的最短(带权)路径长度为:  $dist[2] = 9$

通过 path[] 可知, V0到V2的最短(带权)路径:  $V2 \leftarrow V1 \leftarrow V4 \leftarrow V0$

王道考研/CSKAQYAN.COM

### Dijkstra算法的时间复杂度



标记各顶点是否已找到最短路径

最短路径长度

路径上的前驱

	V0	V1	V2	V3	V4
final[5]	✓	✗	✗	✗	✗
dist[5]	0	10	∞	∞	5
path[5]	-1	0	-1	-1	0

初始: 若从V0开始, 令 final[0]=ture; dist[0]=0; path[0]=-1。

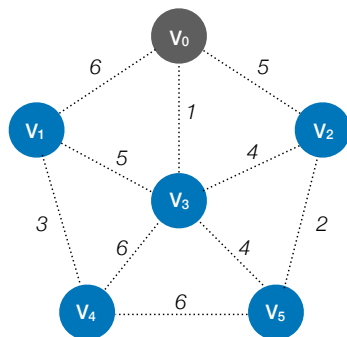
其余顶点final[k]=false; dist[k]=arcs[0][k]; path[k]= (arcs[0][k]==∞) ? -1 : 0

时间复杂度:  
 $O(n^2)$ 即 $O(|V|^2)$

n-1轮处理: 循环遍历所有顶点, 找到还没确定最短路径, 且dist最小的顶点 $V_i$ , 令final[i]=ture。并检查所有邻接自 $V_i$ 的顶点, 对于邻接自 $V_i$ 的顶点 $V_j$ , 若 final[j]==false 且  $dist[i]+arcs[i][j] < dist[j]$ , 则令  $dist[j]=dist[i]+arcs[i][j]$ ; path[j]=i。(注: arcs[i][j]表示 $V_i$ 到 $V_j$ 的弧的权值)

王道考研/CSKAQYAN.COM

### 对比: Prim 算法的实现思想



总时间复杂度  
 $O(n^2)$ , 即 $O(|V|^2)$

从V0开始, 总共需要 n-1 轮处理

每一轮处理: 循环遍历所有个结点, 找到lowCost最低的, 且还没加入树的顶点。

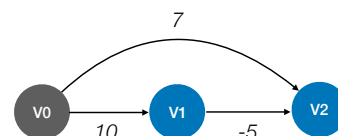
再次循环遍历, 更新还没加入的各个顶点的lowCost值

每一轮时间复杂度 $O(2n)$

	V0	V1	V2	V3	V4	V5
isJoin[6]	✓	✗	✗	✗	✗	✗
lowCost[6]	0	6	5	1	∞	∞

王道考研/CSKAQYAN.COM

### 用于负权值带权图



标记各顶点是否已找到最短路径

最短路径长度

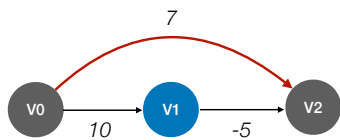
路径上的前驱

	V0	V1	V2
final[3]	✓	✗	✗
dist[3]	0	10	7
path[3]	-1	0	0

王道考研/CSKAQYAN.COM



### 用于负权值带权图



标记各顶点是否  
已找到最短路径

最短路  
径长度

路径上  
的前驱

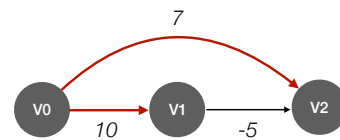
$final[3]$

$dist[3]$

$path[3]$

V0	V1	V2
✓	✗	✓
0	10	7
-1	0	0

### 用于负权值带权图



标记各顶点是否  
已找到最短路径

最短路  
径长度

路径上  
的前驱

$final[3]$

$dist[3]$

$path[3]$

V0	V1	V2
✓	✓	✓
0	10	7
-1	0	0

事实上V0到V2的最短带权路径长度为 5

结论: Dijkstra 算法不适用于有负权值的带权图