本节内容

最短路径

Floyd算法

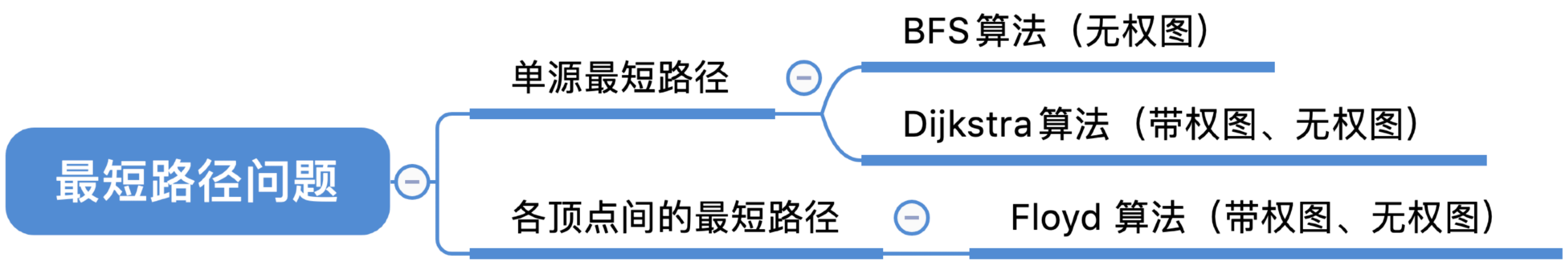
Robert W．Floyd

罗伯特·弗洛伊德

（1936－2001）Robert W．Floyd

1978年图灵奖得主

- Floyd算法（Floyd-Warshall算法 ）
- 堆排序算法



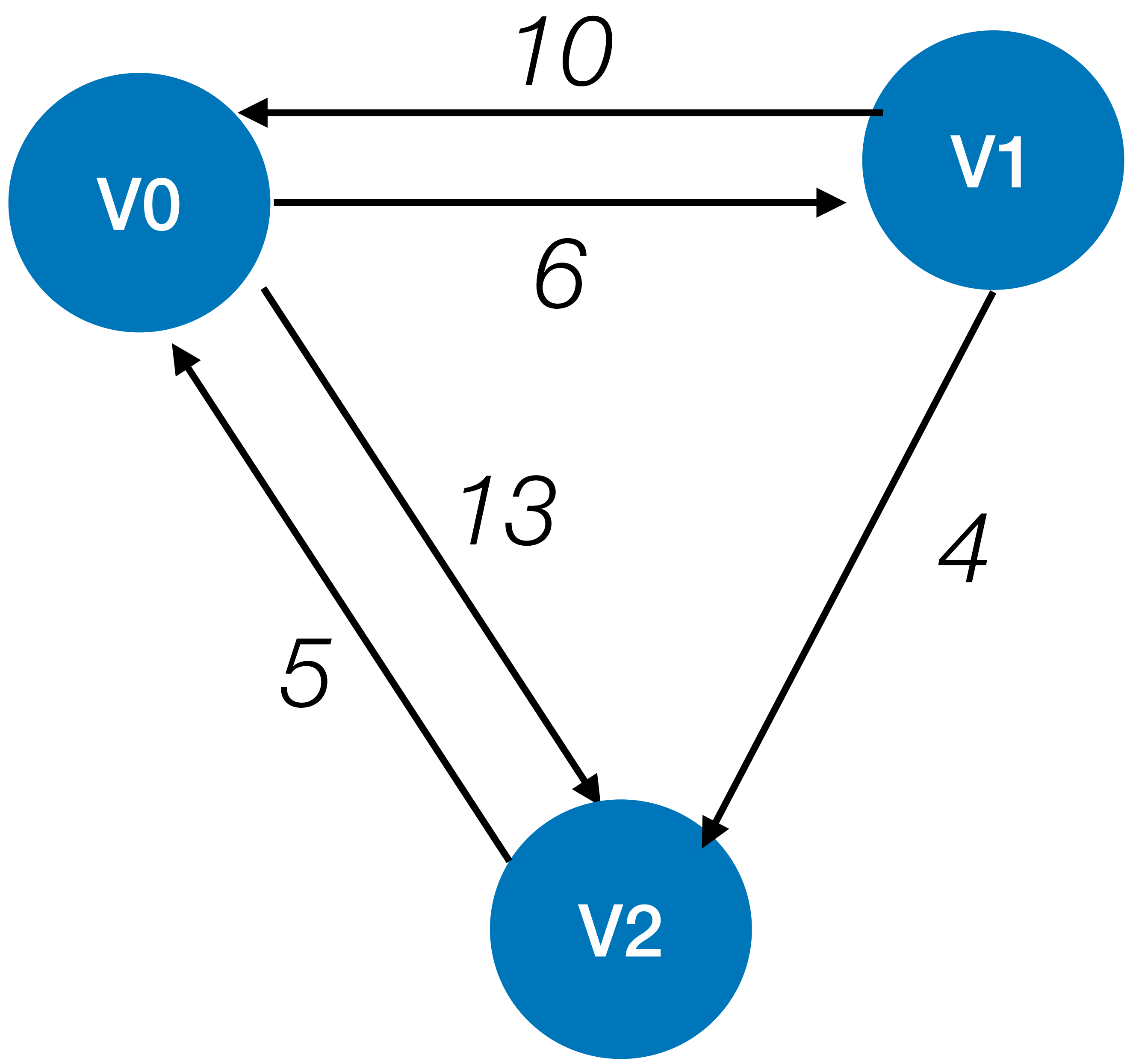
Floyd算法

Floyd算法：求出每一对顶点之间的最短路径

使用动态规划思想，将问题的求解分为多个阶段

对于n个顶点的图G，求任意一对顶点 Vi —> Vj 之间的最短路径可分为如下几个阶段：
#初始：不允许在其他顶点中转，最短路径是?
#0：若允许在 V_0 中转，最短路径是?
#1：若允许在 V_0、V_1 中转，最短路径是?
#2：若允许在 V_0、V_1、V_2 中转，最短路径是?
…
#n-1：若允许在 V_0、V_1、V_2 …… V_{n-1} 中转，最短路径是?

Floyd算法



目前来看，各顶点间的最短路径长度

$A^{(-1)}$ =

	V0	V1	V2
V0	0	6	13
V1	10	0	4
V2	5	∞	0

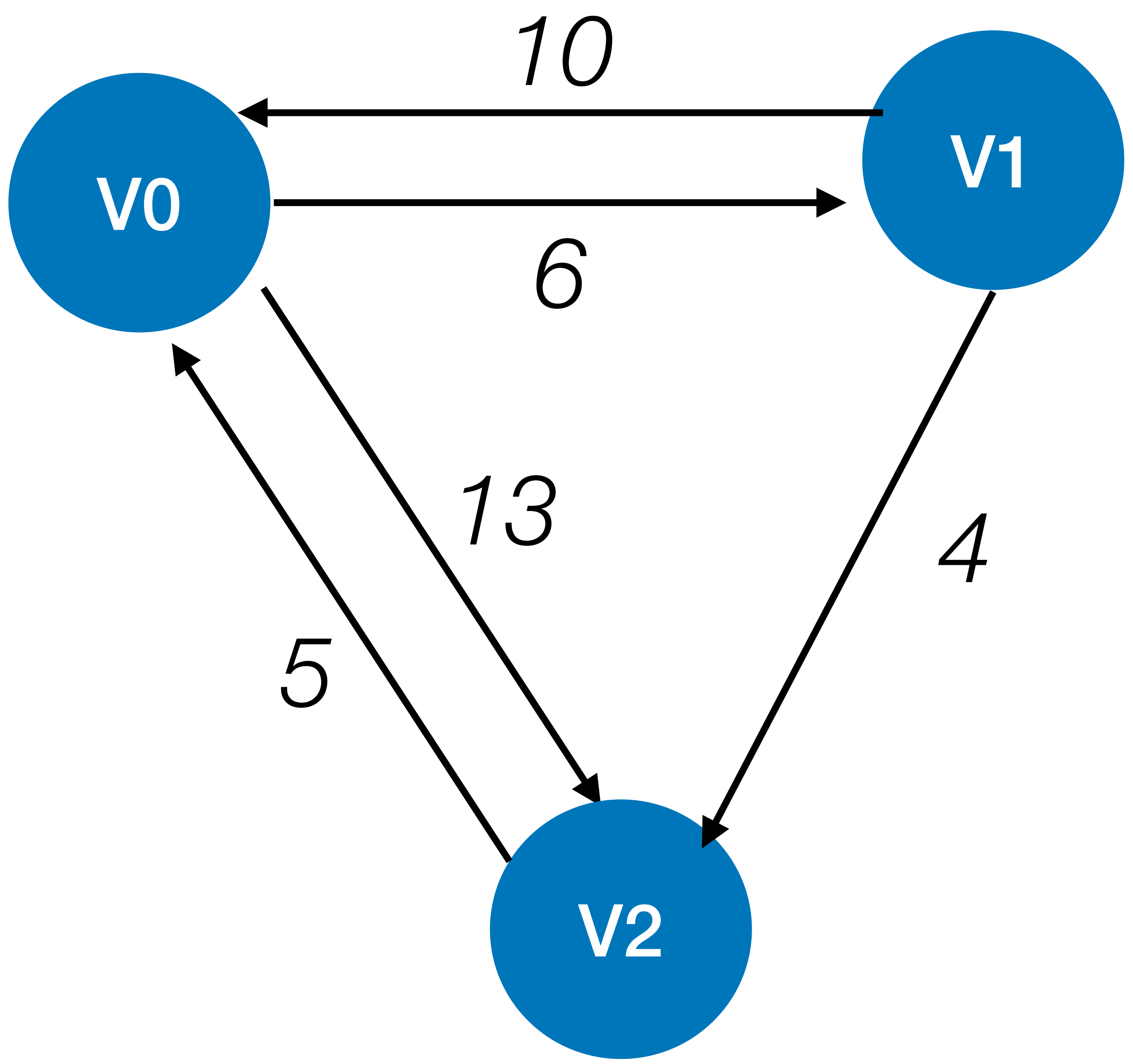
两个顶点之间的中转点

$path^{(-1)}$ =

	V0	V1	V2
V0	-1	-1	-1
V1	-1	-1	-1
V2	-1	-1	-1

#初始：不允许在其他顶点中转，最短路径是？

Floyd算法



目前来看，各顶点间的最短路径长度

$A^{(-1)}$ =

	V0	V1	V2
V0	0	6	13
V1	10	0	4
V2	5	∞	0

两个顶点之间的中转点

$path^{(-1)}$ =

	V0	V1	V2
V0	-1	-1	-1
V1	-1	-1	-1
V2	-1	-1	-1

#0：若允许在 V_0 中转，最短路径是？——求 $A^{(0)}$ 和 $path^{(0)}$

若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$

则 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$;

$path^{(k)}[i][j] = k$

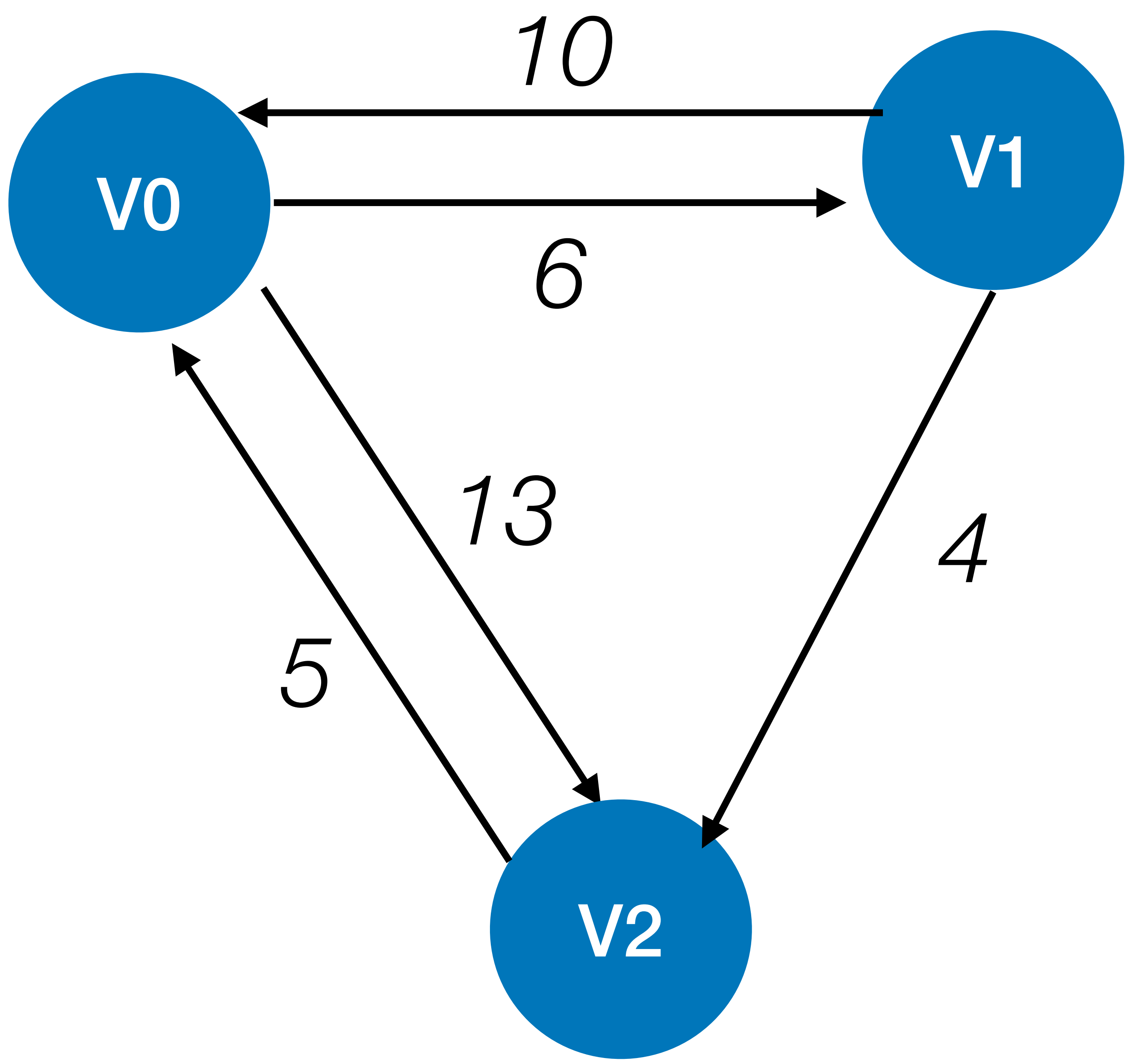
否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

$A^{(-1)}[2][1] > A^{(-1)}[2][0] + A^{(-1)}[0][1] = 11$

$A^{(0)}[2][1] = 11$

$path^{(0)}[2][1] = 0$;

Floyd算法



目前来看，各顶点间的最短路径长度

$A^{(-1)}$ =

	V0	V1	V2
V0	0	6	13
V1	10	0	4
V2	5	∞	0

两个顶点之间的中转点

$path^{(-1)}$ =

	V0	V1	V2
V0	-1	-1	-1
V1	-1	-1	-1
V2	-1	-1	-1

#0：若允许在 V_0 中转，最短路径是？——求 $A^{(0)}$ 和 $path^{(0)}$

$A^{(0)}$ =

	V0	V1	V2
V0	0	6	13
V1	10	0	4
V2	5	11	0

$path^{(0)}$ =

	V0	V1	V2
V0	-1	-1	-1
V1	-1	-1	-1
V2	-1	0	-1

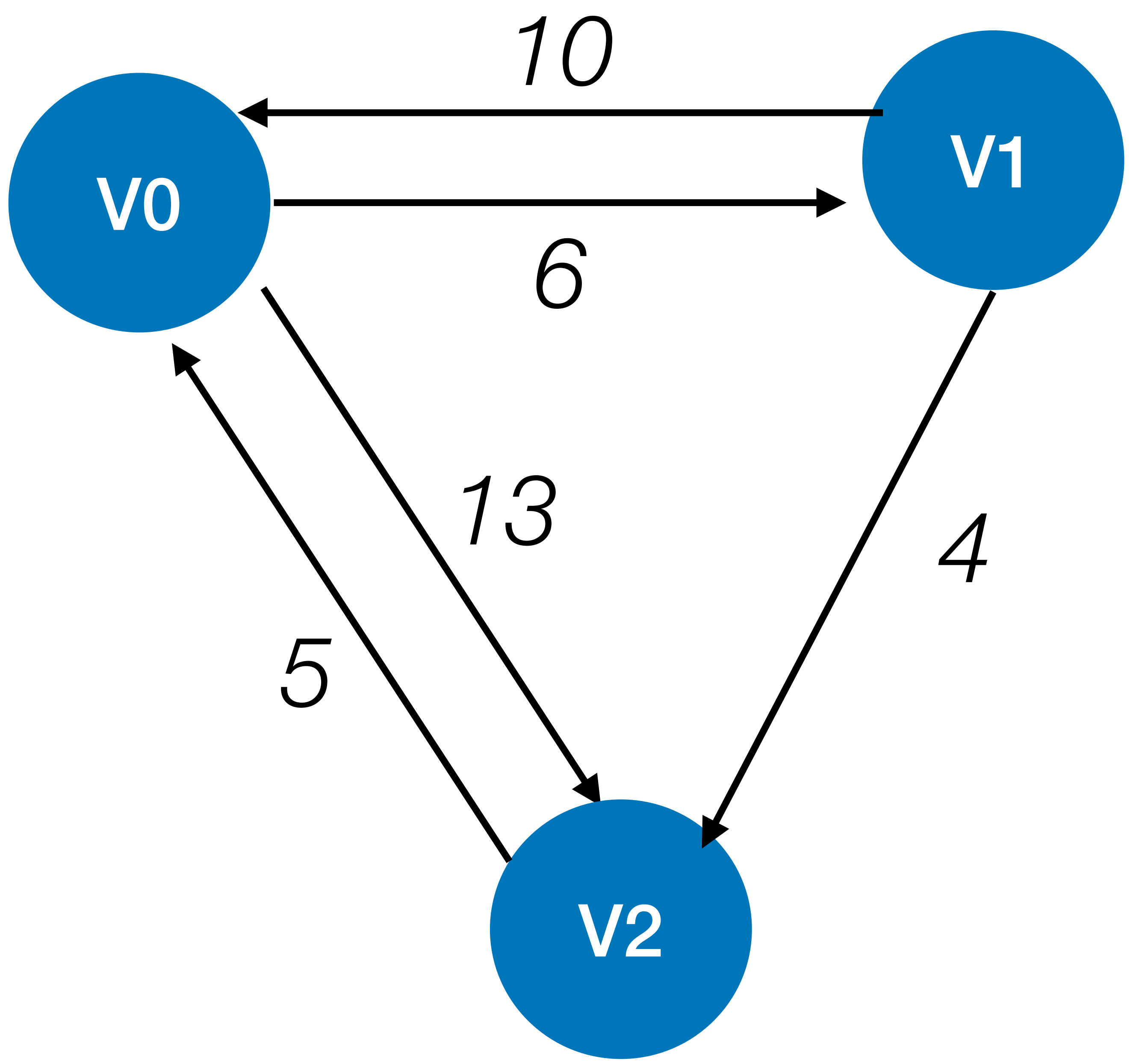
若　$A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$

则　$A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$；

　　$path^{(k)}[i][j] = k$

否则　$A^{(k)}$ 和 $path^{(k)}$ 保持原值

Floyd算法



目前来看，各顶点间的最短路径长度

$A^{(0)}$ =

	V0	V1	V2
V0	0	6	13
V1	10	0	4
V2	5	11	0

两个顶点之间的中转点

$path^{(0)}$ =

	V0	V1	V2
V0	-1	-1	-1
V1	-1	-1	-1
V2	-1	0	-1

#1：若允许在 V_0、V_1中转，最短路径是？——求 $A^{(1)}$ 和 $path^{(1)}$

若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$

则 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$;

$path^{(k)}[i][j] = k$

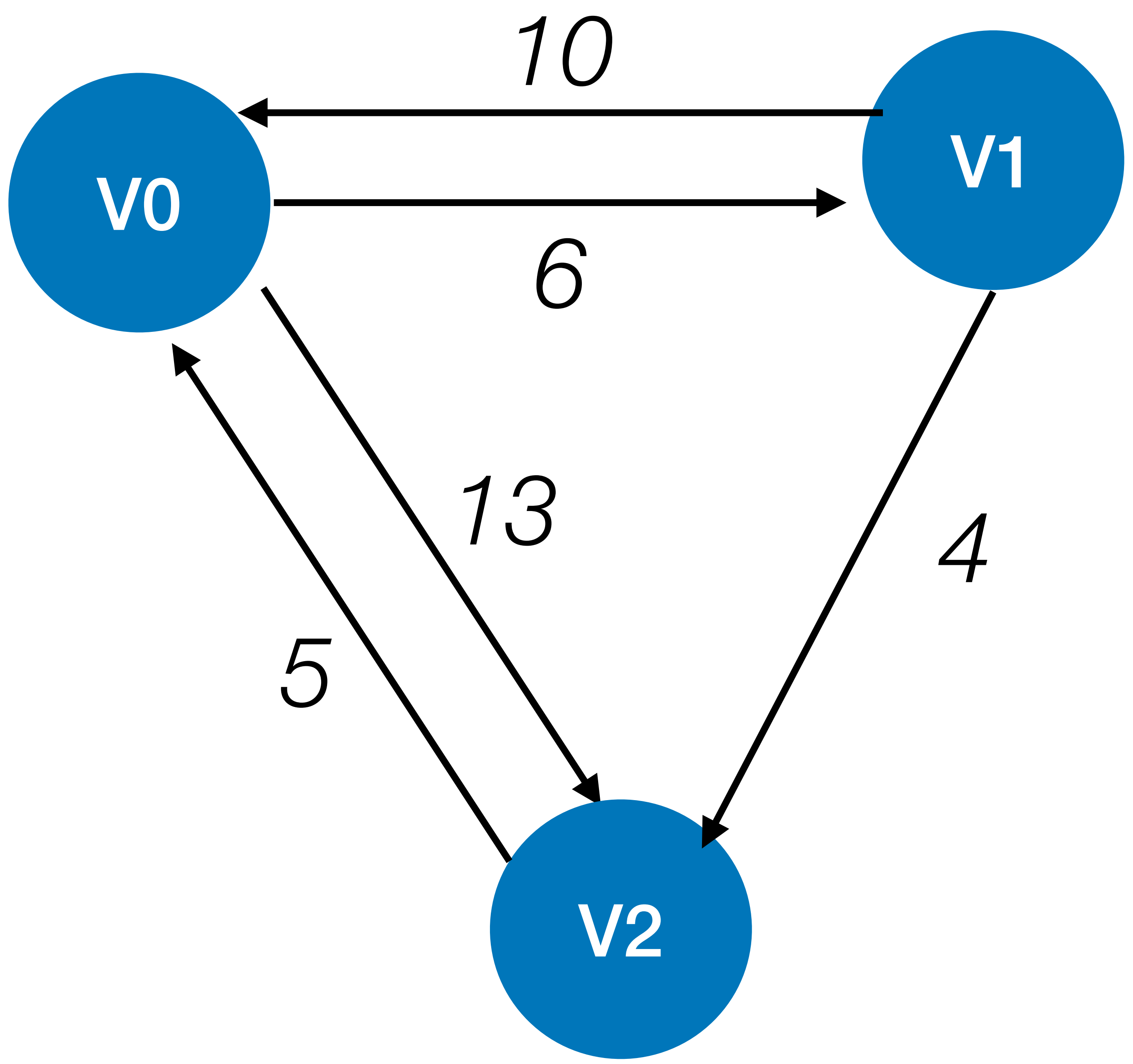
否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

$A^{(0)}[0][2] > A^{(0)}[0][1] + A^{(0)}[1][2] = 10$

$A^{(1)}[0][2] = 10$

$path^{(1)}[0][2] = 1$;

Floyd算法



目前来看，各顶点间的最短路径长度

$A^{(0)}$ =

	V0	V1	V2
V0	0	6	13
V1	10	0	4
V2	5	11	0

两个顶点之间的中转点

$path^{(0)}$ =

	V0	V1	V2
V0	-1	-1	-1
V1	-1	-1	-1
V2	-1	0	-1

#1：若允许在 V_0、V_1中转，最短路径是？——求 $A^{(1)}$ 和 $path^{(1)}$

$A^{(1)}$ =

	V0	V1	V2
V0	0	6	10
V1	10	0	4
V2	5	11	0

$path^{(1)}$ =

	V0	V1	V2
V0	-1	-1	1
V1	-1	-1	-1
V2	-1	0	-1

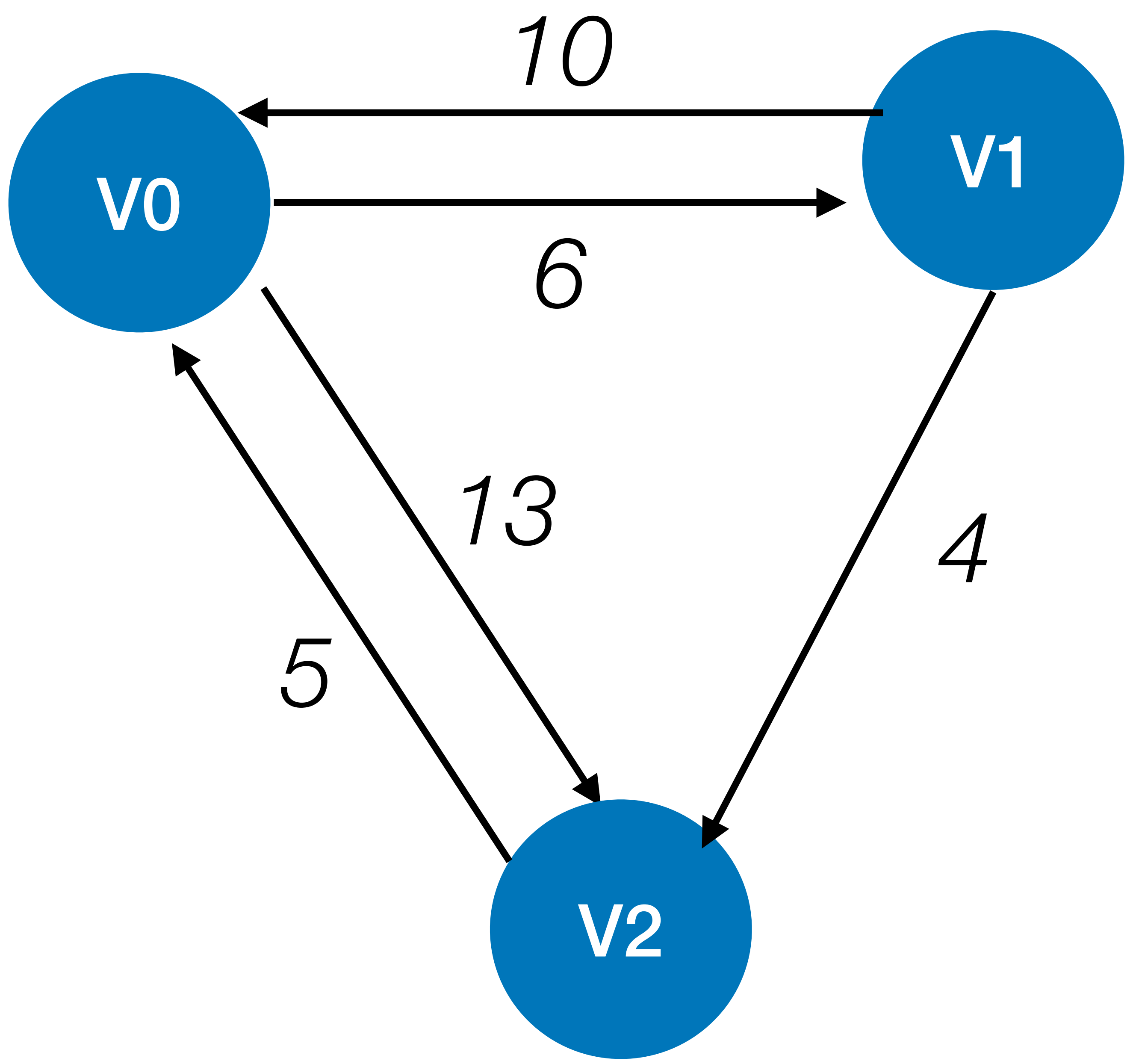
若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$

则 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$;

$path^{(k)}[i][j] = k$

否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

Floyd算法



目前来看，各顶点间的最短路径长度

$A^{(1)}$ =

	V0	V1	V2
V0	0	6	10
V1	10	0	4
V2	5	11	0

两个顶点之间的中转点

$path^{(1)}$ =

	V0	V1	V2
V0	-1	-1	1
V1	-1	-1	-1
V2	-1	0	-1

#2：若允许在 V_0、V_1、V_2 中转，最短路径是？——求 $A^{(2)}$ 和 $path^{(2)}$

若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$

则 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$;

$path^{(k)}[i][j] = k$

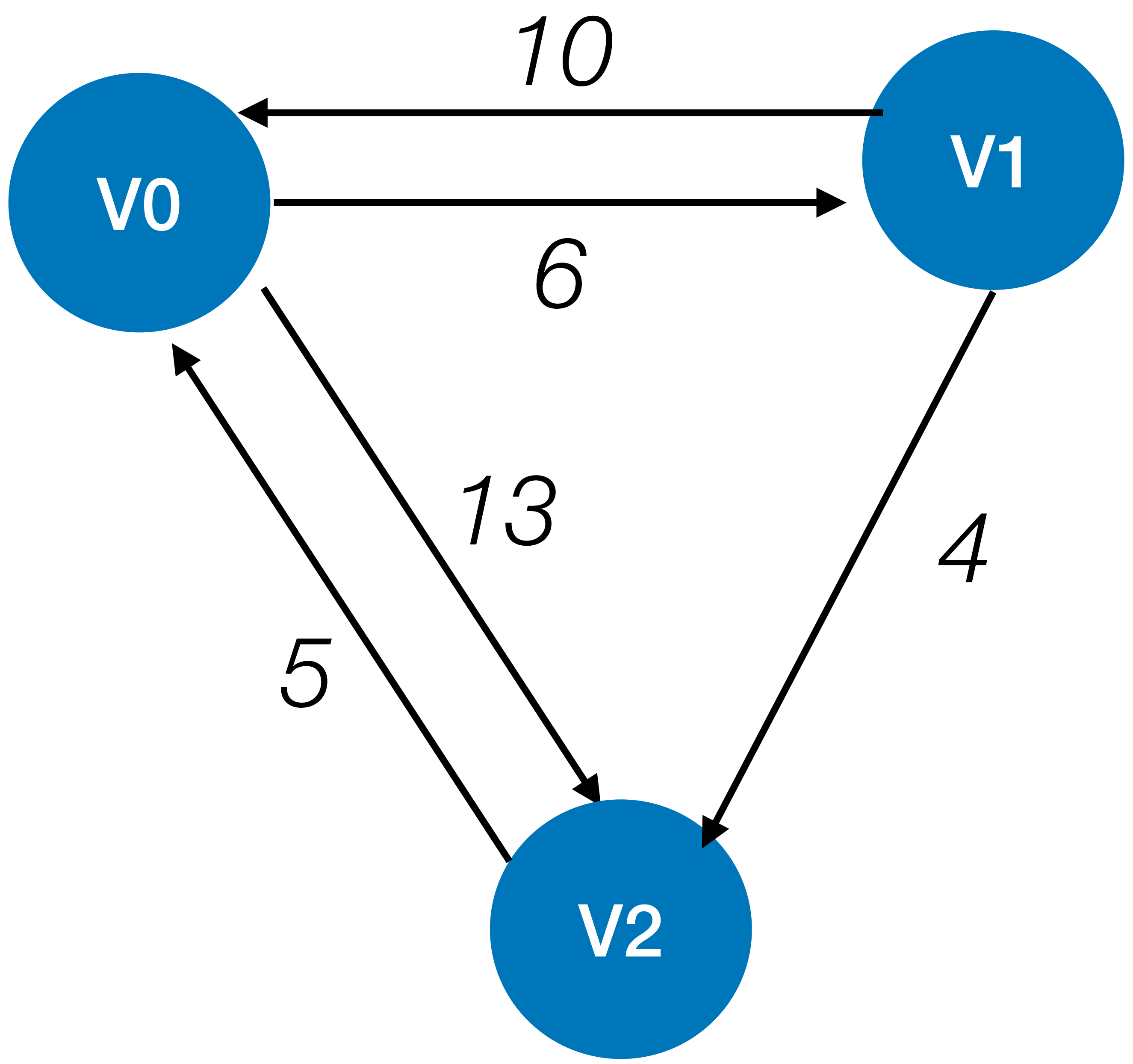
否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

$A^{(1)}[1][0] > A^{(1)}[1][2] + A^{(1)}[2][0] = 9$

$A^{(2)}[1][0] = 9$

$path^{(2)}[1][0] = 2;$

Floyd算法



目前来看，各顶点间的最短路径长度

$A^{(1)}$ =

	V0	V1	V2
V0	0	6	10
V1	10	0	4
V2	5	11	0

两个顶点之间的中转点

$path^{(1)}$ =

	V0	V1	V2
V0	-1	-1	1
V1	-1	-1	-1
V2	-1	0	-1

#2：若允许在 V_0、V_1、V_2 中转，最短路径是？——求 $A^{(2)}$ 和 $path^{(2)}$

$A^{(2)}$ =

	V0	V1	V2
V0	0	6	10
V1	9	0	4
V2	5	11	0

$path^{(2)}$ =

	V0	V1	V2
V0	-1	-1	1
V1	2	-1	-1
V2	-1	0	-1

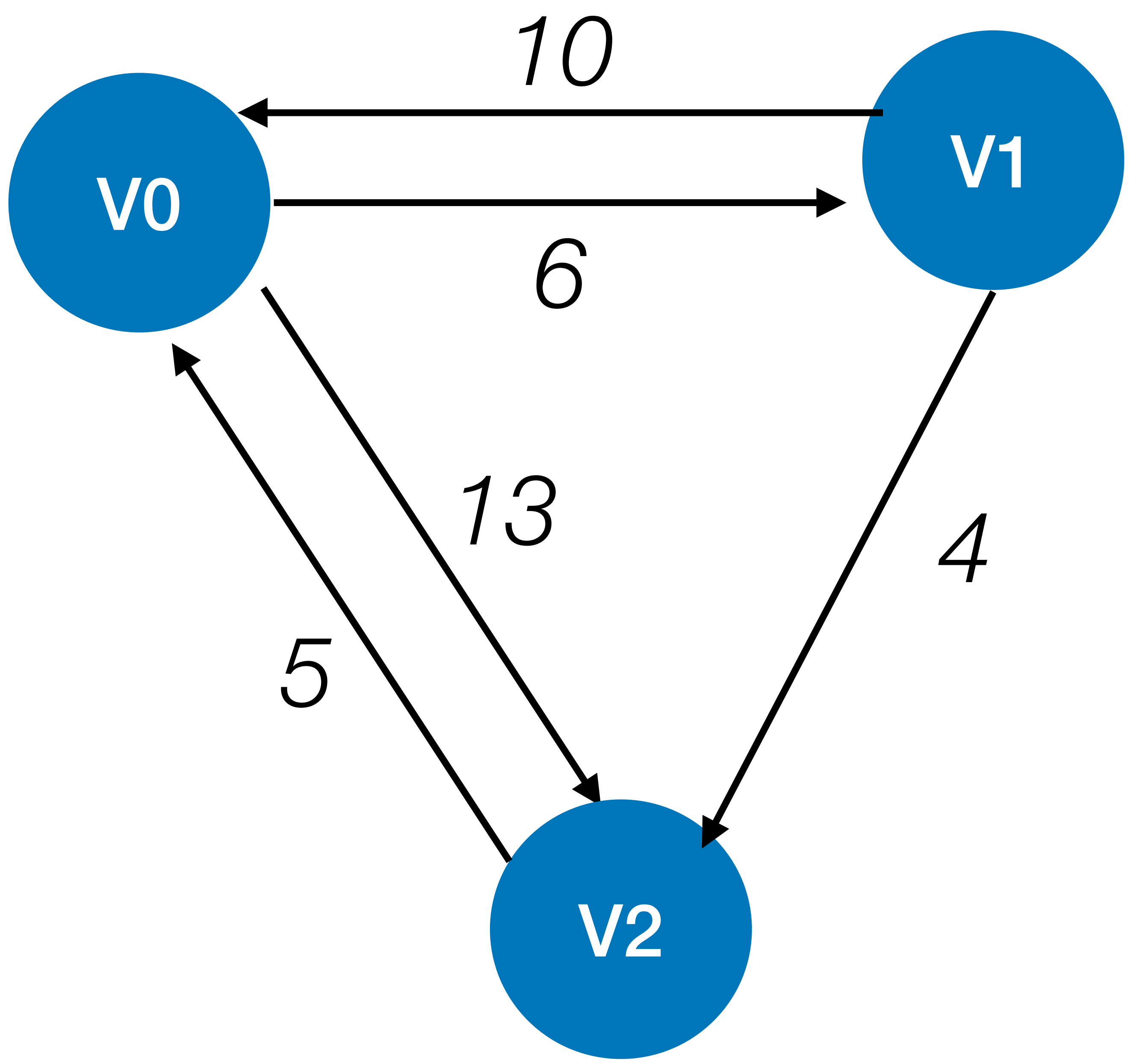
若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$

则 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$；

$path^{(k)}[i][j] = k$

否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

Floyd算法



目前来看，各顶点间的最短路径长度

$A^{(2)}$ =

	V0	V1	V2
V0	0	6	10
V1	9	0	4
V2	5	11	0

两个顶点之间的中转点

$path^{(2)}$ =

	V0	V1	V2
V0	-1	-1	1
V1	2	-1	-1
V2	-1	0	-1

从$A^{(-1)}$ 和 $path^{(-1)}$ 开始，经过 n 轮递推，得到 $A^{(n-1)}$ 和 $path^{(n-1)}$

根据 $A^{(2)}$ 可知，V1到V2 最短路径长度为 4,
根据 $path^{(2)}$ 可知，完整路径信息为 V1_V2

根据 $A^{(2)}$ 可知，V0到V2 最短路径长度为 10,
根据 $path^{(2)}$ 可知，完整路径信息为 V0_V1_V2

根据 $A^{(2)}$ 可知，V1到V0 最短路径长度为 9,
根据 $path^{(2)}$ 可知，完整路径信息为 V1_V2_V0

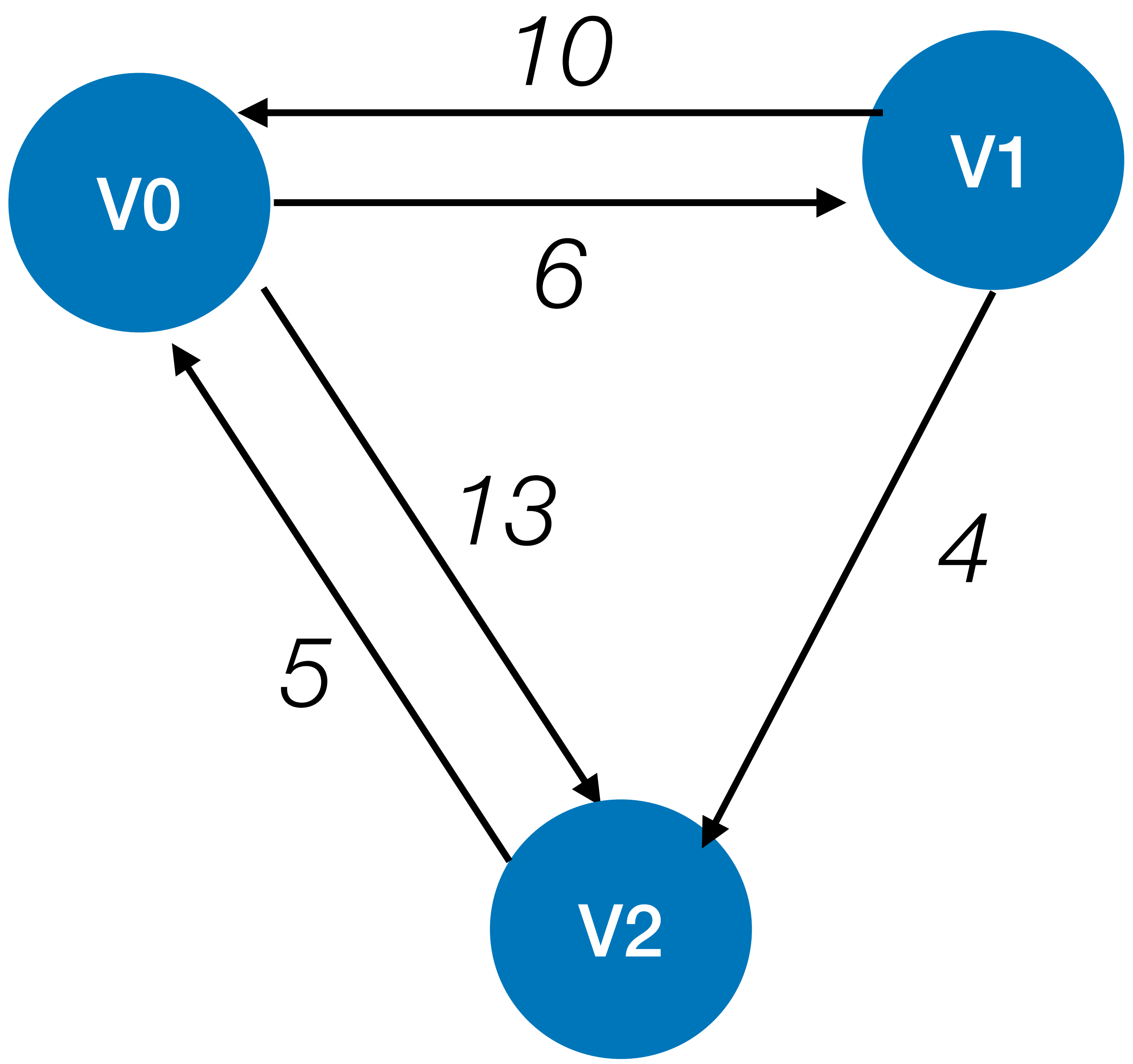
若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$

则 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$;

$path^{(k)}[i][j] = k$

否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

Floyd算法核心代码



A=

	V0	V1	V2
V0	0	6	13
V1	10	0	4
V2	5	∞	0

path=

	V0	V1	V2
V0	-1	-1	-1
V1	-1	-1	-1
V2	-1	-1	-1

```
//......准备工作，根据图的信息初始化矩阵 A 和 path（如上图）
for (int k=0; k<n; k++){      //考虑以 vk 作为中转点
  for(int i=0; i<n; i++) {    //遍历整个矩阵，i为行号，j为列号
    for (int j=0; j<n; j++){
      if (A[i][j]>A[i][k]+A[k][j]){    //以 vk 为中转点的路径更短
        A[i][j]=A[i][k]+A[k][j];       //更新最短路径长度
        path[i][j]=k;                  //中转点
      }
    }
  }
}
```

时间复杂度，$O(|V|^3)$
空间复杂度，$O(|V|^2)$

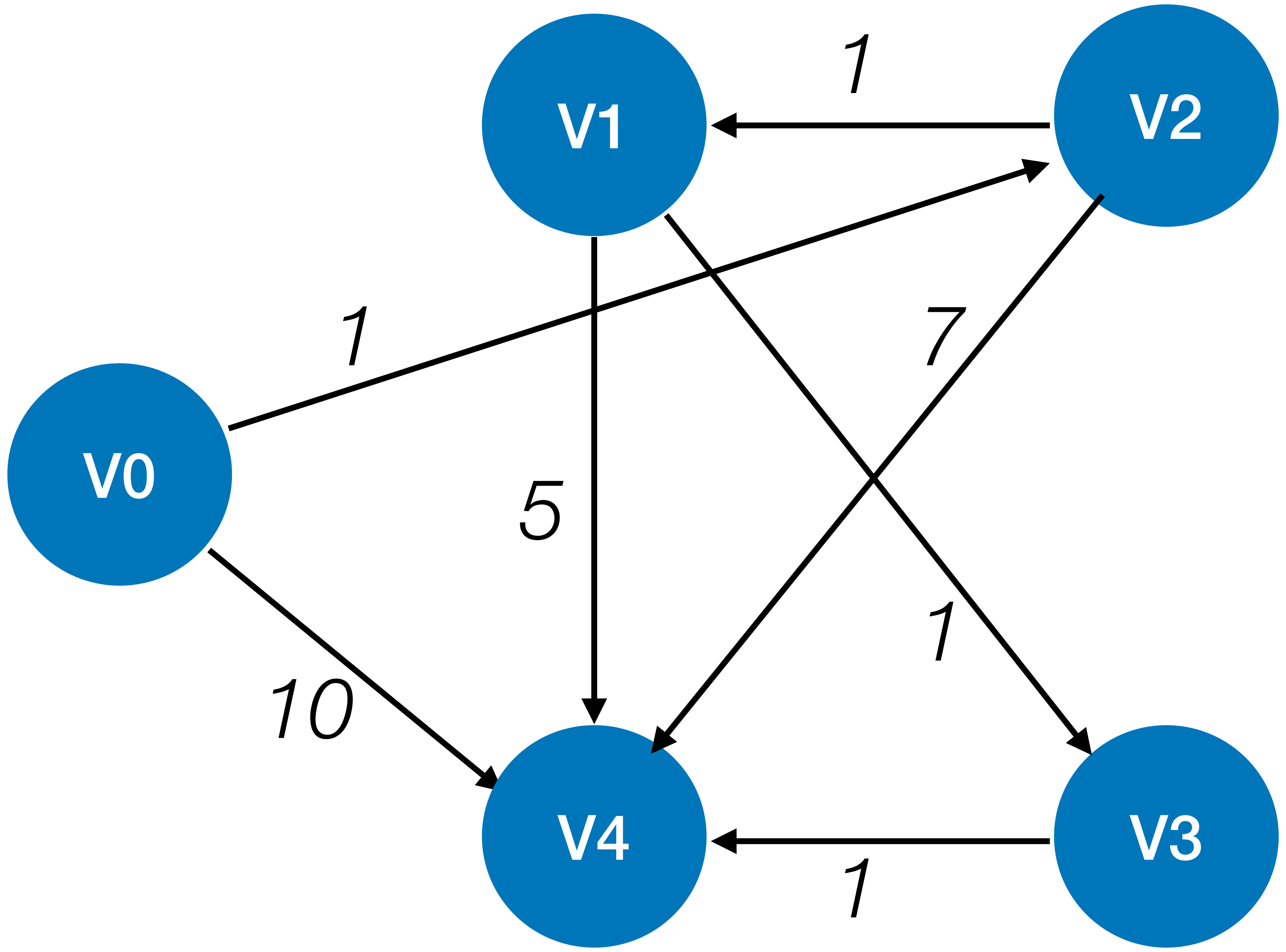
若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$

则 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$;

$path^{(k)}[i][j] = k$

否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

Floyd算法实例



$A^{(-1)} =$

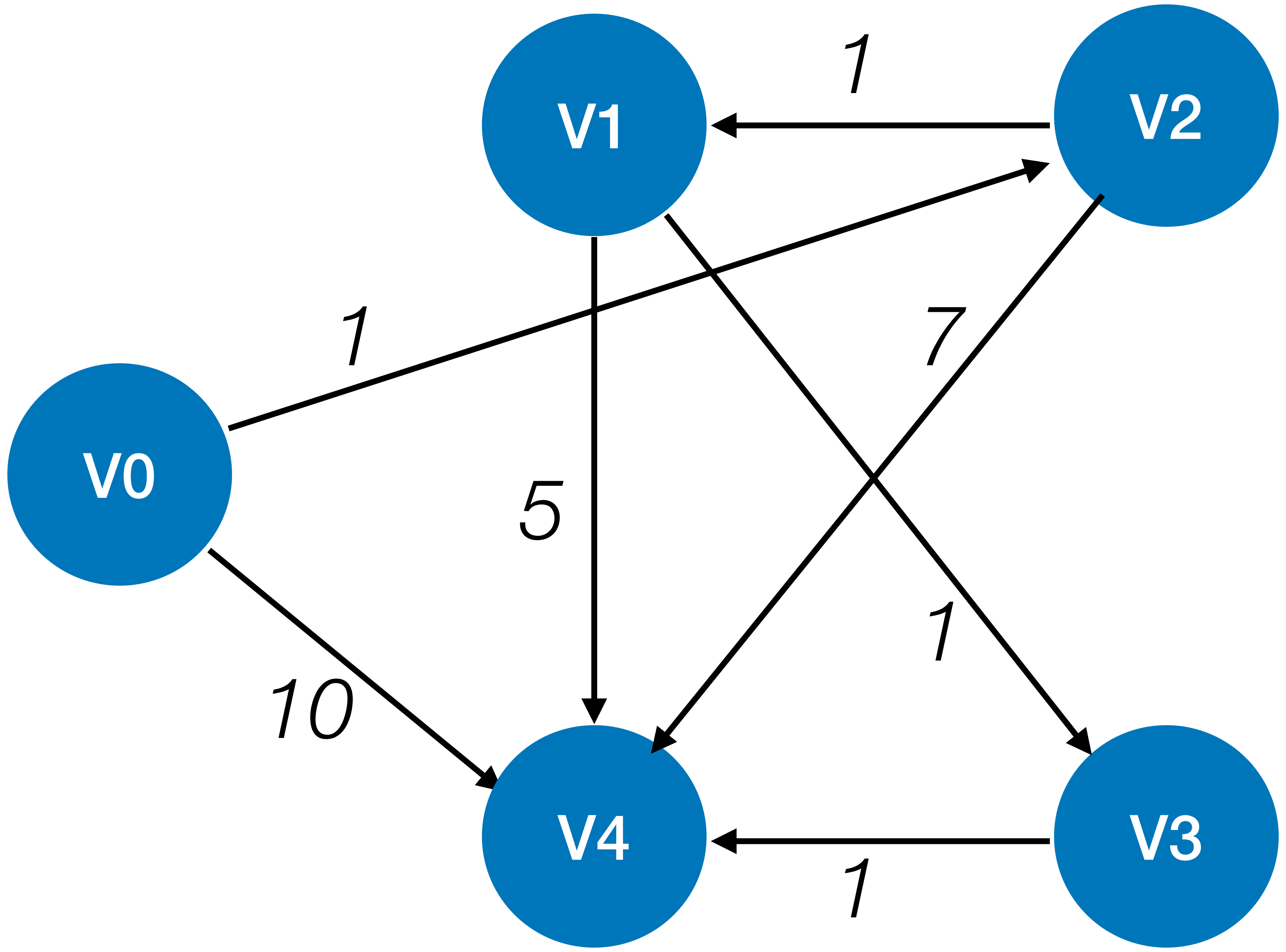
	V0	V1	V2	V3	V4
V0	0	∞	1	∞	10
V1	∞	0	∞	1	5
V2	∞	1	0	∞	7
V3	∞	∞	∞	0	1
V4	∞	∞	∞	∞	0

$path^{(-1)} =$

	V0	V1	V2	V3	V4
V0	-1	-1	-1	-1	-1
V1	-1	-1	-1	-1	-1
V2	-1	-1	-1	-1	-1
V3	-1	-1	-1	-1	-1
V4	-1	-1	-1	-1	-1

#初始：不允许在其他顶点中转，最短路径是?

Floyd算法实例



$A^{(-1)}$ =

	V0	V1	V2	V3	V4
V0	0	∞	1	∞	10
V1	∞	0	∞	1	5
V2	∞	1	0	∞	7
V3	∞	∞	∞	0	1
V4	∞	∞	∞	∞	0

$path^{(-1)}$ =

	V0	V1	V2	V3	V4
V0	-1	-1	-1	-1	-1
V1	-1	-1	-1	-1	-1
V2	-1	-1	-1	-1	-1
V3	-1	-1	-1	-1	-1
V4	-1	-1	-1	-1	-1

#0：若允许在 V_0 中转，最短路径是？——求 $A^{(0)}$ 和 $path^{(0)}$

若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$

则 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$;

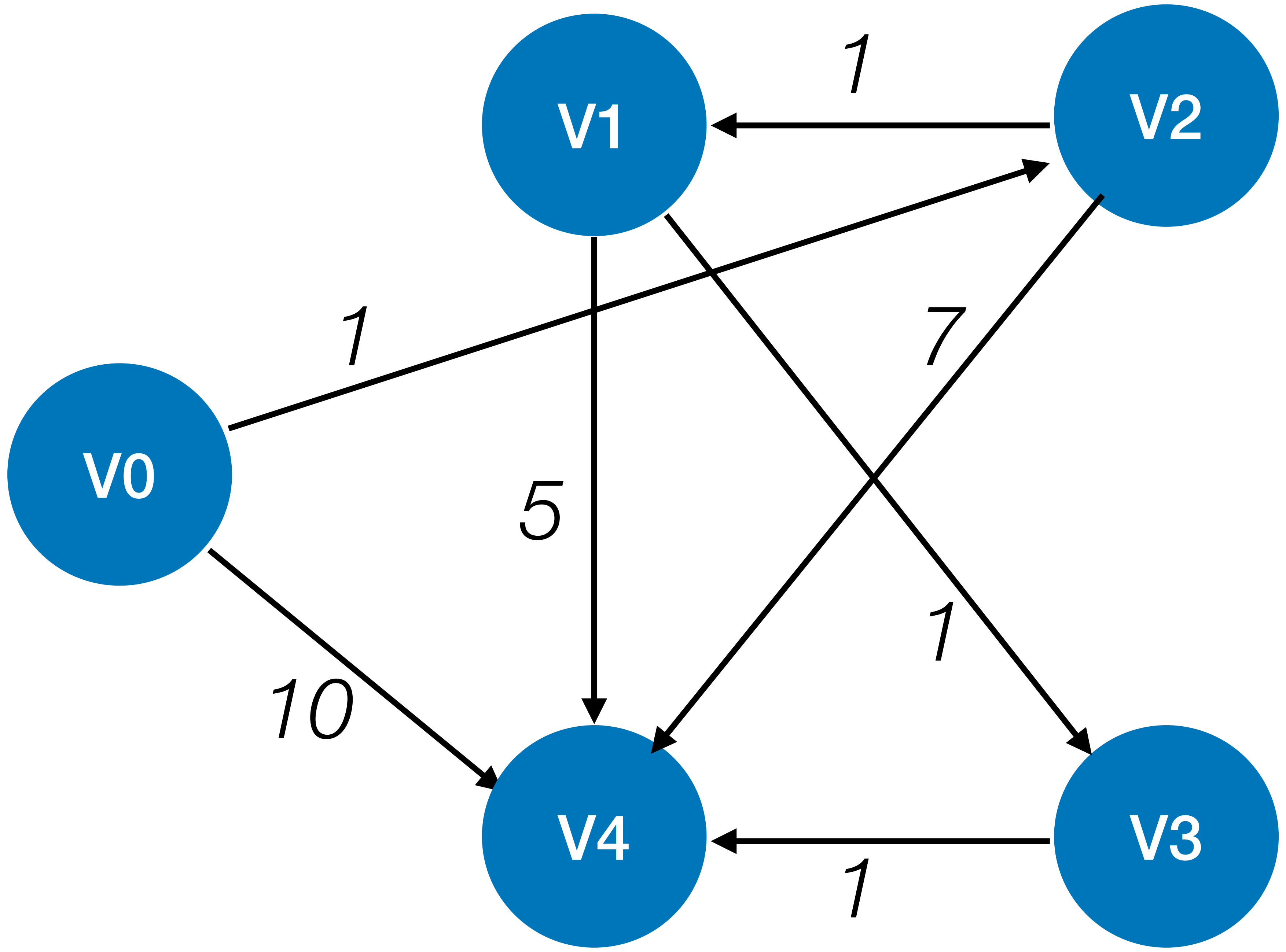
$path^{(k)}[i][j] = k$

否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

```
for(int i=0; i<n; i++) {  //遍历整个矩阵，i为行号，j为列号
  for (int j=0; j<n; j++){
    if (A[i][j]>A[i][k]+A[k][j]){    //以 Vk 为中转点的路径更短
      A[i][j]=A[i][k]+A[k][j];       //更新最短路径长度
      path[i][j]=k;                  //中转点
    }
  }
}
```

k=0

Floyd算法实例



$A^{(-1)}$ =

	V0	V1	V2	V3	V4
V0	0	∞	1	∞	10
V1	∞	0	∞	1	5
V2	∞	1	0	∞	7
V3	∞	∞	∞	0	1
V4	∞	∞	∞	∞	0

$path^{(-1)}$ =

	V0	V1	V2	V3	V4
V0	-1	-1	-1	-1	-1
V1	-1	-1	-1	-1	-1
V2	-1	-1	-1	-1	-1
V3	-1	-1	-1	-1	-1
V4	-1	-1	-1	-1	-1

#0：若允许在 V_0 中转，最短路径是？——求 $A^{(0)}$ 和 $path^{(0)}$

$A^{(0)}$ =

	V0	V1	V2	V3	V4
V0	0	∞	1	∞	10
V1	∞	0	∞	1	5
V2	∞	1	0	∞	7
V3	∞	∞	∞	0	1
V4	∞	∞	∞	∞	0

$path^{(0)}$ =

	V0	V1	V2	V3	V4
V0	-1	-1	-1	-1	-1
V1	-1	-1	-1	-1	-1
V2	-1	-1	-1	-1	-1
V3	-1	-1	-1	-1	-1
V4	-1	-1	-1	-1	-1

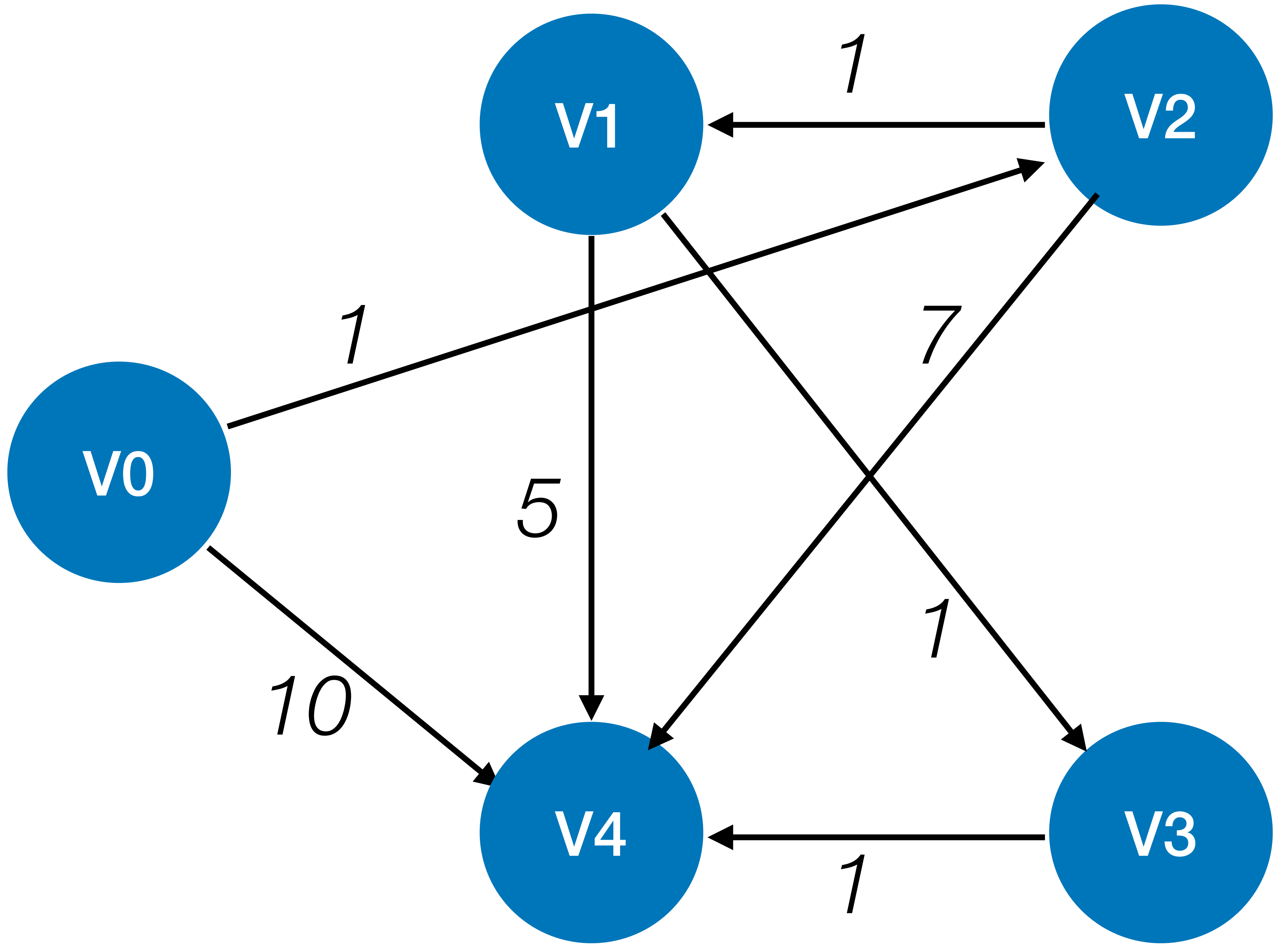
若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$

则 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$;

$path^{(k)}[i][j] = k$

否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

Floyd算法实例



$A^{(0)} =$

	V0	V1	V2	V3	V4
V0	0	∞	1	∞	10
V1	∞	0	∞	1	5
V2	∞	1	0	∞	7
V3	∞	∞	∞	0	1
V4	∞	∞	∞	∞	0

$path^{(0)} =$

	V0	V1	V2	V3	V4
V0	-1	-1	-1	-1	-1
V1	-1	-1	-1	-1	-1
V2	-1	-1	-1	-1	-1
V3	-1	-1	-1	-1	-1
V4	-1	-1	-1	-1	-1

#1：若允许在 V_0、V_1中转，最短路径是？——求 $A^{(1)}$ 和 $path^{(1)}$

```
for(int i=0; i<n; i++) {  //遍历整个矩阵，i为行号，j为列号
  for (int j=0; j<n; j++){
    if (A[i][j]>A[i][k]+A[k][j]){    //以 Vk 为中转点的路径更短
      A[i][j]=A[i][k]+A[k][j];       //更新最短路径长度
      path[i][j]=k;                  //中转点
    }
  }
}
```

k=1

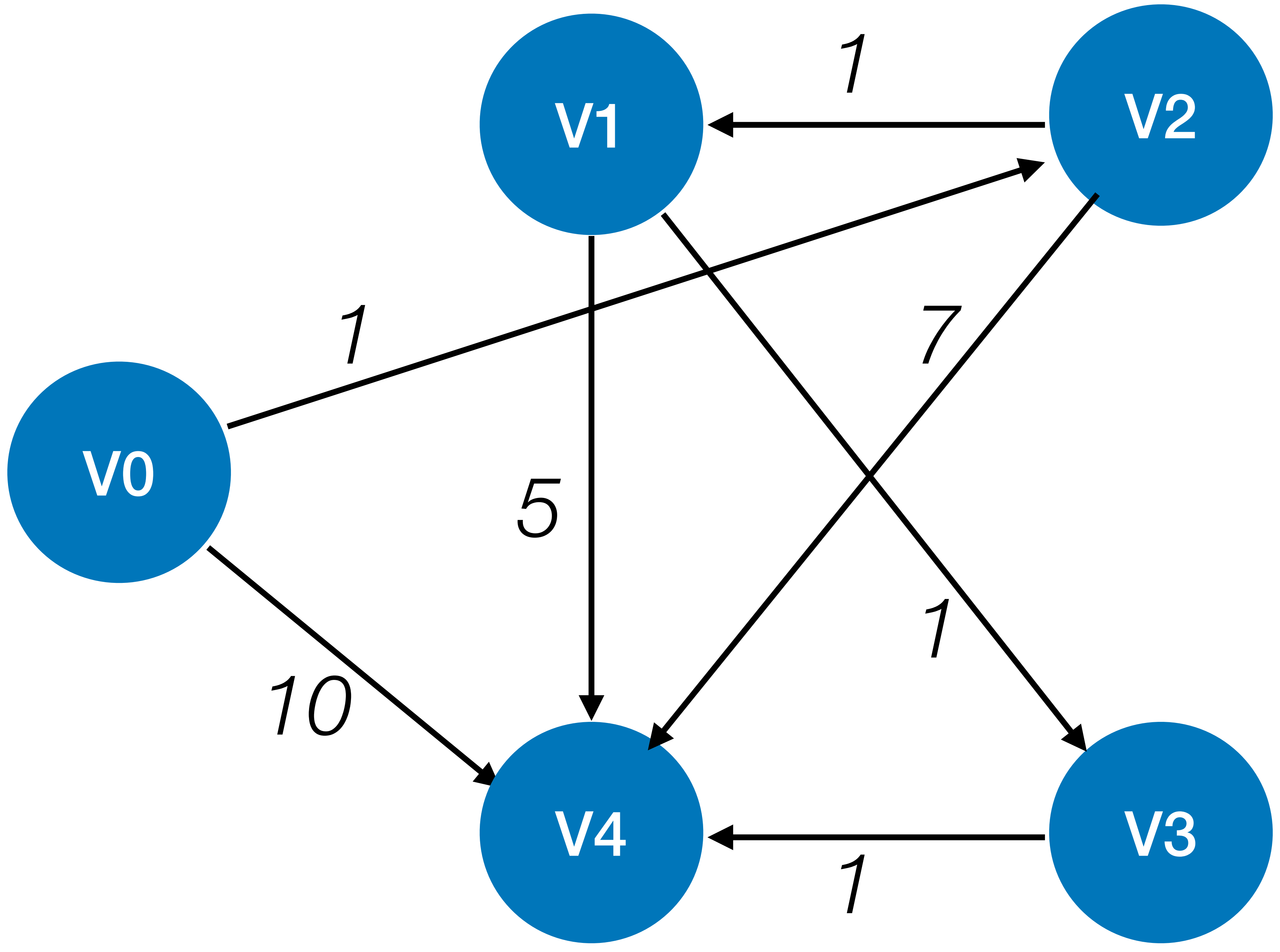
若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$

则 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$；

$path^{(k)}[i][j] = k$

否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

Floyd算法实例



$A^{(0)} =$

	V0	V1	V2	V3	V4
V0	0	∞	1	∞	10
V1	∞	0	∞	1	5
V2	∞	1	0	∞	7
V3	∞	∞	∞	0	1
V4	∞	∞	∞	∞	0

$path^{(0)} =$

	V0	V1	V2	V3	V4
V0	-1	-1	-1	-1	-1
V1	-1	-1	-1	-1	-1
V2	-1	-1	-1	-1	-1
V3	-1	-1	-1	-1	-1
V4	-1	-1	-1	-1	-1

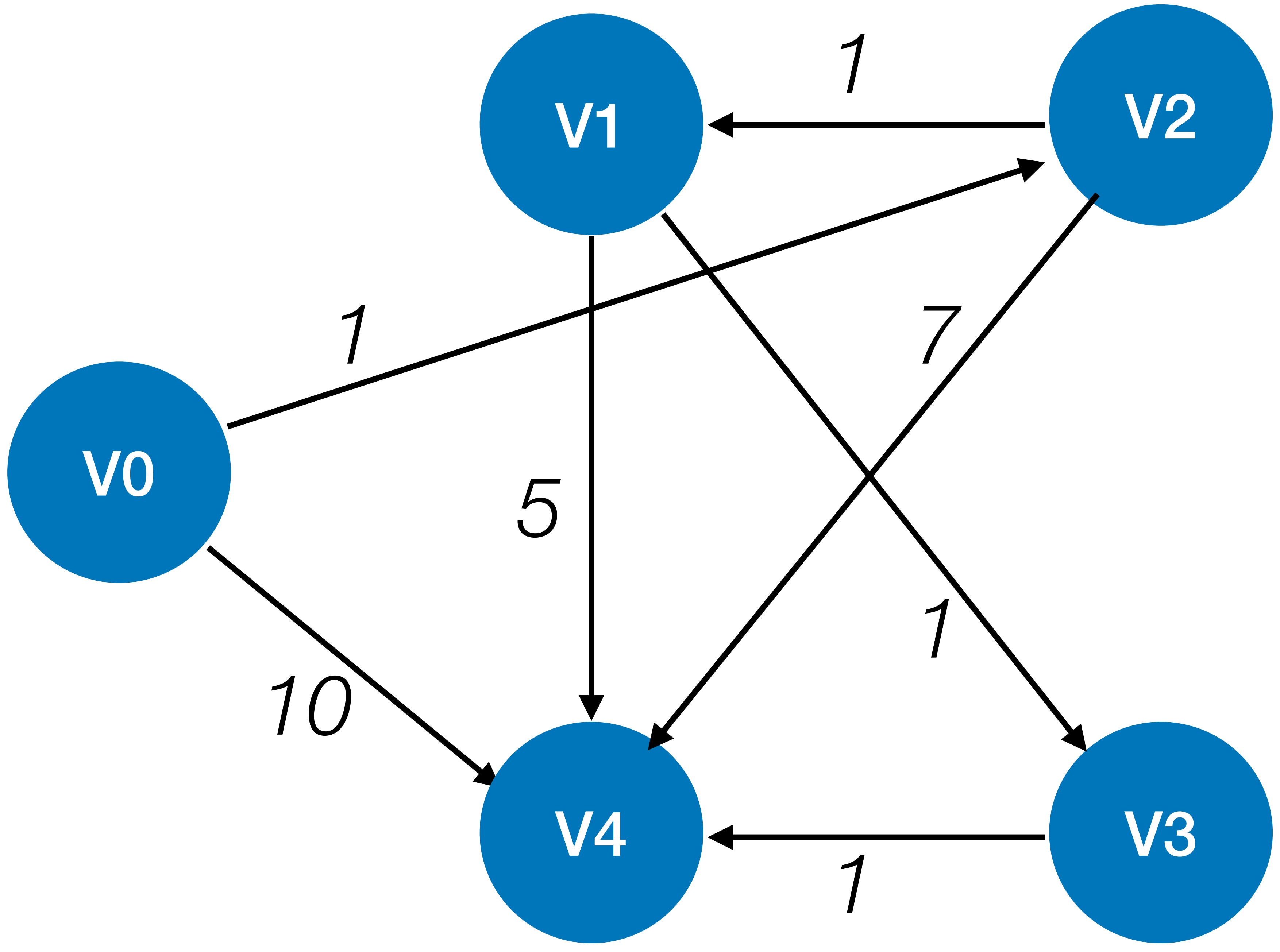
#1：若允许在 V_0、V_1中转，最短路径是？——求 $A^{(1)}$ 和 $path^{(1)}$

$A^{(0)}[2][3] > A^{(0)}[2][1] + A^{(0)}[1][3] = 2$
$A^{(1)}[2][3] = 2$
$path^{(1)}[2][3] = 1;$

$A^{(0)}[2][4] > A^{(0)}[2][1] + A^{(0)}[1][4] = 6$
$A^{(1)}[2][4] = 6$
$path^{(1)}[2][4] = 1;$

若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$
则 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j];$
$path^{(k)}[i][j] = k$
否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

Floyd算法实例



若　$A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$

则　$A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$;

　　$path^{(k)}[i][j] = k$

否则　$A^{(k)}$ 和 $path^{(k)}$ 保持原值

$A^{(0)}$ =

	V0	V1	V2	V3	V4
V0	0	∞	1	∞	10
V1	∞	0	∞	1	5
V2	∞	1	0	∞	7
V3	∞	∞	∞	0	1
V4	∞	∞	∞	∞	0

$path^{(0)}$ =

	V0	V1	V2	V3	V4
V0	-1	-1	-1	-1	-1
V1	-1	-1	-1	-1	-1
V2	-1	-1	-1	-1	-1
V3	-1	-1	-1	-1	-1
V4	-1	-1	-1	-1	-1

#1：若允许在 V_0、V_1中转，最短路径是？——求 $A^{(1)}$ 和 $path^{(1)}$

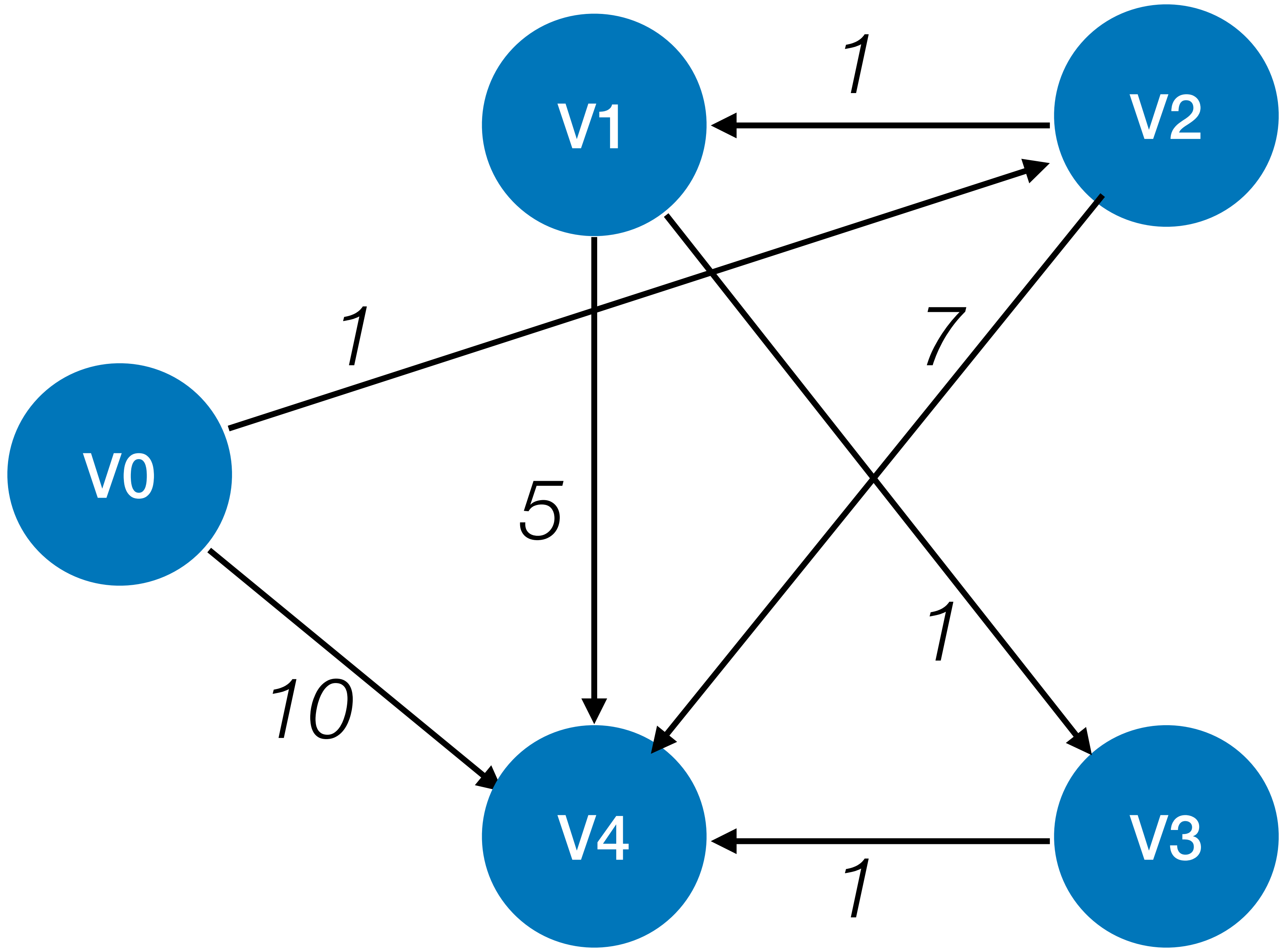
$A^{(1)}$ =

	V0	V1	V2	V3	V4
V0	0	∞	1	∞	10
V1	∞	0	∞	1	5
V2	∞	1	0	2	6
V3	∞	∞	∞	0	1
V4	∞	∞	∞	∞	0

$path^{(1)}$ =

	V0	V1	V2	V3	V4
V0	-1	-1	-1	-1	-1
V1	-1	-1	-1	-1	-1
V2	-1	-1	-1	1	1
V3	-1	-1	-1	-1	-1
V4	-1	-1	-1	-1	-1

Floyd算法实例



$A^{(1)} =$

	V0	V1	V2	V3	V4
V0	0	∞	1	∞	10
V1	∞	0	∞	1	5
V2	∞	1	0	2	6
V3	∞	∞	∞	0	1
V4	∞	∞	∞	∞	0

$path^{(1)} =$

	V0	V1	V2	V3	V4
V0	-1	-1	-1	-1	-1
V1	-1	-1	-1	-1	-1
V2	-1	-1	-1	1	1
V3	-1	-1	-1	-1	-1
V4	-1	-1	-1	-1	-1

#2：若允许在 V_0、V_1、V_2中转，最短路径是？——求 $A^{(2)}$ 和 $path^{(2)}$

```
for(int i=0; i<n; i++) {  //遍历整个矩阵，i为行号，j为列号
  for (int j=0; j<n; j++){
    if (A[i][j]>A[i][k]+A[k][j]){   //以 Vk 为中转点的路径更短
      A[i][j]=A[i][k]+A[k][j];      //更新最短路径长度
      path[i][j]=k;                 //中转点
    }
  }
}
```

k=2

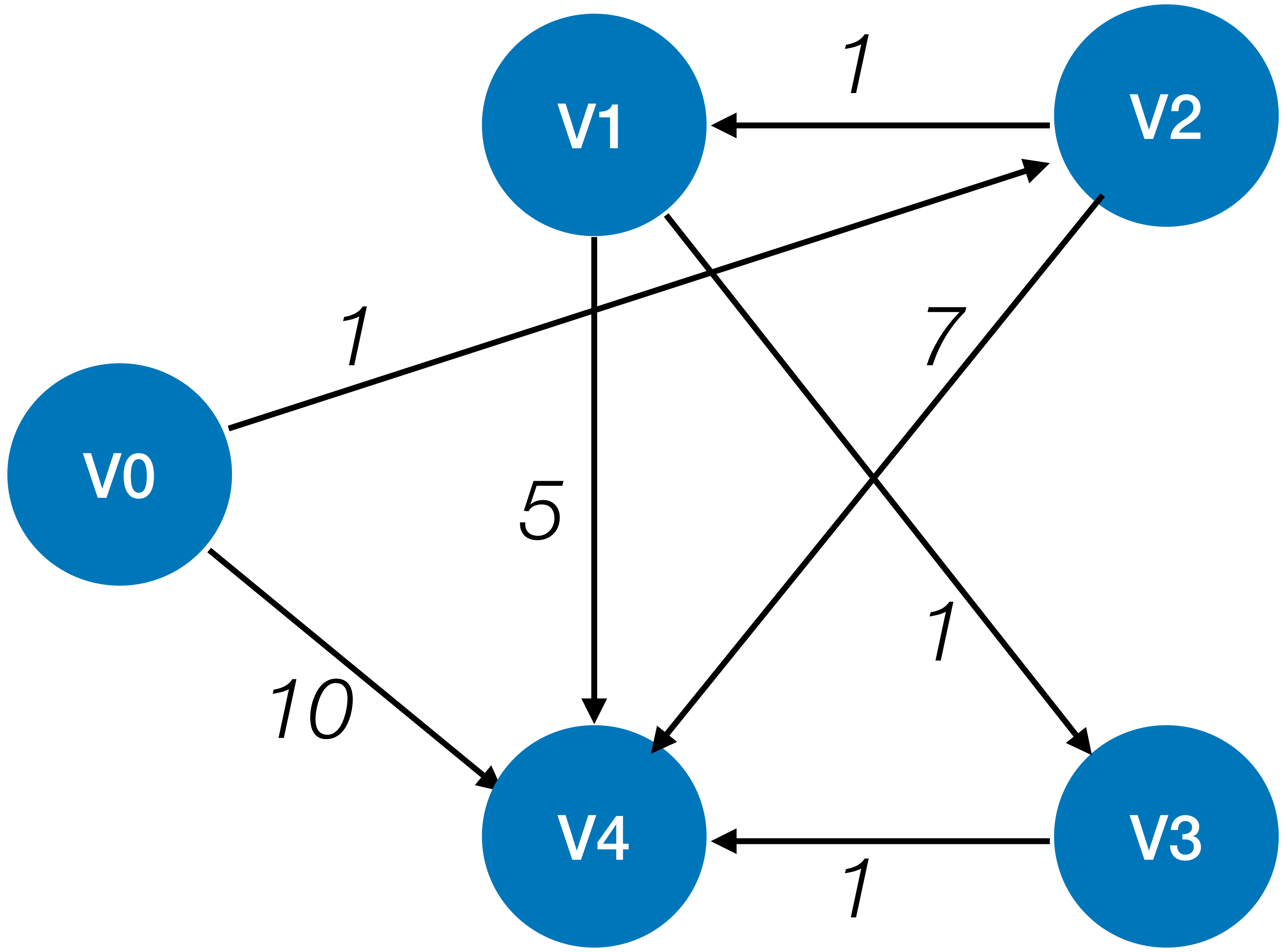
若　$A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$

则　$A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$;

　　$path^{(k)}[i][j] = k$

否则　$A^{(k)}$ 和 $path^{(k)}$ 保持原值

Floyd算法实例



若 $A^{(k-1)}[i][j]>A^{(k-1)}[i][k]+A^{(k-1)}[k][j]$

则 $A^{(k)}[i][j]=A^{(k-1)}[i][k]+A^{(k-1)}[k][j]$;

$path^{(k)}[i][j]=k$

否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

$A^{(1)}$ =

	V0	V1	V2	V3	V4
V0	0	∞	1	∞	10
V1	∞	0	∞	1	5
V2	∞	1	0	2	6
V3	∞	∞	∞	0	1
V4	∞	∞	∞	∞	0

$path^{(1)}$ =

	V0	V1	V2	V3	V4
V0	-1	-1	-1	-1	-1
V1	-1	-1	-1	-1	-1
V2	-1	-1	-1	1	1
V3	-1	-1	-1	-1	-1
V4	-1	-1	-1	-1	-1

#2：若允许在 V_0、V_1、V_2中转，最短路径是？——求 $A^{(2)}$ 和 $path^{(2)}$

$A^{(1)}[0][1] > A^{(1)}[0][2]+A^{(1)}[2][1] = 2$

$A^{(2)}[0][1] = 2$; $path^{(2)}[0][1] = 2$;

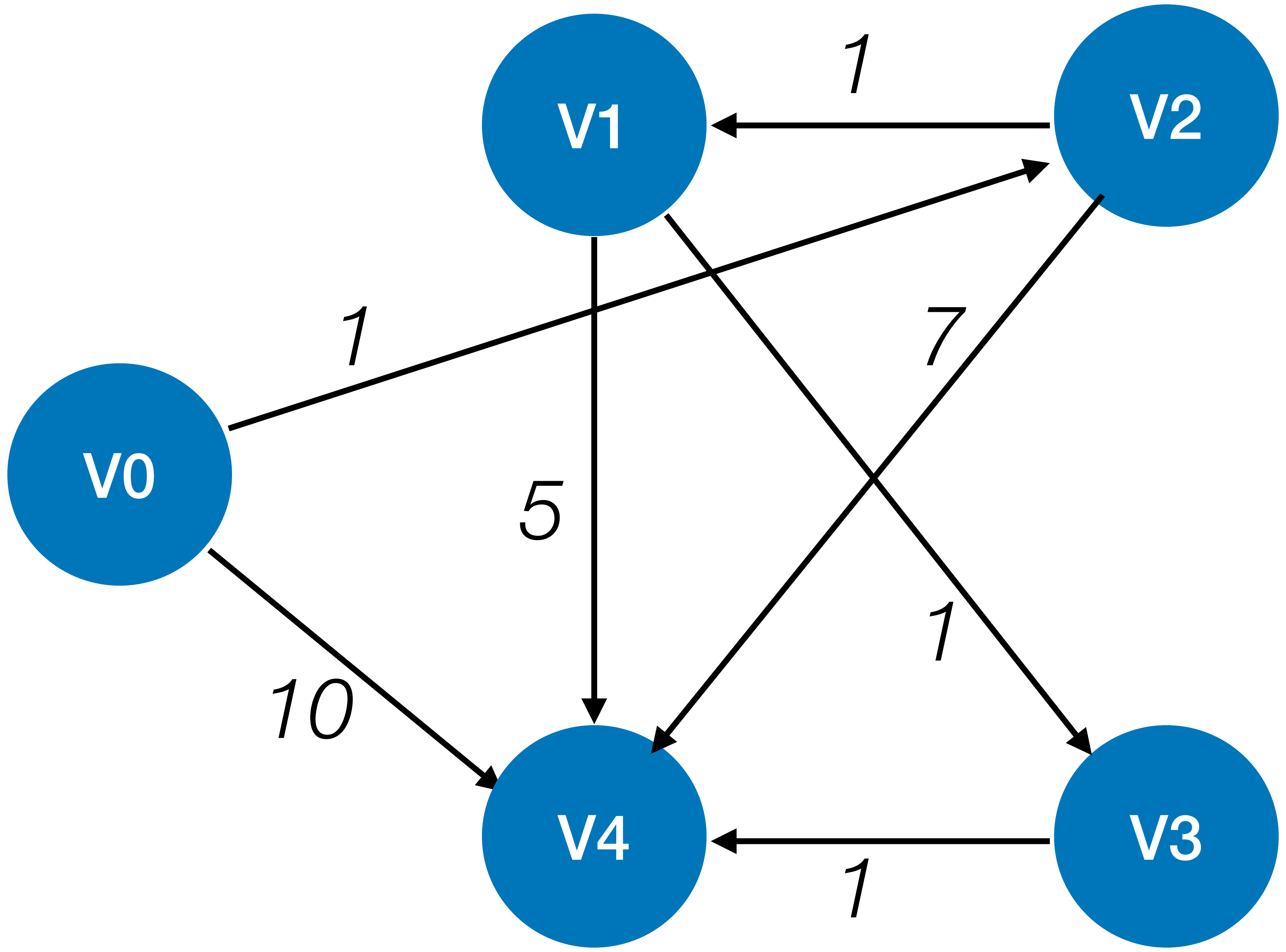
$A^{(1)}[0][3] > A^{(1)}[0][2]+A^{(1)}[2][3] = 3$

$A^{(2)}[0][3] = 3$; $path^{(2)}[0][3] = 2$;

$A^{(1)}[0][4] > A^{(1)}[0][2]+A^{(1)}[2][4] = 7$

$A^{(2)}[0][4] = 7$; $path^{(2)}[0][4] = 2$;

Floyd算法实例



若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$

则 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$;

$path^{(k)}[i][j] = k$

否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

$A^{(1)} =$

	V0	V1	V2	V3	V4
V0	0	∞	1	∞	10
V1	∞	0	∞	1	5
V2	∞	1	0	2	6
V3	∞	∞	∞	0	1
V4	∞	∞	∞	∞	0

$path^{(1)} =$

	V0	V1	V2	V3	V4
V0	-1	-1	-1	-1	-1
V1	-1	-1	-1	-1	-1
V2	-1	-1	-1	1	1
V3	-1	-1	-1	-1	-1
V4	-1	-1	-1	-1	-1

#2：若允许在 V_0、V_1、V_2中转，最短路径是？——求 $A^{(2)}$ 和 $path^{(2)}$

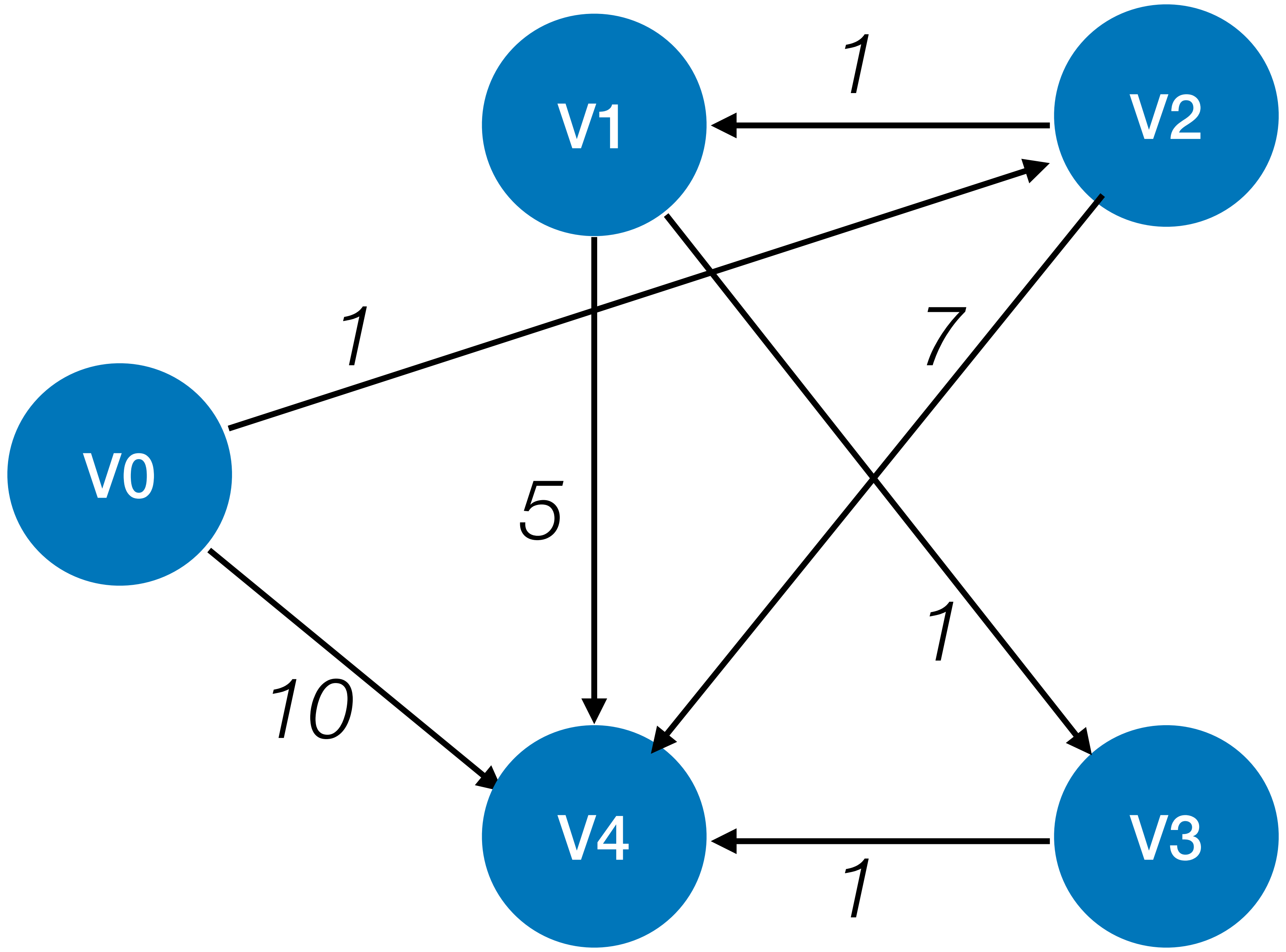
$A^{(2)} =$

	V0	V1	V2	V3	V4
V0	0	2	1	3	7
V1	∞	0	∞	1	5
V2	∞	1	0	2	6
V3	∞	∞	∞	0	1
V4	∞	∞	∞	∞	0

$path^{(2)} =$

	V0	V1	V2	V3	V4
V0	-1	2	-1	2	2
V1	-1	-1	-1	-1	-1
V2	-1	-1	-1	1	1
V3	-1	-1	-1	-1	-1
V4	-1	-1	-1	-1	-1

Floyd算法实例



$A^{(2)} =$

	V0	V1	V2	V3	V4
V0	0	2	1	3	7
V1	∞	0	∞	1	5
V2	∞	1	0	2	6
V3	∞	∞	∞	0	1
V4	∞	∞	∞	∞	0

$path^{(2)} =$

	V0	V1	V2	V3	V4
V0	-1	2	-1	2	2
V1	-1	-1	-1	-1	-1
V2	-1	-1	-1	1	1
V3	-1	-1	-1	-1	-1
V4	-1	-1	-1	-1	-1

#3：若允许在 V_0、V_1、V_2、V_3中转，最短路径是？——求 $A^{(3)}$ 和 $path^{(3)}$

```
for(int i=0; i<n; i++) {  //遍历整个矩阵，i为行号，j为列号
  for (int j=0; j<n; j++){
    if (A[i][j]>A[i][k]+A[k][j]){    //以 Vk 为中转点的路径更短
      A[i][j]=A[i][k]+A[k][j];       //更新最短路径长度
      path[i][j]=k;                  //中转点
    }
  }
}
```

k=3

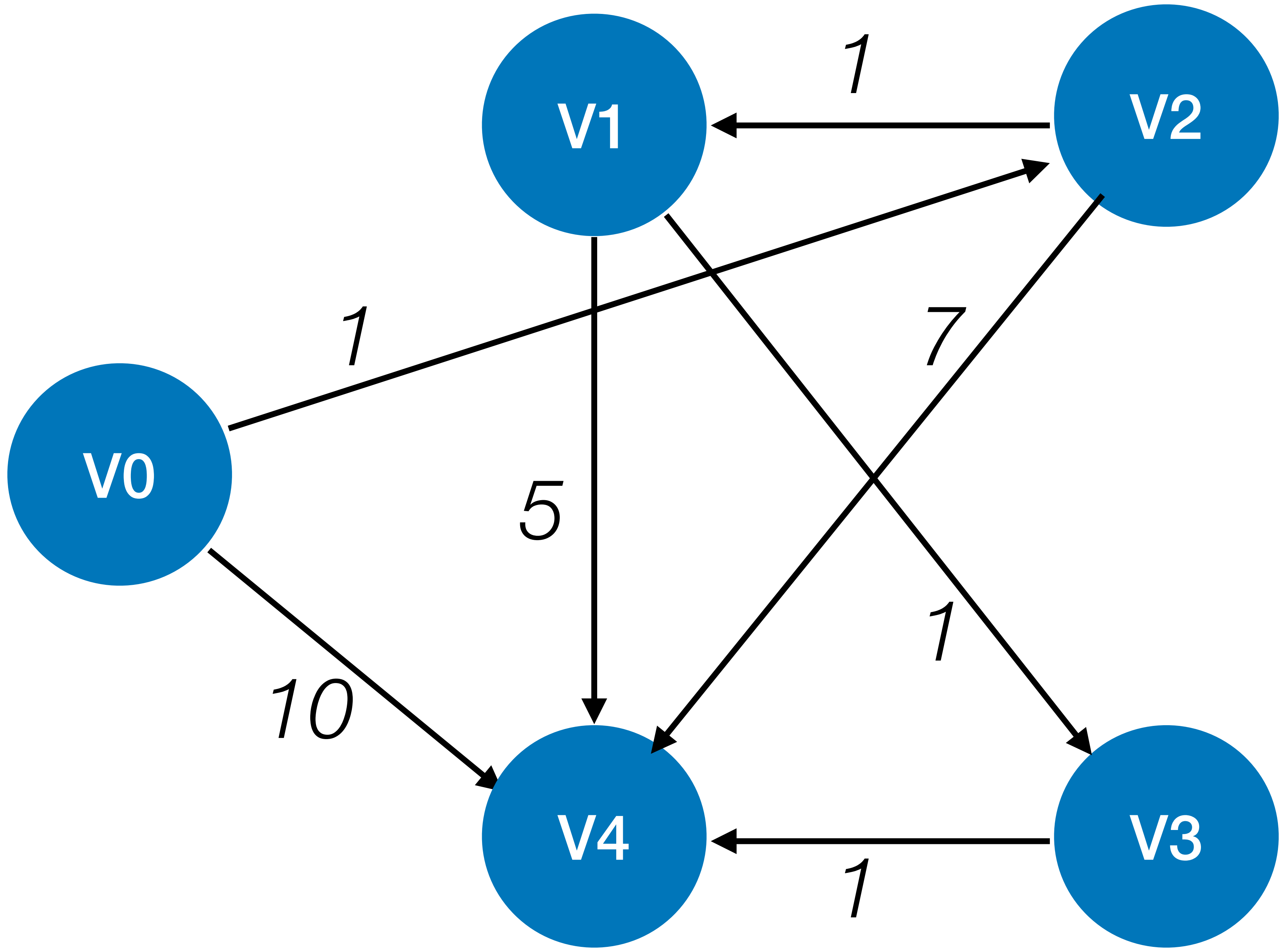
若 $A^{(k-1)}[i][j]>A^{(k-1)}[i][k]+A^{(k-1)}[k][j]$

则 $A^{(k)}[i][j]=A^{(k-1)}[i][k]+A^{(k-1)}[k][j]$;

$path^{(k)}[i][j]=k$

否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

Floyd算法实例



$A^{(2)}$ =

	V0	V1	V2	V3	V4
V0	0	2	1	3	7
V1	∞	0	∞	1	5
V2	∞	1	0	2	6
V3	∞	∞	∞	0	1
V4	∞	∞	∞	∞	0

$path^{(2)}$ =

	V0	V1	V2	V3	V4
V0	-1	2	-1	2	2
V1	-1	-1	-1	-1	-1
V2	-1	-1	-1	1	1
V3	-1	-1	-1	-1	-1
V4	-1	-1	-1	-1	-1

#3：若允许在 V_0、V_1、V_2、V_3中转，最短路径是？——求 $A^{(3)}$ 和 $path^{(3)}$

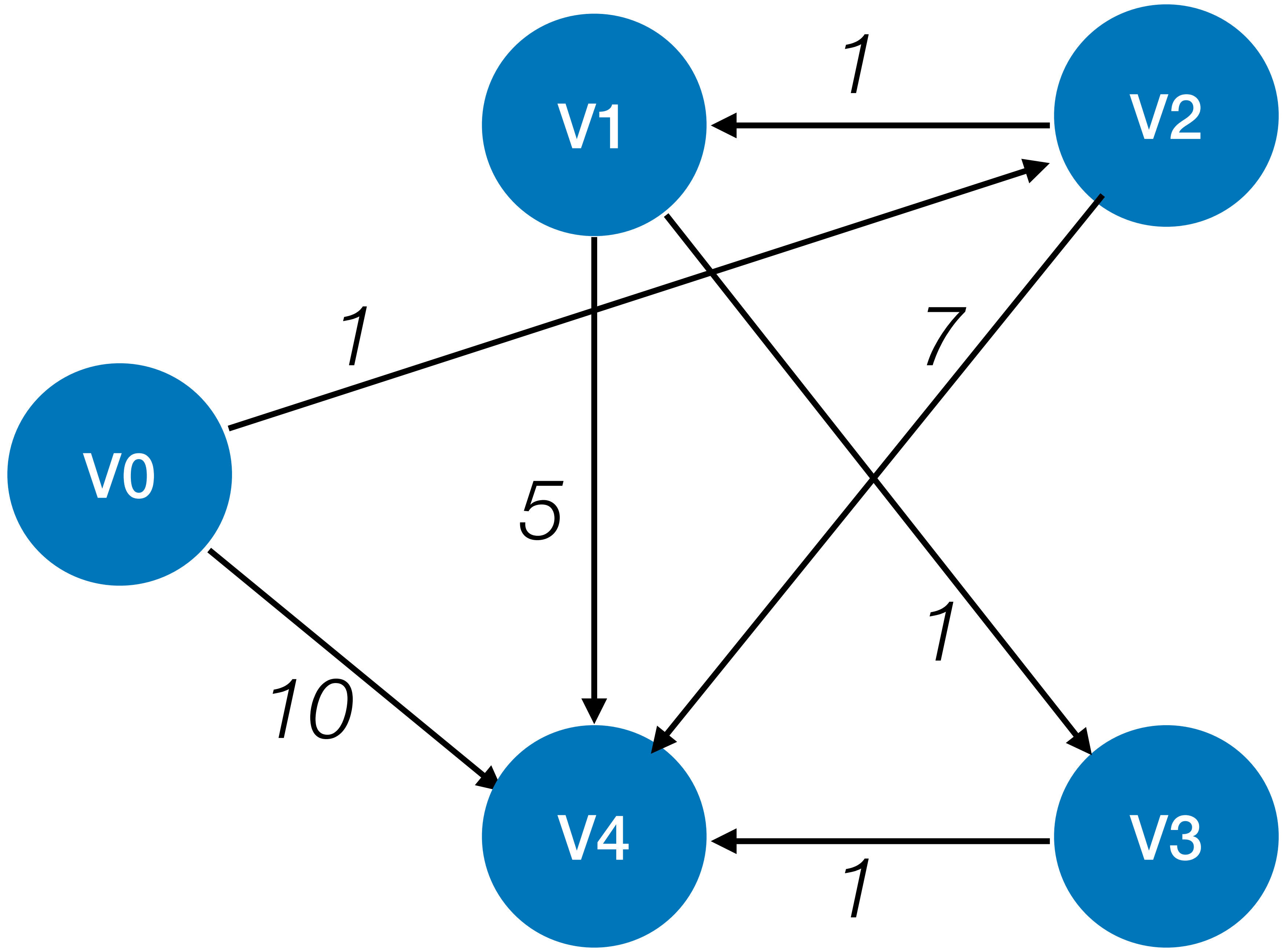
$A^{(2)}[0][4] > A^{(2)}[0][3] + A^{(2)}[3][4] = 4$
$A^{(3)}[0][4] = 4;\ path^{(3)}[0][4] = 3;$

$A^{(2)}[1][4] > A^{(2)}[1][3] + A^{(2)}[3][4] = 2$
$A^{(3)}[1][4] = 2;\ path^{(3)}[1][4] = 3;$

$A^{(2)}[2][4] > A^{(2)}[2][3] + A^{(2)}[3][4] = 3$
$A^{(3)}[2][4] = 3;\ path^{(3)}[2][4] = 3;$

若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$
则 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$;
$path^{(k)}[i][j] = k$
否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

Floyd算法实例



$A^{(2)} =$

	V0	V1	V2	V3	V4
V0	0	2	1	3	7
V1	∞	0	∞	1	5
V2	∞	1	0	2	6
V3	∞	∞	∞	0	1
V4	∞	∞	∞	∞	0

$path^{(2)} =$

	V0	V1	V2	V3	V4
V0	-1	2	-1	2	2
V1	-1	-1	-1	-1	-1
V2	-1	-1	-1	1	1
V3	-1	-1	-1	-1	-1
V4	-1	-1	-1	-1	-1

#3：若允许在 V_0、V_1、V_2、V_3中转，最短路径是？——求 $A^{(3)}$ 和 $path^{(3)}$

若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$

则 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$;

$path^{(k)}[i][j] = k$

否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

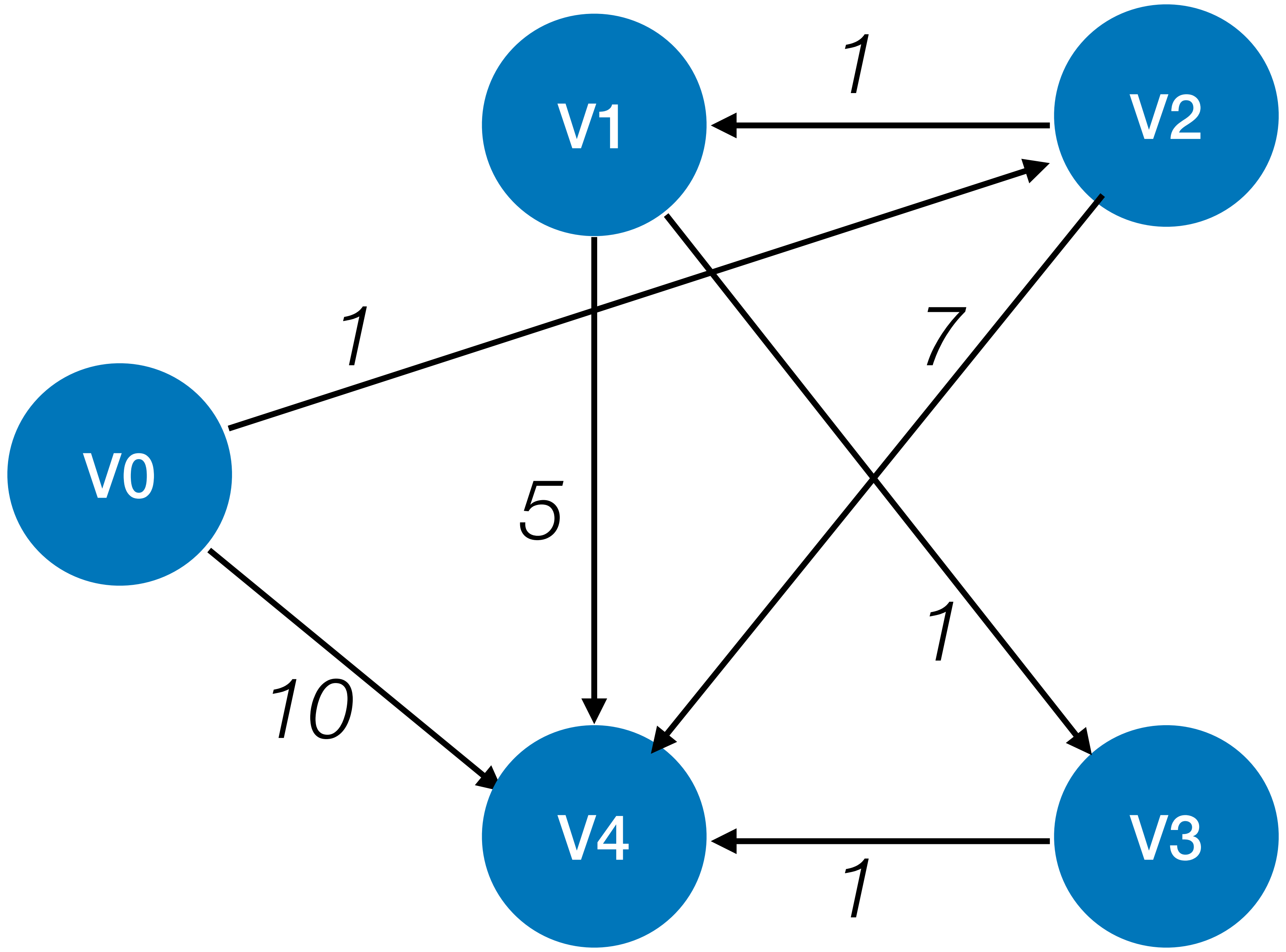
$A^{(3)} =$

	V0	V1	V2	V3	V4
V0	0	2	1	3	4
V1	∞	0	∞	1	2
V2	∞	1	0	2	3
V3	∞	∞	∞	0	1
V4	∞	∞	∞	∞	0

$path^{(3)} =$

	V0	V1	V2	V3	V4
V0	-1	2	-1	2	3
V1	-1	-1	-1	-1	3
V2	-1	-1	-1	1	3
V3	-1	-1	-1	-1	-1
V4	-1	-1	-1	-1	-1

Floyd算法实例



$A^{(3)}$ =

	V0	V1	V2	V3	V4
V0	0	2	1	3	4
V1	∞	0	∞	1	2
V2	∞	1	0	2	3
V3	∞	∞	∞	0	1
V4	∞	∞	∞	∞	0

$path^{(3)}$ =

	V0	V1	V2	V3	V4
V0	-1	2	-1	2	3
V1	-1	-1	-1	-1	3
V2	-1	-1	-1	1	3
V3	-1	-1	-1	-1	-1
V4	-1	-1	-1	-1	-1

#4：若允许在 V_0、V_1、V_2、V_3、V_4中转，最短路径是？——求 $A^{(4)}$ 和 $path^{(4)}$

```
for(int i=0; i<n; i++) {  //遍历整个矩阵，i为行号，j为列号
  for (int j=0; j<n; j++){
    if (A[i][j]>A[i][k]+A[k][j]){    //以 Vk 为中转点的路径更短
      A[i][j]=A[i][k]+A[k][j];       //更新最短路径长度
      path[i][j]=k;                  //中转点
    }
  }
}
```

k=4

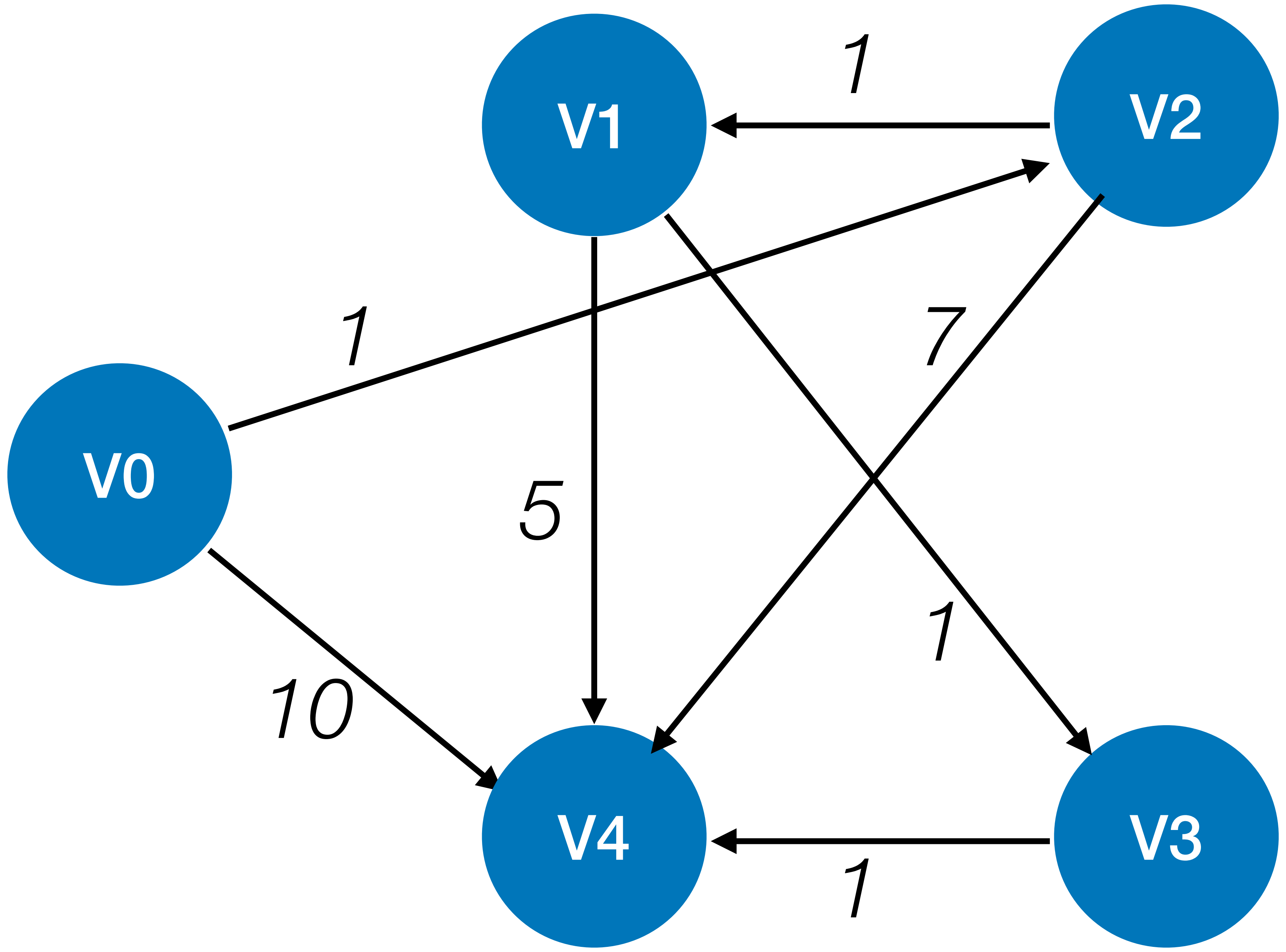
若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$

则 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$;

$path^{(k)}[i][j] = k$

否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

Floyd算法实例



$A^{(3)}$ =

	V0	V1	V2	V3	V4
V0	0	2	1	3	4
V1	∞	0	∞	1	2
V2	∞	1	0	2	3
V3	∞	∞	∞	0	1
V4	∞	∞	∞	∞	0

$path^{(3)}$ =

	V0	V1	V2	V3	V4
V0	-1	2	-1	2	3
V1	-1	-1	-1	-1	3
V2	-1	-1	-1	1	3
V3	-1	-1	-1	-1	-1
V4	-1	-1	-1	-1	-1

#4：若允许在 V_0、V_1、V_2、V_3、V_4中转，最短路径是？——求 $A^{(4)}$ 和 $path^{(4)}$

$A^{(4)}$ =

	V0	V1	V2	V3	V4
V0	0	2	1	3	4
V1	∞	0	∞	1	2
V2	∞	1	0	2	3
V3	∞	∞	∞	0	1
V4	∞	∞	∞	∞	0

$path^{(4)}$ =

	V0	V1	V2	V3	V4
V0	-1	2	-1	2	3
V1	-1	-1	-1	-1	3
V2	-1	-1	-1	1	3
V3	-1	-1	-1	-1	-1
V4	-1	-1	-1	-1	-1

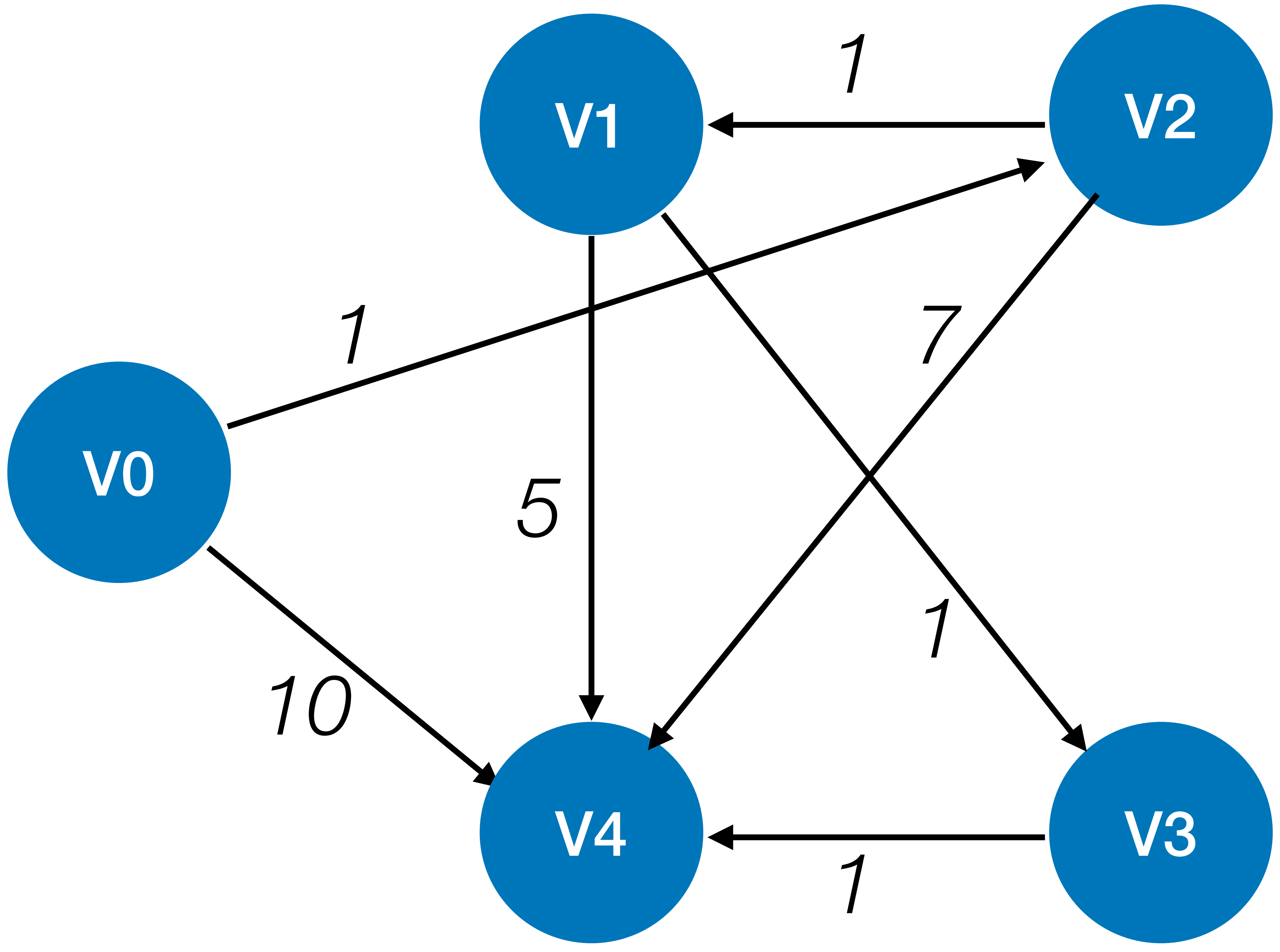
若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$

则 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$;

$path^{(k)}[i][j] = k$

否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

Floyd算法实例



A =

	V0	V1	V2	V3	V4
V0	0	2	1	3	4
V1	∞	0	∞	1	2
V2	∞	1	0	2	3
V3	∞	∞	∞	0	1
V4	∞	∞	∞	∞	0

path =

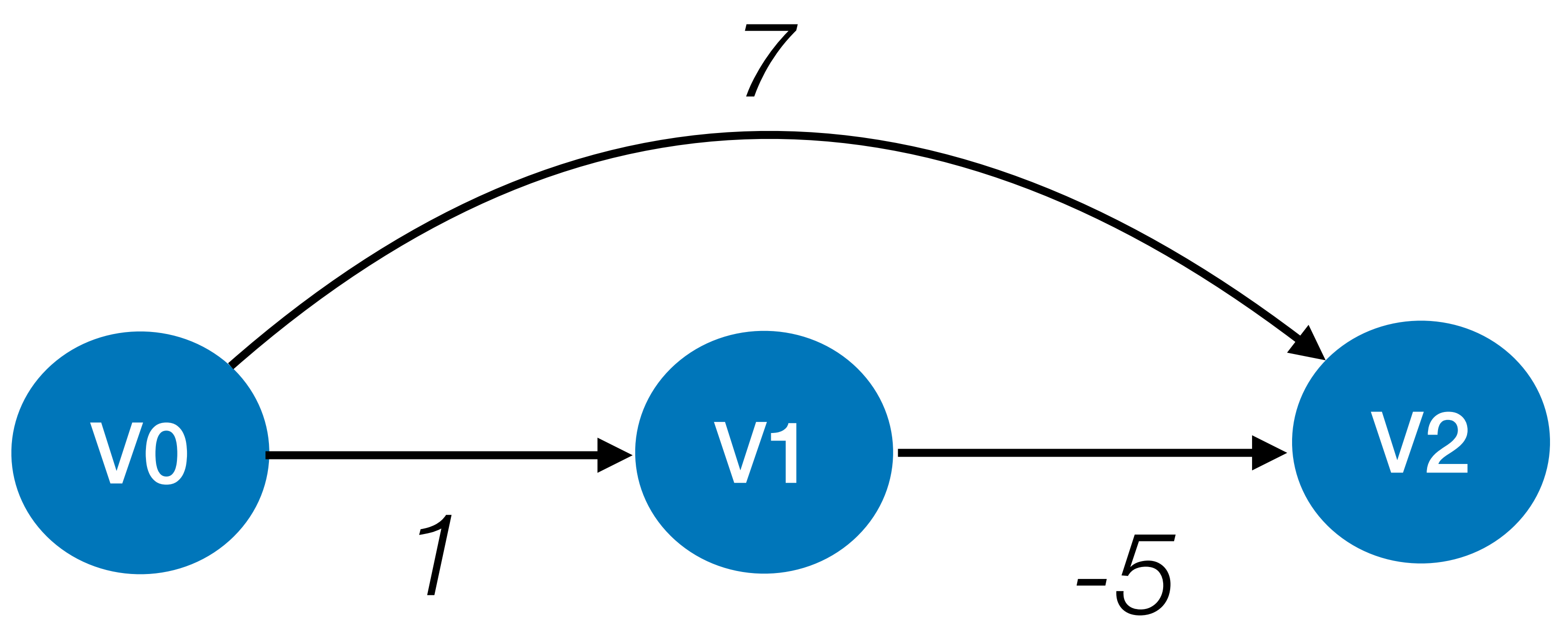
	V0	V1	V2	V3	V4
V0	-1	2	-1	2	3
V1	-1	-1	-1	-1	3
V2	-1	-1	-1	1	3
V3	-1	-1	-1	-1	-1
V4	-1	-1	-1	-1	-1

V0到V4 最短路径长度为 A[0][4]=4

通过path矩阵递归地找到完整路径：

V0 V3 V4
V0 V2 V3 V4
V0 V2 V1 V3 V4

练习：Floyd算法用于负权图



A=

	V0	V1	V2
V0	0	1	7
V1	∞	0	-5
V2	∞	∞	0

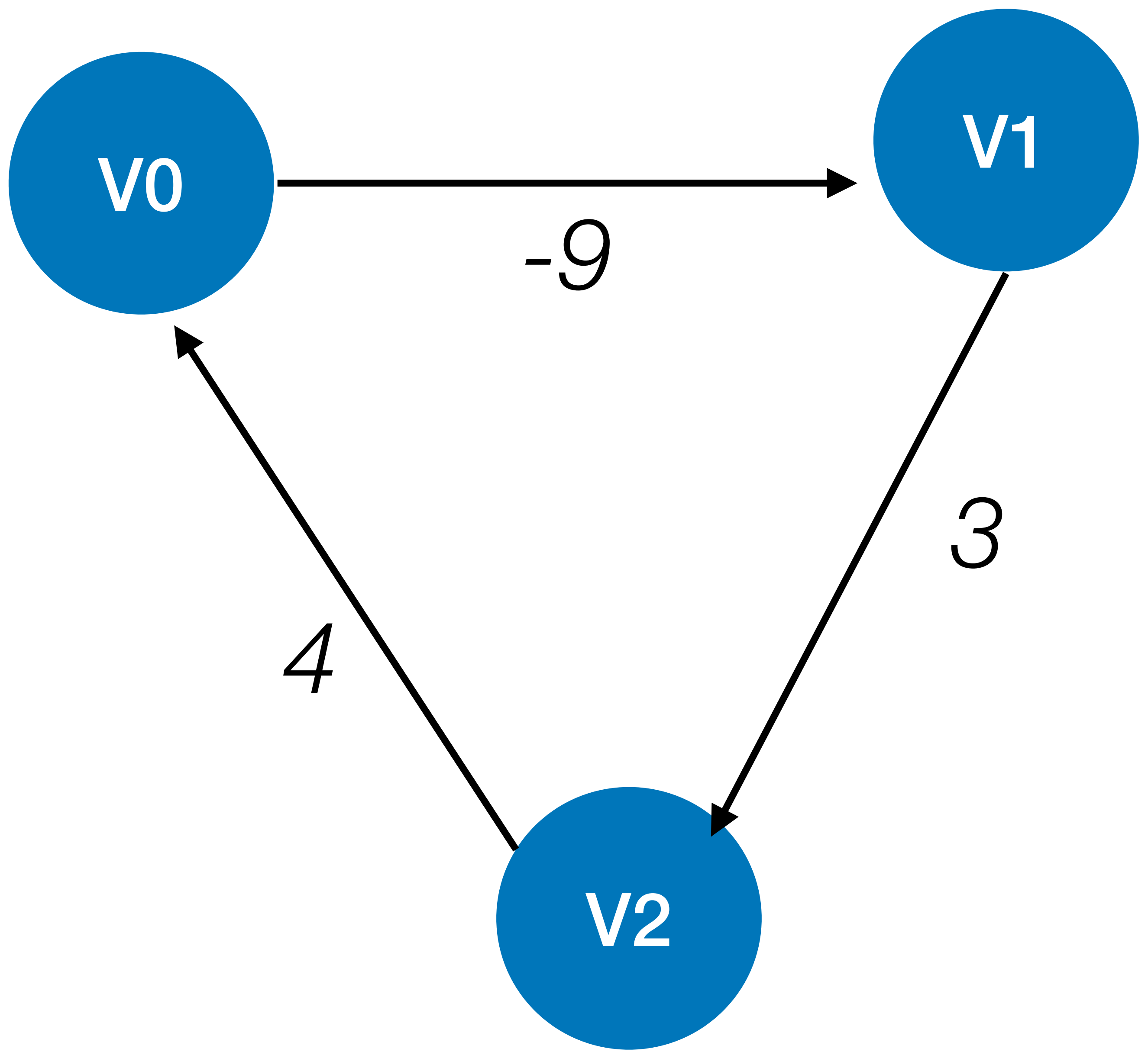
path=

	V0	V1	V2
V0	-1	-1	-1
V1	-1	-1	-1
V2	-1	-1	-1

Floyd算法可以用于负权值带权图

```
//......准备工作，根据图的信息初始化矩阵 A 和 path（如上图）
for (int k=0; k<n; k++){     //考虑以 vk 作为中转点
  for(int i=0; i<n; i++) {  //遍历整个矩阵，i为行号，j为列号
    for (int j=0; j<n; j++){
      if (A[i][j]>A[i][k]+A[k][j]){    //以 vk 为中转点的路径更短
        A[i][j]=A[i][k]+A[k][j];       //更新最短路径长度
        path[i][j]=k;                  //中转点
      }
    }
  }
}
```

不能解决的问题



Floyd 算法不能解决带有“负权回路”的图（有负权值的边组成回路），这种图有可能没有最短路径

知识点回顾与重要考点

	BFS 算法	Dijkstra 算法	Floyd 算法
无权图	✓	✓	✓
带权图	✗	✓	✓
带负权值的图	✗	✗	✓
带负权回路的图	✗	✗	✗
时间复杂度	$O(\|V\|^2)$或$O(\|V\|+\|E\|)$	$O(\|V\|^2)$	$O(\|V\|^3)$
通常用于	求无权图的单源最短路径	求带权图的单源最短路径	求带权图中各顶点间的最短路径

注：也可用 Dijkstra 算法求所有顶点间的最短路径，重复 |V| 次即可，总的时间复杂度也是$O(|V|^3)$

欢迎大家对本节视频进行评价~

学员评分：6.4.2_3 最...

扫一扫二维码打开或分享给好友

– 腾讯文档 –

可多人实时在线编辑，权限安全可控

公众号：王道在线

b站：王道计算机教育

抖音：王道计算机考研