

《算法设计与分析》

第四章 贪心方法

马丙鹏

2024年10月28日



中国科学院大学

University of Chinese Academy of Sciences 1

第四章 贪心方法

- 4.1 一般方法
- 4.2 背包问题
- 4.3 带有限期的作业排序
- 4.4 最优归并模式
- 4.5 最小生成树
- 4.6 单源点最短路径



4.5 最小生成树

■ 1. 问题的描述

□ 设 $G=(V, E)$ 是一个无向连通图。

□ 子图：

➤ 从原图中删去一些点或删去一些线或既删去一些点又删去一些线，剩下的部分(当然必须仍然是图)。

□ 生成子图：

➤ 同“子图”，但只允许删去线，不允许删去点。

□ 生成树：

➤ 如果 G 的生成子图 $T=(V, E')$ 是一棵树，则称 T 是 G 的一棵生成树。

□ 最小生成树：

➤ G 中具有最小成本的生成树。

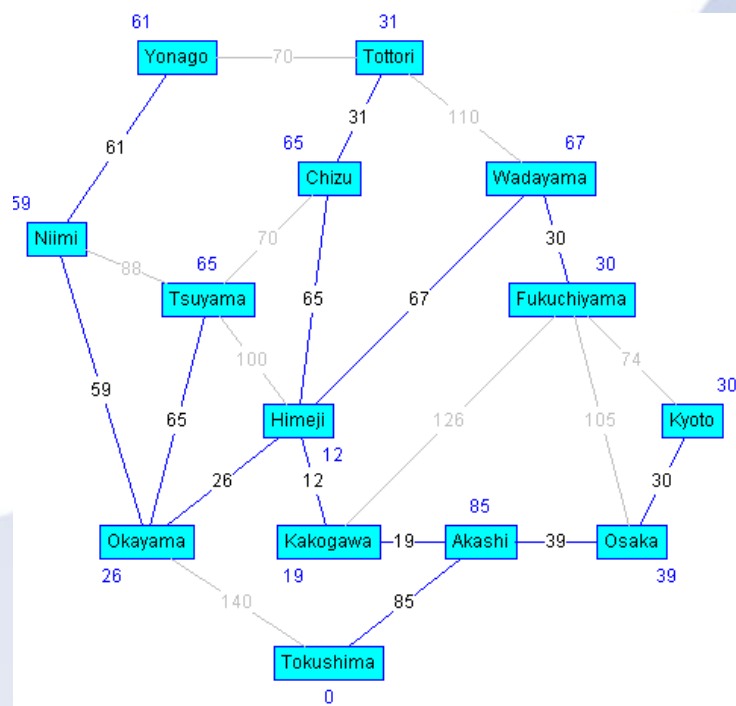
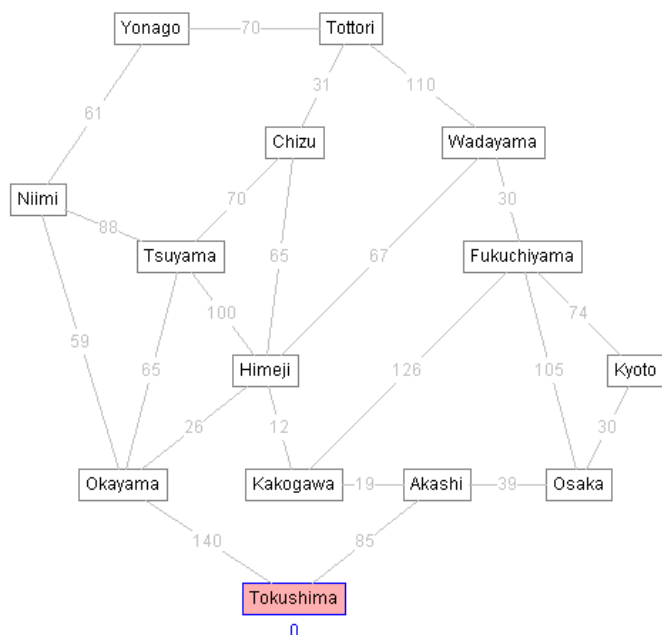


4.5 最小生成树

■ 2. 贪心策略

□ 实际应用:

- 在设计通信网络时，用图的顶点表示城市，用边的权表示建立城市和城市之间的通信线路所需的费用，则最小生成树就给出了建立通信网络的最经济的方案



4.5 最小生成树

■ 2. 贪心策略

□ 度量标准:

- 选择能使迄今为止所计入的边的**成本和**有**最小增加**的那条边。
- Prim算法
- Kruskal算法

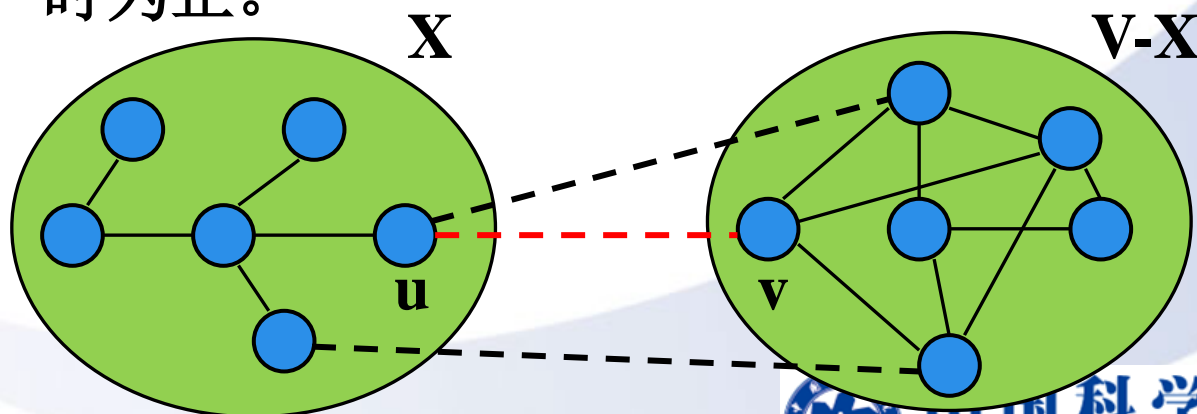


4.5 最小生成树

■ 3. Prim算法

□策略：使得迄今所选择的边的集合 A 构成一棵树；对将要计入到 A 中的下一条边 (u, v) ，应是 E 中一条当前不在 A 中且使得 $A \cup \{(u, v)\}$ 也是一棵树的**最小成本边**。

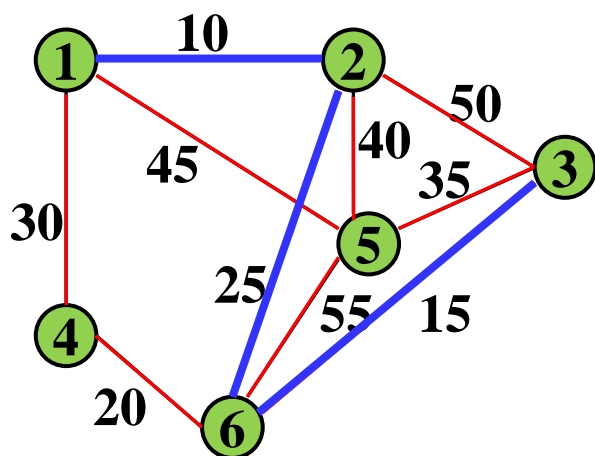
- 设 X 为已部分构建的最小生成树的顶点集合
- 选取满足条件 $u \in X, v \in V-X$ ，且成本最小的边，并将顶点 v 添加到 X 中。这个过程一直进行到 $X=V$ 时为止。



4.5 最小生成树

■ 3. Prim算法

□策略：使得迄今所选择的边的集合A构成一棵树；对将要计入到A中的下一条边(u, v)，应是E中一条当前不在A中且使得 $A \cup \{(u, v)\}$ 也是一棵树的最小成本边。



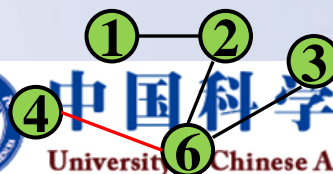
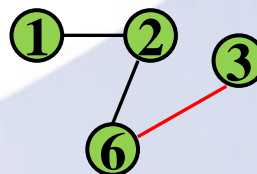
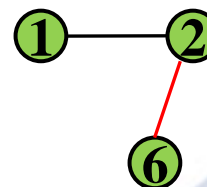
边 成本

(1, 2) 10

(2, 6) 25

(3, 6) 15

(6, 4) 20



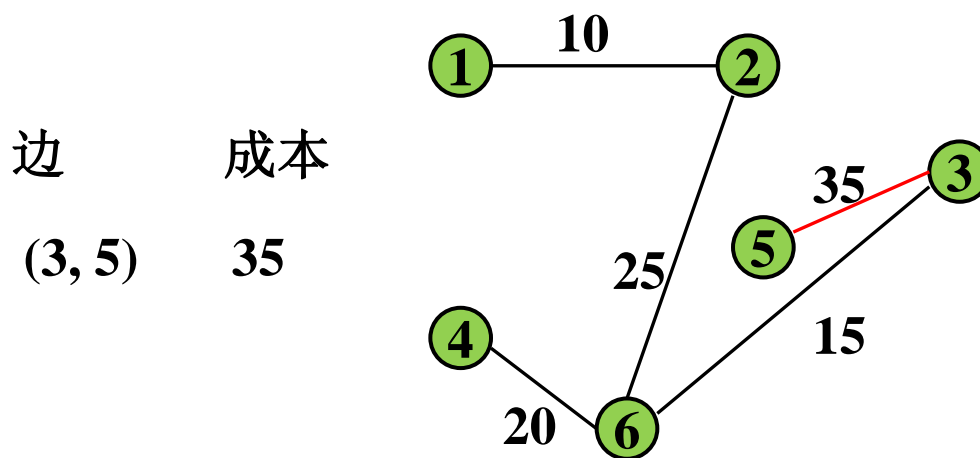
中国科学院大学

University of Chinese Academy of Sciences 7

4.5 最小生成树

■ 3. Prim算法

□策略：使得迄今所选择的边的集合A构成一棵树；对将要计入到A中的下一条边(u, v)，应是E中一条当前不在A中且使得 $A \cup \{(u, v)\}$ 也是一棵树的最小成本边。



➤ $V(T_P) = \{1, 2, 3, 4, 5, 6\}$

➤ $E(T_P) = \{ (1, 2), (2, 6), (3, 5), (4, 6), (3, 6) \}$



算法4.7 Prim最小生成树算法

procedure PRIM($E, \text{COST}, n, T, \text{mincost}$)

// E 是 G 的边集。 $\text{COST}(n, n)$ 是 n 结点图 G 的**成本邻接矩阵**，矩阵元素 $\text{COST}(i, j)$ 是一个正实数，如果不存在边 (i, j) ，则为 $+\infty$ 。计算一棵最小生成树并把它作为一个集合存放到数组 $T(1:n-1, 2)$ 中($T(i, 1), T(i, 2)$)是最小成本生成树的一条边。最小成本生成树的总成本最后赋给 mincost //

```
1  real  $\text{COST}(n, n), \text{mincost}$ 
2  integer  $\text{NEAR}(n), n, i, k, l, T(1:n-1, 2)$ 
3   $(k, l) \leftarrow$  具有最小成本的边
4   $\text{mincost} \leftarrow \text{COST}(k, l)$ 
5   $(T(1, 1), T(1, 2)) \leftarrow (k, l)$ 
6  for  $i \leftarrow 1$  to  $n$  do //将NEAR置初值//
7      if  $\text{COST}(i, l) < \text{COST}(i, k)$  then  $\text{NEAR}(i) \leftarrow l$ 
8          else  $\text{NEAR}(i) \leftarrow k$ 
9      endif
10 repeat
11  $\text{NEAR}(k) \leftarrow \text{NEAR}(l) \leftarrow 0$ 
```



```

12  for i←2 to n-1 do //找T的其余n-2条边//
13      设j是NEAR(j)≠0 且COST(j, NEAR(j))最小的下标
14      (T(i, 1), T(i, 2))←(j, NEAR(j))
15      mincost←mincost + COST(j, NEAR(j))
16      NEAR(j)←0
17      for k←1 to n do //修改NEAR//
18          if NEAR(k)≠0 and COST(k, NEAR(k))>COST(k, j)
19              then NEAR(k)←j
20          endif
21      repeat
22  repeat
23      if mincost>∞ then print('no spanning tree') endif
24  end PRIM

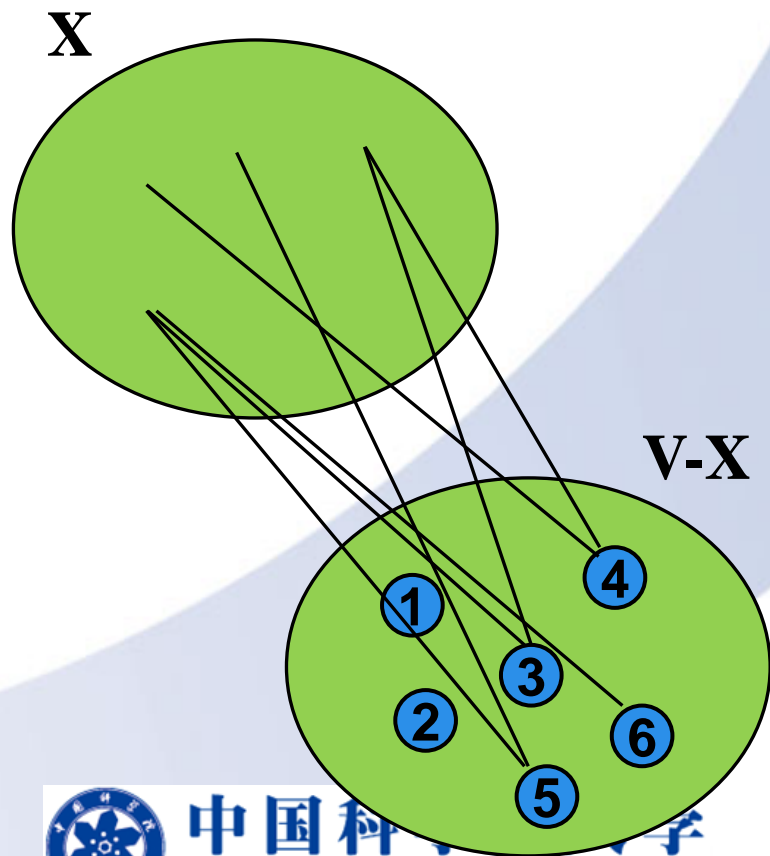
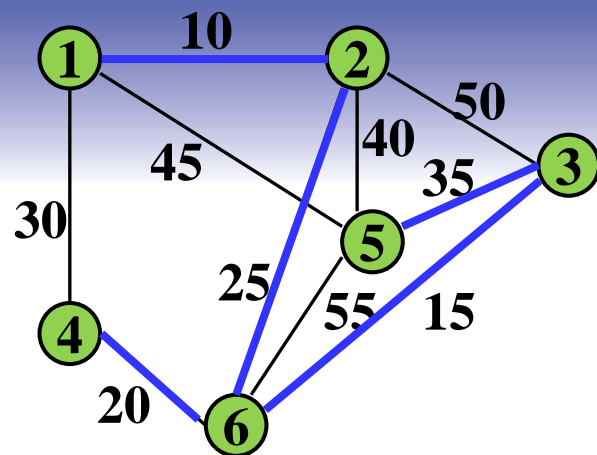
```



4.5 最小生成树

■ 3. Prim算法

i	1	2	3	4	5	6
NEAR	0	0	2	1	2	2
COST	0	0	50	30	40	25
NEAR	0	0	2	1	2	0
NEAR	0	0	6	6	2	0
COST	0	0	15	20	40	0
NEAR	0	0	0	6	2	0
NEAR	0	0	0	6	3	0
COST	0	0	0	20	35	0
NEAR	0	0	0	0	3	0



4.5 最小生成树

■ 3. Prim算法

□ 计算复杂性:

- 第3行花费 $\Theta(e)$ ($e=|E|$)时间,
- 第4行花费 $\Theta(1)$ 时间;
- 第6-9行的循环花费 $\Theta(n)$ 时间;
- 第12行和第17-21行的循环分别要求 $\Theta(n)$ 时间, 因此第12-21行循环要花费 $\Theta(n^2)$ 时间。
- 所以PRIM算法具有 $\Theta(n^2)$ 的时间复杂度



4.5 最小生成树

■ 3. Prim算法

□ 另一种PRIM算法

➤ 最小生成树中包含了与每个结点 v 相关的一条最小成本边。

证明略。

➤ 方法：

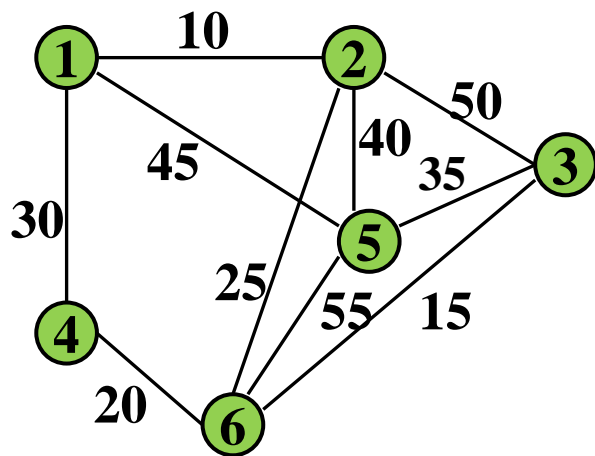
- ✓ 从一棵包含任何一个随意指定的结点而没有边的树开始这一算法，
- ✓ 然后再逐条增加边。



4.5 最小生成树

■ 4. Kruskal算法

□ (连通)图的边按成本的非降次序排列，下一条计入生成树T中的边是还没有计入的边中具有最小成本、且和T中现有的边不会构成环路的边。

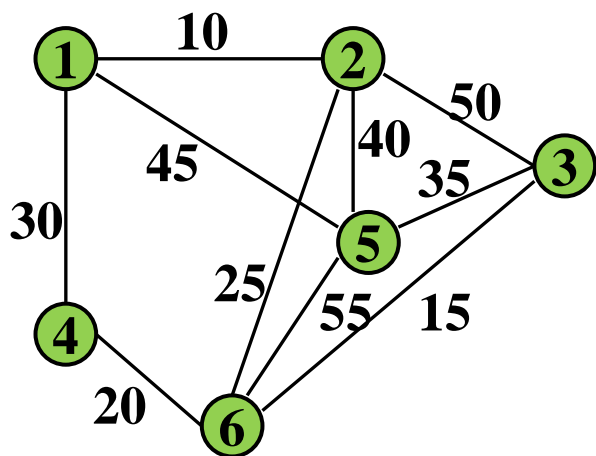


边	成本
(1, 2)	10
(3, 6)	15
(4, 6)	20
(2, 6)	25
(1, 4)	30
(3, 5)	35
(2, 5)	40
(1, 5)	45
(2, 3)	50
(5, 6)	55

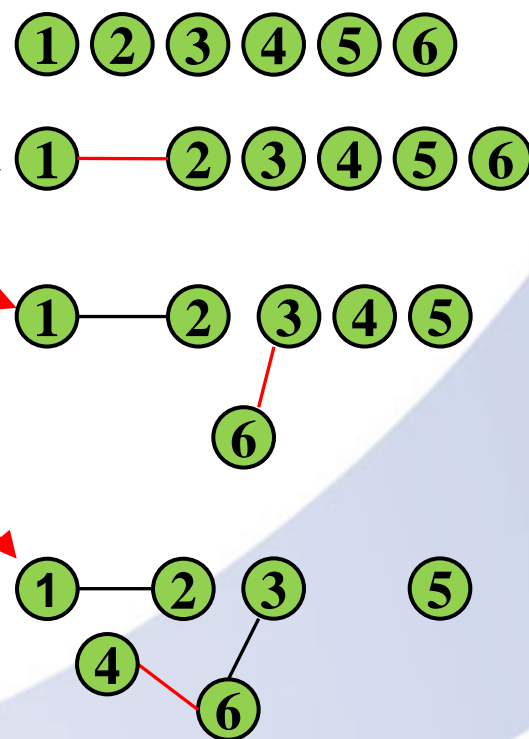


4.5 最小生成树

■ 4. Kruskal算法



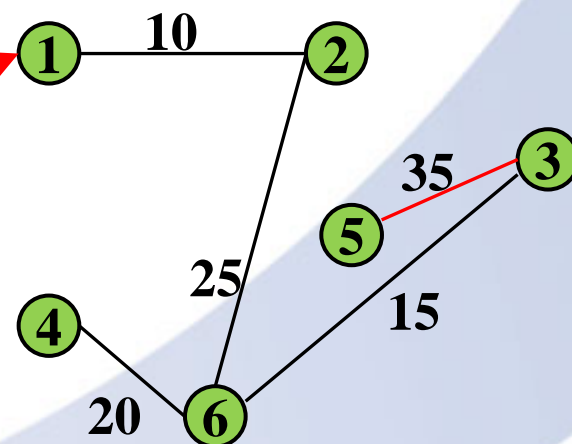
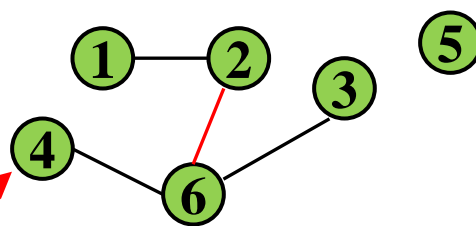
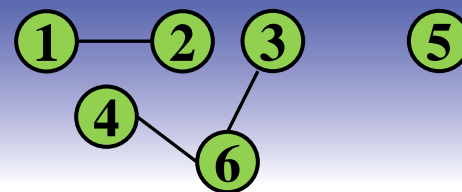
边	成本
(1, 2)	10
(3, 6)	15
(4, 6)	20
(2, 6)	25
(1, 4)	30
(3, 5)	35
(2, 5)	40
(1, 5)	45
(2, 3)	50
(5, 6)	55



4.5 最小生成树

■ 4. Kruskal算法

边	成本
(1, 2)	10
(3, 6)	15
(4, 6)	20
(2, 6)	25
(1, 4)	30
(3, 5)	35
(2, 5)	40
(1, 5)	45
(2, 3)	50
(5, 6)	55



➤ $V(T_K) = \{1, 2, 3, 4, 5, 6\}$

➤ $E(T_K) = \{ (1, 2), (2, 6), (3, 5), (4, 6), (3, 6) \}$



中国科学院大学

University of Chinese Academy of Sciences 16

4.5 最小生成树

■ 4. Kruskal算法

□Kruskal算法的概略描述

算法4.8 Kruskal算法的概略描述

```
1      T ← 空
2      while T的边少于n-1 do
3          从E中选取一条最小成本的边(v, w)
4          从E中删除(v, w)
5          if (v, w)在T中不生成环
6              then 将(v, w)加入到T中
7          else 舍弃(v, w)
8          endif
9      repeat
```



算法4.9 Kruskal算法

procedure KRUSKAL($E, \text{COST}, n, T, \text{mincost}$)

real mincost, $\text{COST}(1:n, 1:n);$

integer PARENT($1:n$), $T(1:n-1, 2), n;$

以边成本为元素构造一个min堆

PARENT $\leftarrow -1$ //每个结点都在不同的集合中//

$i \leftarrow \text{mincost} \leftarrow 0$

while $i < n-1$ **and** 堆非空 **do**

从堆中删去最小成本边 (u, v) 并重新构造堆

$j \leftarrow \text{FIND}(u); k \leftarrow \text{FIND}(v)$

if $(j \neq k)$ **then** $i \leftarrow i+1$

$T(i, 1) \leftarrow u; T(i, 2) \leftarrow v$

$\text{mincost} \leftarrow \text{mincost} + \text{COST}(u, v)$

call UNION(j, k)

endif

repeat

if $i \neq n-1$ **then** print('no spanning tree') **endif**

return

end KRUSKAL

//G有 n 个结点, E 是G的边集。
 $\text{COST}(u, v)$ 是边 (u, v) 的成本。
 T 是最小成本生成树的边集,
 mincost 是它的成本//



中国科学院大学

University of Chinese Academy of Sciences 18

4.5 最小生成树

■ 4. Kruskal算法

□注:

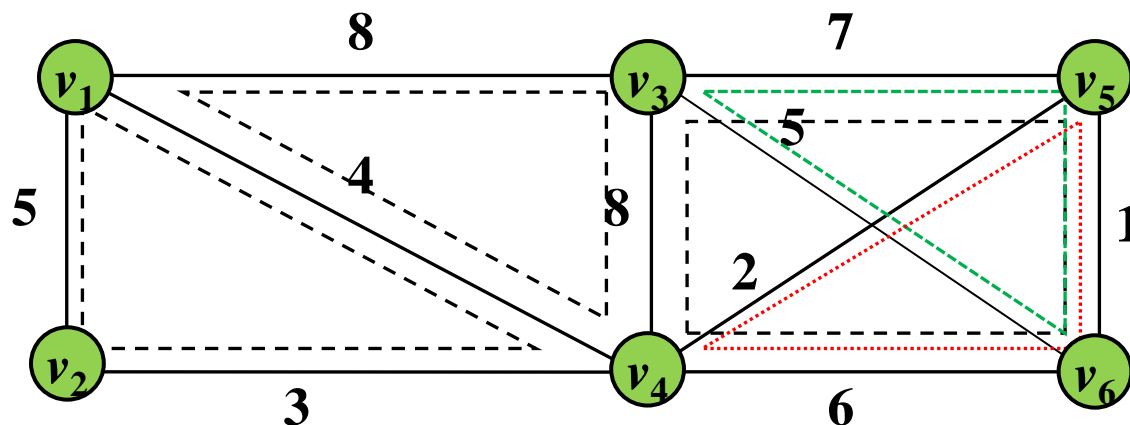
- **FIND(i)**:查找含有元素*i*的树根,
- **UNION(i, j)**:使用加权规则合并根为*i*和*j*的两个树
- 边集以min-堆的形式保存, 一条当前最小成本边可以在 $O(\log e)$ 的时间内找到;
- 算法的计算时间是 $O(e \log e)$ 。



4.5 最小生成树

■ 5. 破圈法

□ 任取一圈，去掉圈中最长边，直到无圈。



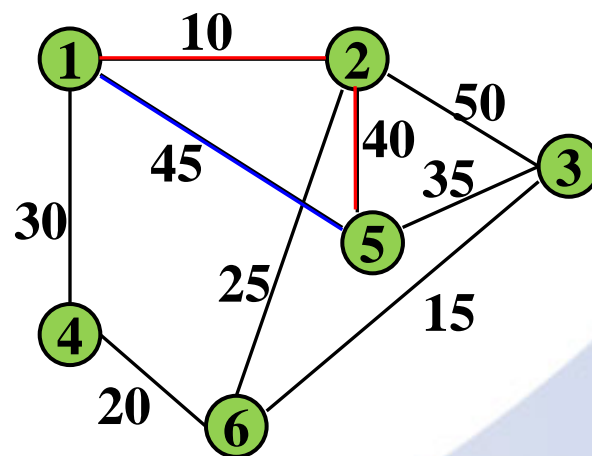
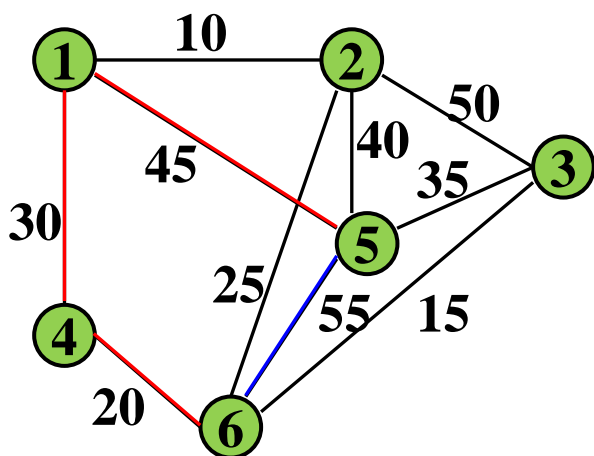
最小树长为 $C(T)=4+3+5+2+1=15$ 。



4.5 最小生成树

■ 5. 破圈法

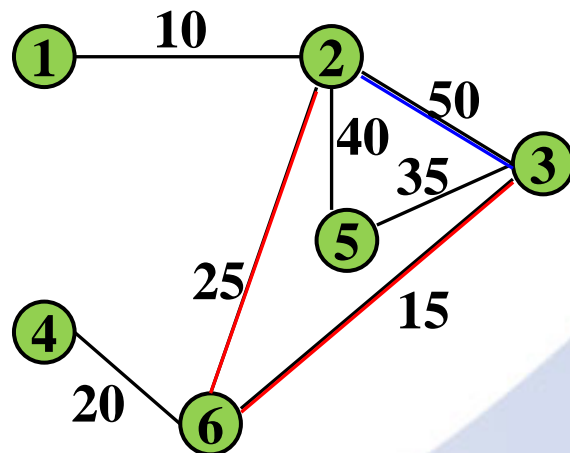
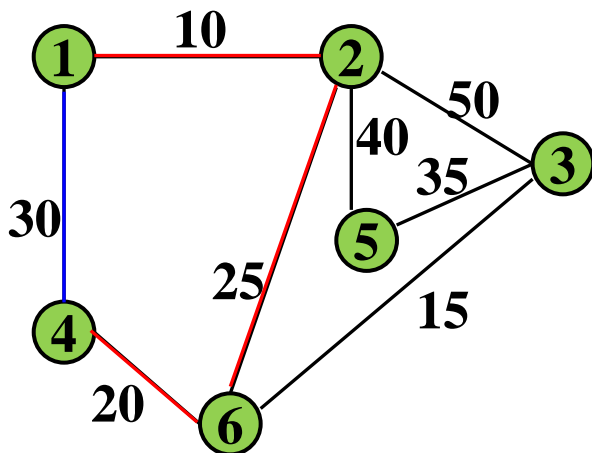
□ 任取一圈，去掉圈中最长边，直到无圈。



4.5 最小生成树

■ 5. 破圈法

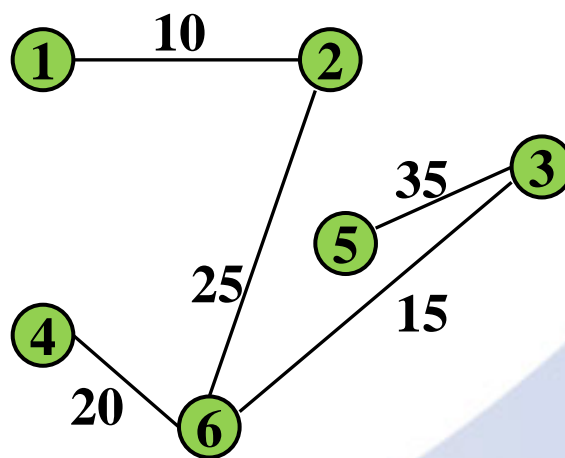
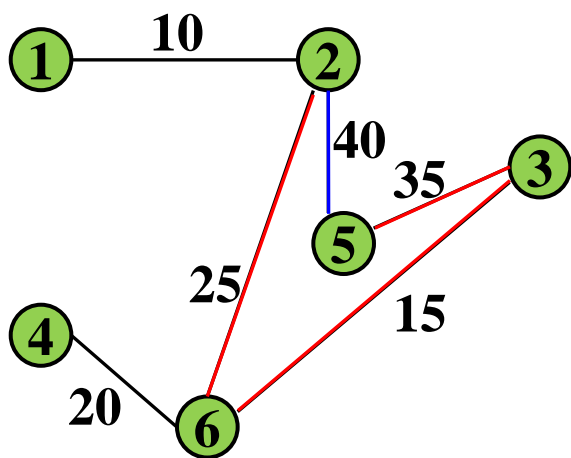
□ 任取一圈，去掉圈中最长边，直到无圈。



4.5 最小生成树

■ 5. 破圈法

□ 任取一圈，去掉圈中最长边，直到无圈。



第四章 贪心方法

- 4.1 一般方法
- 4.2 背包问题
- 4.3 带有限期的作业排序
- 4.4 最优归并模式
- 4.5 最小生成树
- 4.6 单源点最短路径



4.6 单源点最短路径

■ 1. 问题描述

□ 最短路径问题

- 单源点最短路径问题
- 每对结点之间的路径问题
- 特定线路下的最短路径问题等

□ 单源点最短路径问题

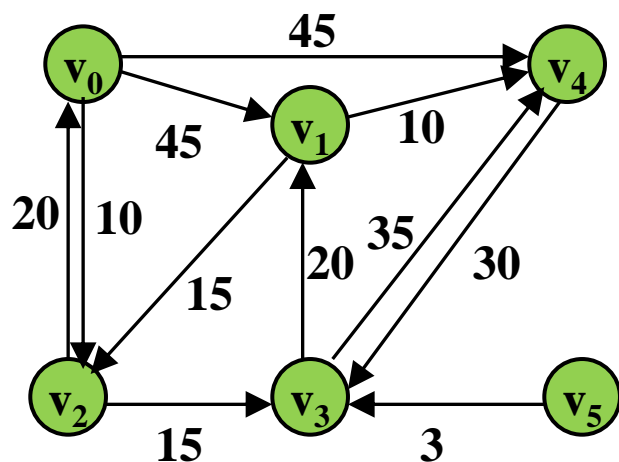
- 已知一个 n 结点有向图 $G=(V, E)$ 和边的权函数 $c(e)$, 求由 G 中某指定结点 v_0 到其它各结点的最短路径。
- 路径长度: 路径上所有边的权值之和。
- 最短路径: 具有最小长度的路径。
- 假定边的权值为正。



4.6 单源点最短路径

■ 1. 问题描述

□ 例4.10 如图所示。设 v_0 是起始点，求 v_0 到其它各结点的最短路径。



	路径	长度
(1)	v_0v_2	10
(2)	$v_0v_2v_3$	25
(3)	$v_0v_2v_3v_1$	45
(4)	v_0v_4	45

➤ 注：路径按照长度的非降次序给出



4.6 单源点最短路径

■ 2. 贪心策略求解

□ 度量标准

➤ 度量标准的选择：

✓ 逐条构造最短路径，可以使用迄今已生成的**所有路径长度之和**作为度量，

✓ ——为使之达到最小，其中任意一条路径都应具有最小长度。

➤ 假定已经构造了*i*条最短路径，则下一条要构造的路径应是下一条最短的路径。

➤ 处理规则：

✓ 按照**路径长度**的**非降次序**依次生成从结点 v_0 到其它各结点的最短路径。

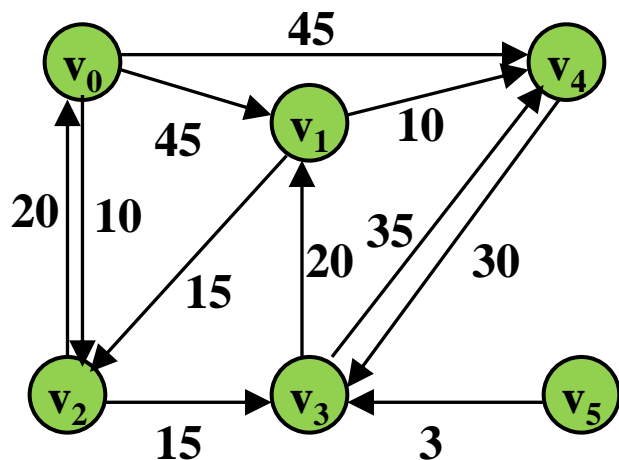


4.6 单源点最短路径

■ 2. 贪心策略求解

□ 度量标准

➤ 例



问题：如何对尚未生成的路径长度进行排序，以确定其中最短者？

路径	长度
(1) v_0v_2	10
(2) $v_0v_2v_3$	25
(3) $v_0v_2v_3v_1$	45
(4) v_0v_4	45

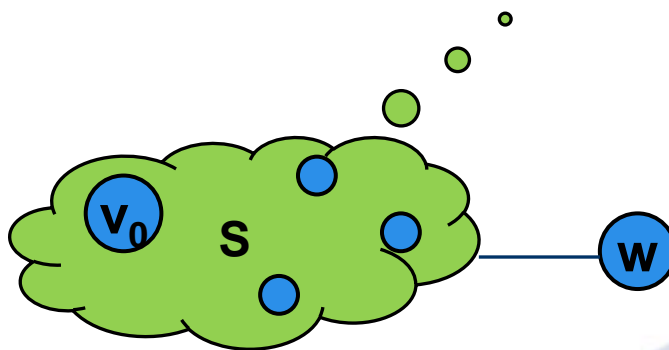


4.6 单源点最短路径

■ 2. 贪心策略求解

□ 贪心算法

- 设 **S** 是已经生成了最短路径的 **结点集合** (包括 v_0)
- 对于当前不在 **S** 中的结点 **w**, 记 **DIST(w)** 是从 v_0 开始, **只经过** **S** 中的结点而在 **w** 结束的那条最短路径的长度。



- 则有,



4.6 单源点最短路径

■ 2. 贪心策略求解

□ 贪心算法

(1) 如果下一条最短路径是到结点 u ，则这条路径是从结点 v_0 出发在 u 处终止，且只经过那些在 S 中的结点，即由 v_0 至 u 的这条最短路径上的所有中间结点都是 S 中的结点：

$v_0, \underline{s_1, s_2, \dots, s_{m-1}}, u$



均在 S 中

➤ 证明：设 w 是这条路径上的任意中间结点，则从 v_0 到 u 的路径也包含了一条从 v_0 到 w 的路径，且其长度小于从 v_0 到 u 的路径长度。

$v_0, s_1, s_2, \dots, \mathbf{w}, \dots, s_{m-1}, u$



中国科学院大学

University of Chinese Academy of Sciences 30

4.6 单源点最短路径

■ 2. 贪心策略求解

□ 贪心算法

➤ 根据生成规则：

✓ 最短路径是按照路径长度的非降次序生成的，因此从 v_0 到 w 的最短路径应该已经生成。

✓ 从而 w 也应该在 S 中。

➤ 故，不存在不在 S 中的中间结点。

(2) 所生成的下一条路径的终点 u 必定是所有不在 S 内的结点中且具有最小距离 $DIST(u)$ 的结点。



4.6 单源点最短路径

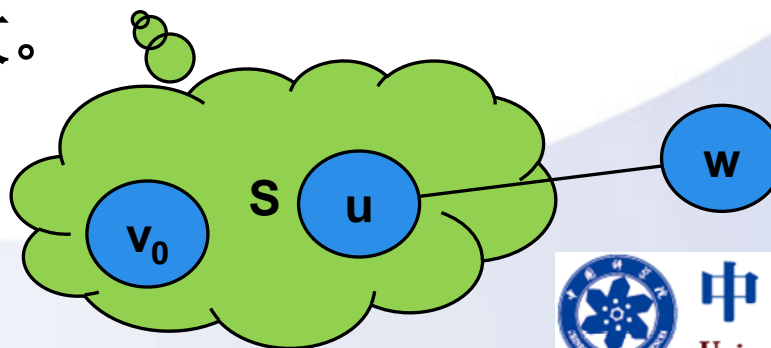
■ 2. 贪心策略求解

□ 贪心算法

(3) 如果选出了这样结点 u 并生成了从 v_0 到 u 的最短路径之后，结点 u 将成为 S 中的一个成员。

此时，那些从 v_0 出发，**只经过** S 中的结点并且在 S 外的结点 w 处结束的最短路径可能会**减少**—— $\text{DIST}(w)$ 的值变小：

➤ 如果这样的路径的长度发生了改变，则这些路径必定是一条从 v_0 开始，经过 u 然后到 w 的更短的路径所致。



4.6 单源点最短路径

■ 2. 贪心策略求解

□ 贪心算法

- 根据 $\text{DIST}(w)$ 的定义，它所表示的 v_0 至 w 的最短路径上的所有中间结点都在 S 中；
- 故只考虑 $\langle u, w \rangle \in E$ 和 $\langle u, w \rangle \notin E$ 的情况
- 对于从 v_0 至 w ，且经过最后一个中间结点为 u 的最短路径，有

$$\text{DIST}(w) = \text{DIST}(u) + c(u, w)$$

- 随着 u 的加入， $\text{DIST}(w)$ 调整为

$$\text{DIST}(w) = \min(\text{DIST}(w), \text{DIST}(u) + c(u, w))$$



算法4.10 生成最短路径的贪心算法

procedure SHORTEST-PATHS(v , $COST$, $DIST$, n)

boolean $S(1:n)$;

real $COST(1:n, 1:n)$, $DIST(1:n)$

integer u, v, n, num, i, w

for $i \leftarrow 1$ **to** n **do** //将集合 S 初始化为空//

$S(i) \leftarrow 0$; $DIST(i) \leftarrow COST(v, i)$

repeat

$S(v) \leftarrow 1$; $DIST(v) \leftarrow 0$ //结点 v 计入 S //

for $num \leftarrow 2$ **to** $n-1$ **do** //确定由结点 v 出发的 $n-1$ 条路//

选取结点 u ,它使得 $DIST(u) = \min_{S(w)=0} \{DIST(w)\}$

$S(u) \leftarrow 1$ //结点 u 计入 S //

for 所有 $S(w) = 0$ 的结点 w **do** //修改 $DIST(w)$ //

$DIST(w) = \min(DIST(w), DIST(u) + COST(u, w))$

repeat

repeat

end SHORTEST-PATHS

// G 是一个 n 结点有向图，它由其成本邻接矩阵 $COST(n, n)$ 表示， $DIST(j)$ 被置以结点 v 到结点 j 的最短路径长度，这里 $1 \leq j \leq n$ 。 $DIST(v)$ 被置成零//



中国科学院大学

University of Chinese Academy of Sciences 34

4.6 单源点最短路径

■ 2. 贪心策略求解

□ 计算时间

➤ 算法4.10的计算时间是: $O(n^2)$

(1) **for** $i \leftarrow 1$ **to** n **do** $\longrightarrow \Theta(n)$

$S(i) \leftarrow 0; \text{DIST}(i) \leftarrow \text{COST}(v, i)$

repeat

(2) **for** $\text{num} \leftarrow 2$ **to** $n-1$ **do** $\longrightarrow O(n-2)$

选取结点 u , 它使得 $\text{DIST}(u) = \min_{S(w)=0} \{ \text{DIST}(w) \} \longrightarrow O(n)$

$S(u) \leftarrow 1$

for 所有 $S(w) = 0$ 的结点 w **do** $\longrightarrow O(n)$

$\text{DIST}(w) = \min(\text{DIST}(w), \text{DIST}(u) + \text{COST}(u, w))$

repeat

repeat



中国科学院大学

University of Chinese Academy of Sciences 35

4.6 单源点最短路径

■ 2. 贪心策略求解

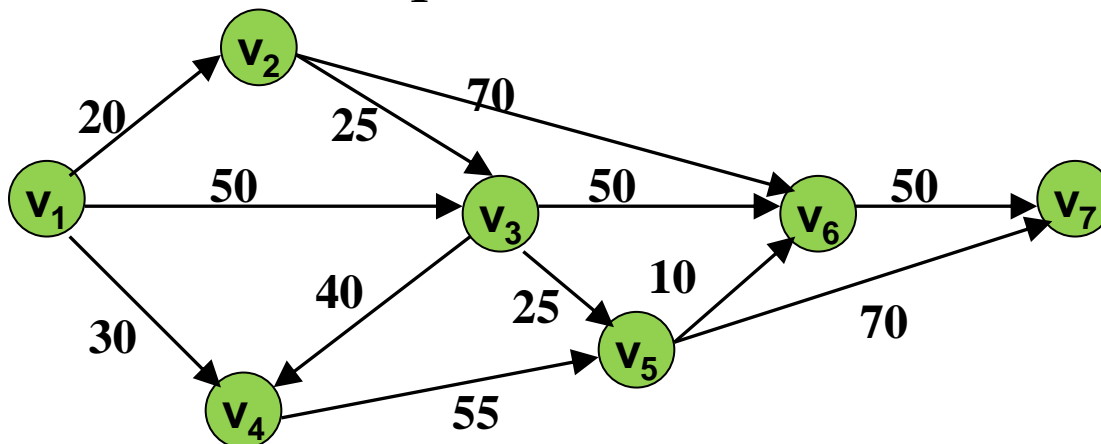
□ 计算时间

➤ 最短路径算法的时间复杂度

- ✓ 由于任何一条边都有可能是最短路径中的边，所以任何最短路径算法都必须至少检查图中的每条边一次，所以这样的算法的最小时间是 $O(e)$ 。
- ✓ 由于用邻接矩阵表示图，要确定哪些边在图中正好需要 $O(n^2)$ 时间，因此任何使用这种表示法的最短路径算法必定花费 $O(n^2)$ 时间。
- ✓ 算法 SHORTEST-PATHS 在常因子范围内是最优的。



例4.11 求下图中从 v_1 出发到其余各结点的最短路径



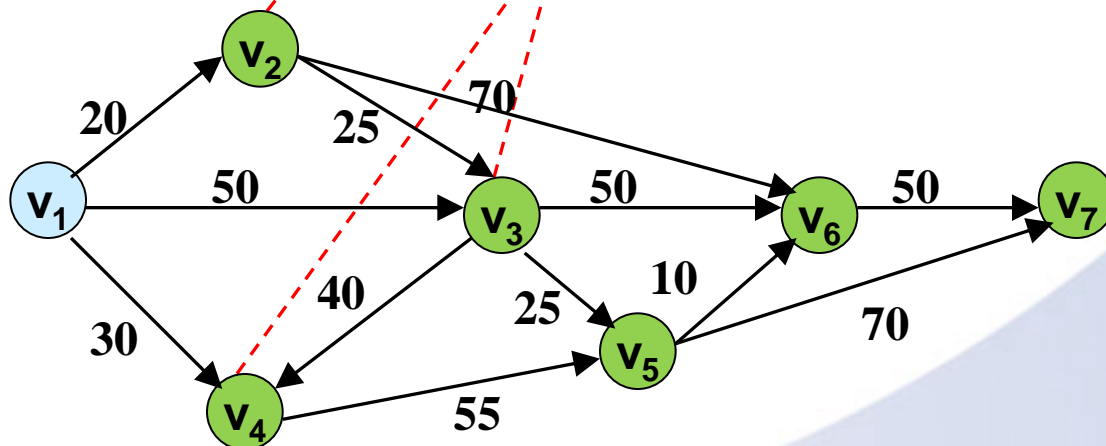
图的成本邻接矩阵:

0	20	50	30	$+\infty$	$+\infty$	$+\infty$
$+\infty$	0	25	$+\infty$	$+\infty$	70	$+\infty$
$+\infty$	$+\infty$	0	40	25	50	$+\infty$
$+\infty$	$+\infty$	$+\infty$	0	55	$+\infty$	$+\infty$
$+\infty$	$+\infty$	$+\infty$	$+\infty$	0	10	70
$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	0	50
$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	0



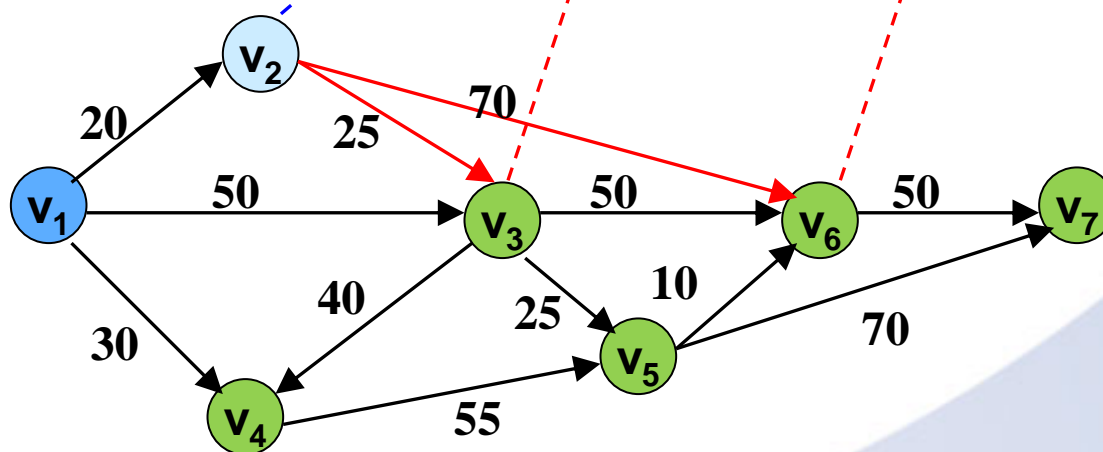
算法的执行轨迹描述

迭代	选取的 结点	S	DIST						
			(1)	(2)	(3)	(4)	(5)	(6)	(7)
置初值	—	1	0	20	50	30	$+\infty$	$+\infty$	$+\infty$



算法的执行轨迹描述

迭代	选取的 结点	S	DIST						
			(1)	(2)	(3)	(4)	(5)	(6)	(7)
置初值 1	—	1	0	20	50	30	$+\infty$	$+\infty$	$+\infty$
	2	1, 2	0	20	45	30	$+\infty$	90	$+\infty$

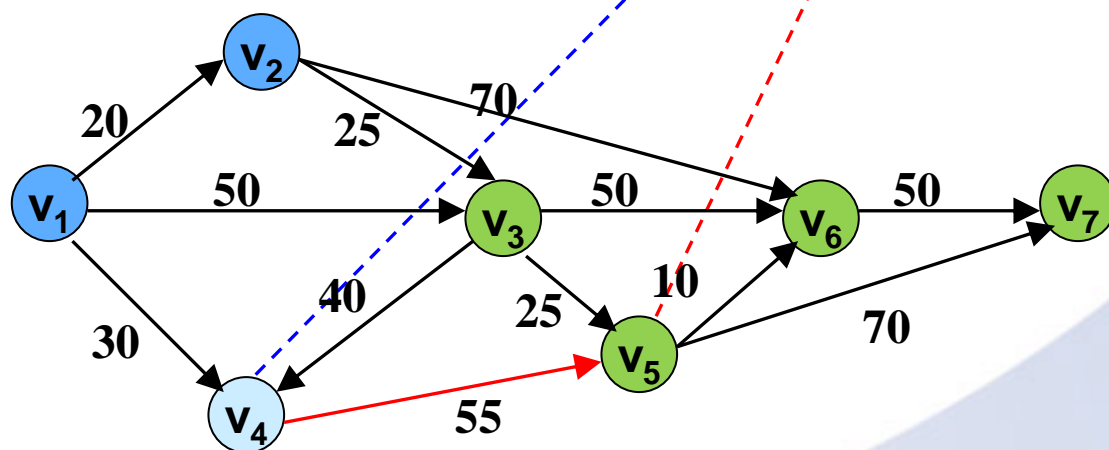


$$\begin{aligned}
 \text{DIST}(3) &= \min(\text{DIST}(3), \text{DIST}(2) + C(2, 3)) \\
 &= \min(50, 20 + 25) \\
 &= 45
 \end{aligned}$$

$$\begin{aligned}
 \text{DIST}(6) &= \min(\text{DIST}(6), \text{DIST}(2) + C(2, 6)) \\
 &= \min(+\infty, 20 + 70) \\
 &= 90
 \end{aligned}$$

算法的执行轨迹描述

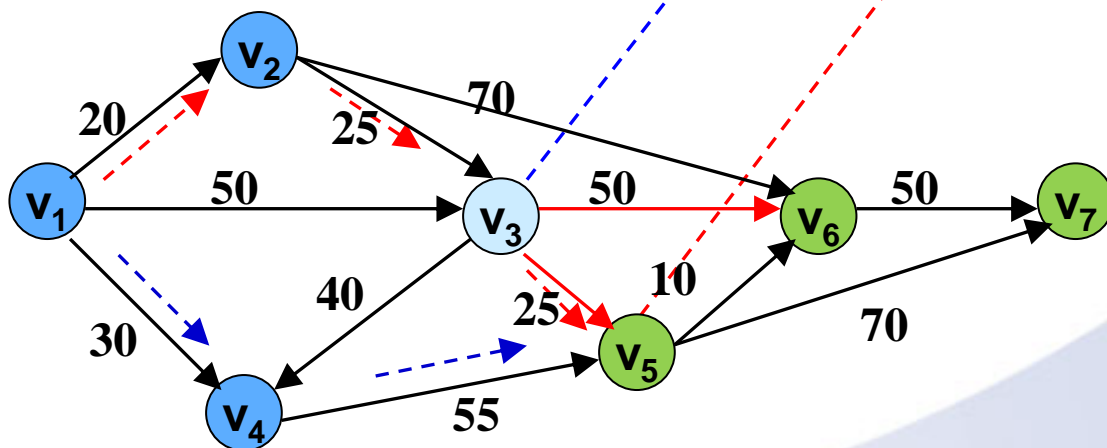
迭代	选取的 结点	S	DIST						
			(1)	(2)	(3)	(4)	(5)	(6)	(7)
置初值	—	1	0	20	50	30	$+\infty$	$+\infty$	$+\infty$
1	2	1, 2	0	20	45	30	$+\infty$	90	$+\infty$
2	4	1, 2, 4	0	20	45	30	85	90	$+\infty$



$$\begin{aligned}
 \text{DIST}(5) &= \min(\text{DIST}(5), \text{DIST}(4) + C(4, 5)) \\
 &= \min(+\infty, 30 + 55) \\
 &= 85
 \end{aligned}$$

算法的执行轨迹描述

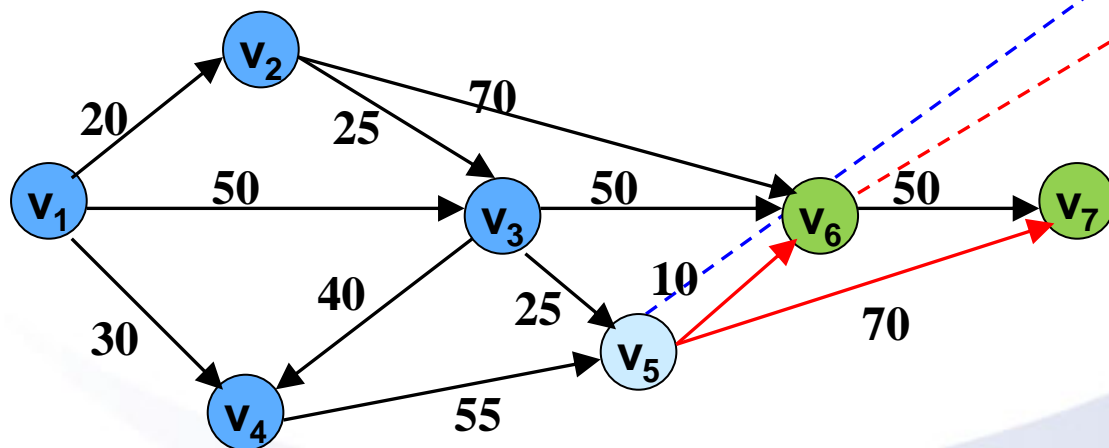
迭代	选取的 结点	S	DIST						
			(1)	(2)	(3)	(4)	(5)	(6)	(7)
置初值	—	1	0	20	50	30	$+\infty$	$+\infty$	$+\infty$
1	2	1, 2	0	20	45	30	$+\infty$	90	$+\infty$
2	4	1, 2, 4	0	20	45	30	85	90	$+\infty$
3	3	1, 2, 4, 3	0	20	45	30	70	90	$+\infty$



$$\begin{aligned}
 \text{DIST}(5) &= \min(\text{DIST}(5), \text{DIST}(3) + C(3, 5)) \\
 &= \min(85, 45 + 25) \\
 &= 70
 \end{aligned}$$

算法的执行轨迹描述

迭代	选取的 结点	S	DIST						
			(1)	(2)	(3)	(4)	(5)	(6)	(7)
置初值	—	1	0	20	50	30	$+\infty$	$+\infty$	$+\infty$
1	2	1, 2	0	20	45	30	$+\infty$	90	$+\infty$
2	4	1, 2, 4	0	20	45	30	85	90	$+\infty$
3	3	1, 2, 4, 3	0	20	45	30	70	90	$+\infty$
4	5	1, 2, 4, 3, 5	0	20	45	30	70	80	140

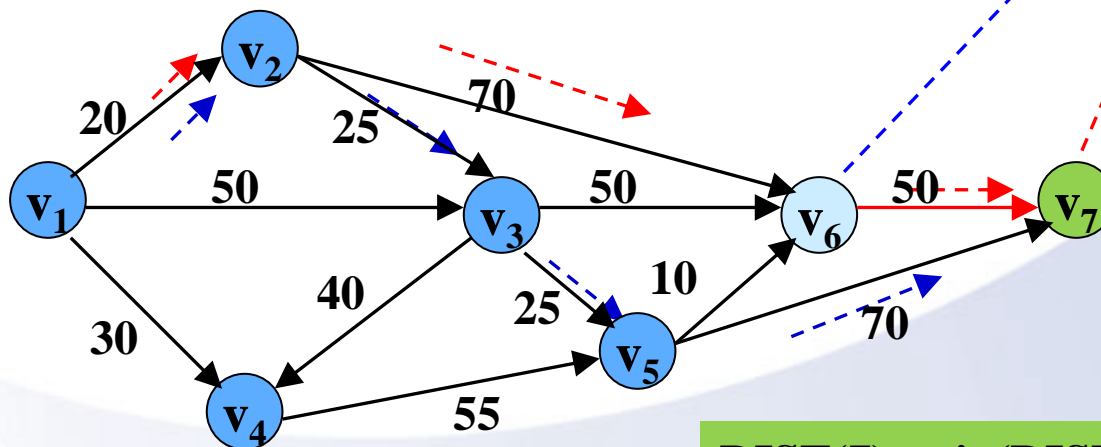


$$\begin{aligned} \text{DIST}(6) &= \min(\text{DIST}(6), \text{DIST}(5) + C(5, 6)) \\ &= \min(90, 70 + 10) = 80 \end{aligned}$$

$$\begin{aligned} \text{DIST}(7) &= \min(\text{DIST}(7), \text{DIST}(5) + C(5, 7)) \\ &= \min(+\infty, 70 + 70) = 140 \end{aligned}$$

算法的执行轨迹描述

迭代	选取的 结点	S	DIST						
			(1)	(2)	(3)	(4)	(5)	(6)	(7)
置初值	—	1	0	20	50	30	$+\infty$	$+\infty$	$+\infty$
1	2	1, 2	0	20	45	30	$+\infty$	90	$+\infty$
2	4	1, 2, 4	0	20	45	30	85	90	$+\infty$
3	3	1, 2, 4, 3	0	20	45	30	70	90	$+\infty$
4	5	1, 2, 4, 3, 5	0	20	45	30	70	80	140
5	6	1, 2, 4, 3, 5, 6	0	20	45	30	70	80	130



$$\begin{aligned}
 \text{DIST}(7) &= \min(\text{DIST}(7), \text{DIST}(6) + C(6, 7)) \\
 &= \min(140, 80 + 50) = 130
 \end{aligned}$$

算法的执行轨迹描述

迭代	选取的 结点	S	DIST						
			(1)	(2)	(3)	(4)	(5)	(6)	(7)
置初值	—	1	0	20	50	30	$+\infty$	$+\infty$	$+\infty$
1	2	1, 2	0	20	45	30	$+\infty$	90	$+\infty$
2	4	1, 2, 4	0	20	45	30	85	90	$+\infty$
3	3	1, 2, 4, 3	0	20	45	30	70	90	$+\infty$
4	5	1, 2, 4, 3, 5	0	20	45	30	70	80	140
5	6	1, 2, 4, 3, 5, 6	0	20	45	30	70	80	130

算法的执行在有 **$n-1$** 个结点加入到S中后终止，此时求出了 v_0 至其它各结点的最短路径。



4.6 单源点最短路径

■ 2. 贪心策略求解

□ 如何求出所有这些最短路径？

➤ 提示：如果 $\text{DIST}(w)$ 是通过计算

$$\text{DIST}(w) = \min(\text{DIST}(w), \text{DIST}(u) + \text{COST}(u, w))$$

且因为 $\text{DIST}(u) + \text{COST}(u, w)$ 较小而得来的，问 w 之前的那个节点应该是谁？

➤ 应该是 u

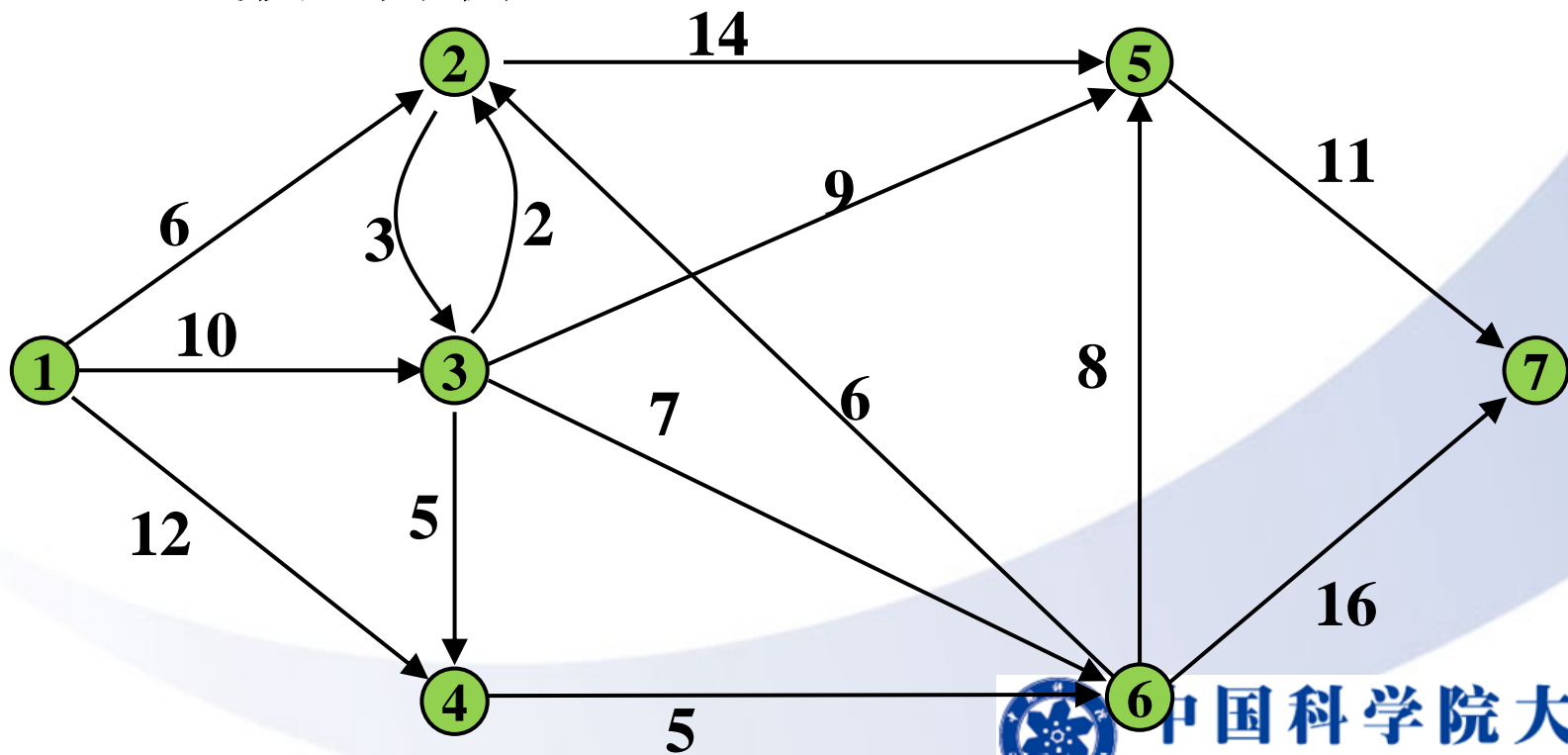
➤ u 之前又是哪个节点呢？



4.6 单源点最短路径

■ 2. 贪心策略求解

□ 下图中的权 c_{ij} 表示 v_i 到 v_j 的距离（费用、时间），从 v_1 修一条公路或架设一条高压线到 v_7 ，如何选择一条路线使距离最短。

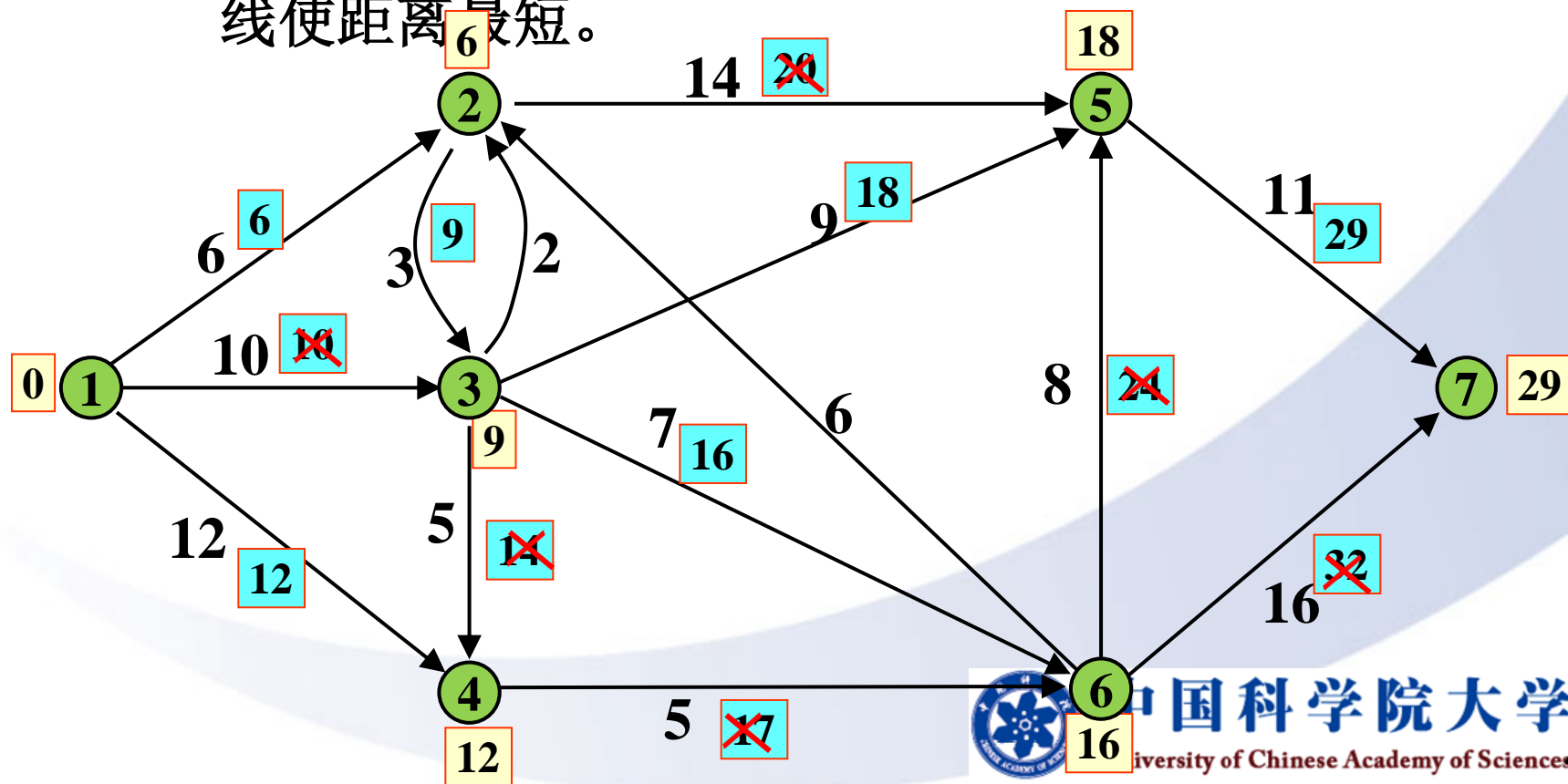


4.6 单源点最短路径

■ 2. 贪心策略求解

□ 下图中的权 c_{ij} 表示 v_i 到 v_j 的距离（费用、时间），从 v_1 修一条公路或架设一条高压线到 v_7 ，如何选择一条路线使距离最短。

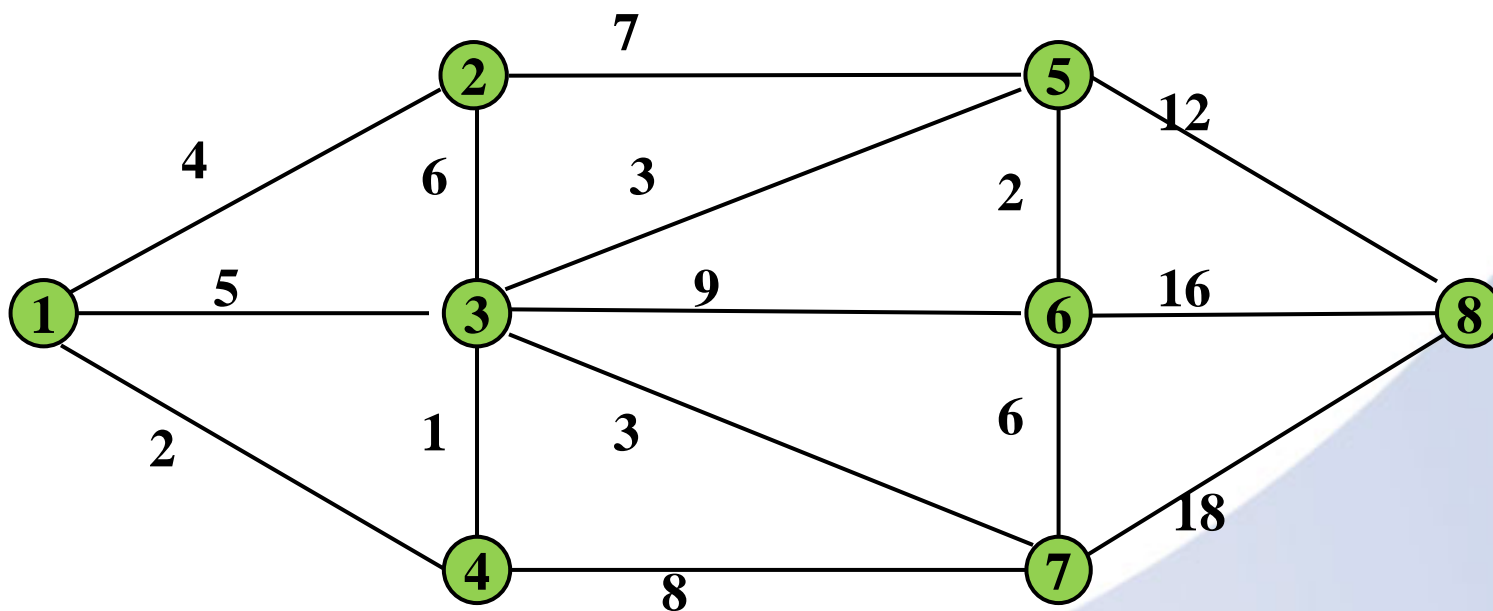
v_1 到 v_7 的最短路为：
 $p_{17} = \{v_1, v_2, v_3, v_5, v_7\}$ ，
最短路长为 $L_{17} = 29$



4.6 单源点最短路径

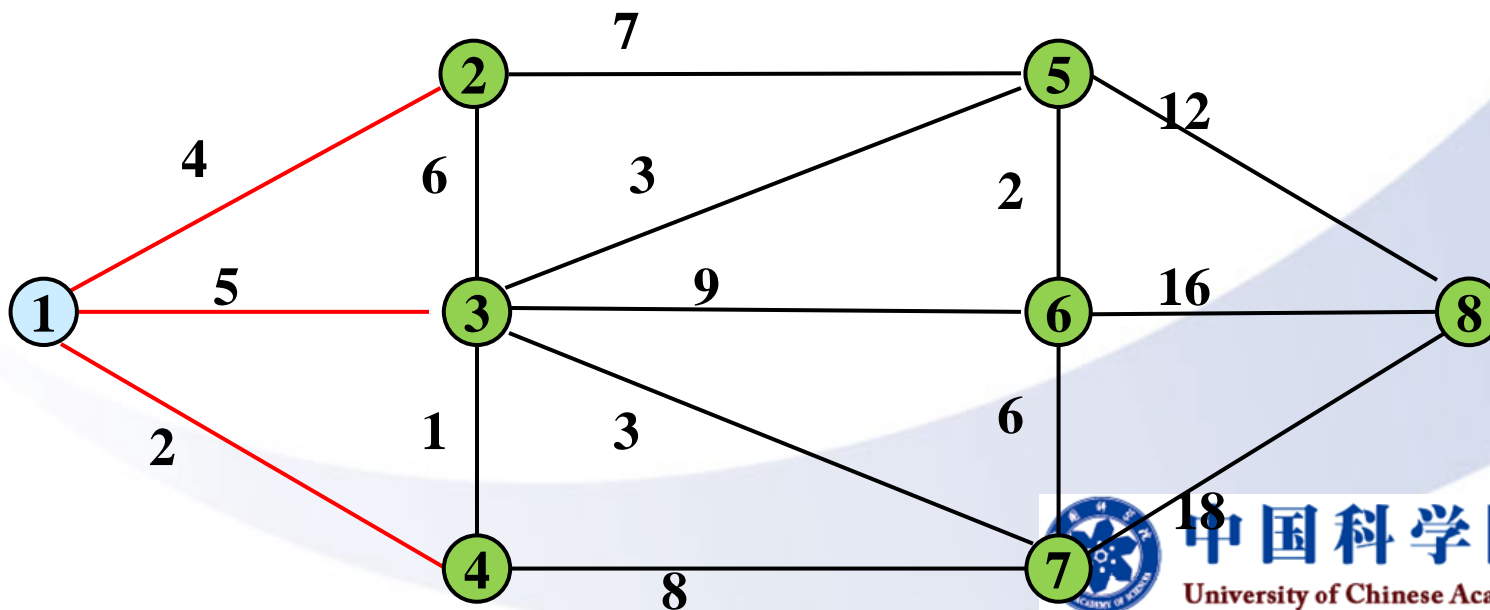
■ 2. 贪心策略求解

□ 求图所示 v_1 到各点的最短路及最短距离



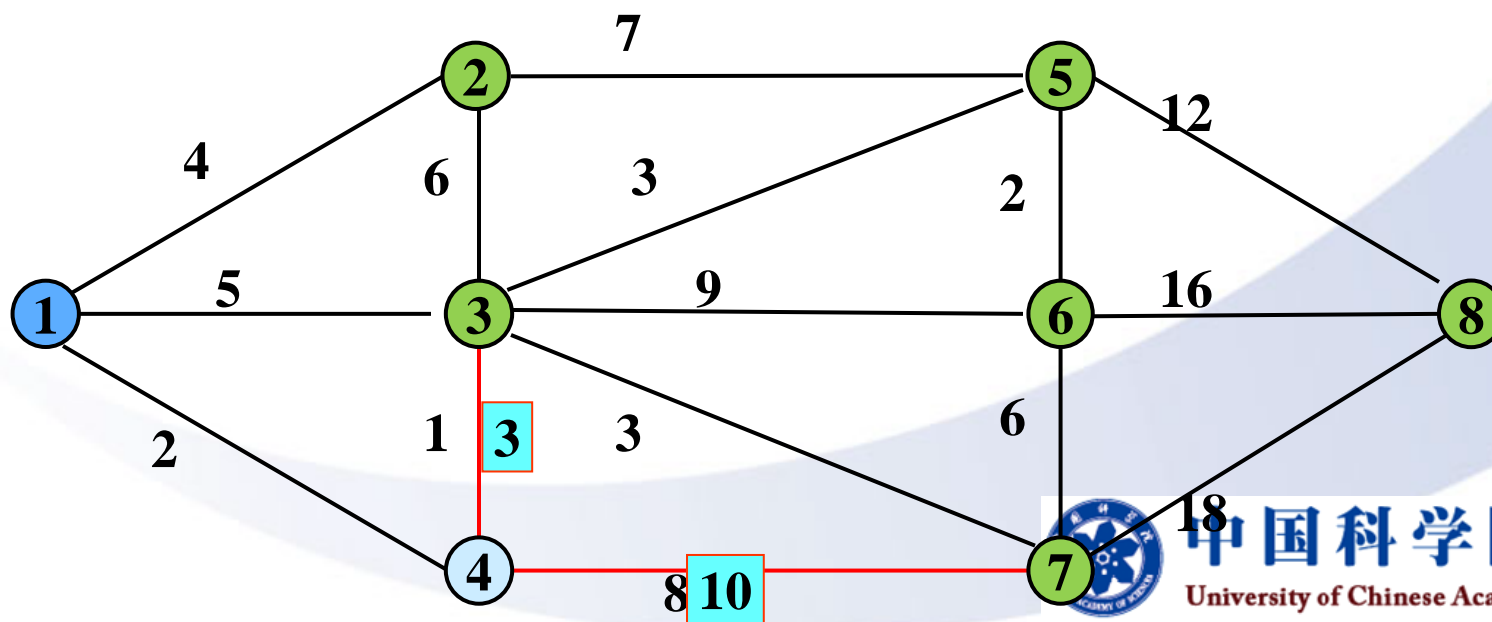
算法的执行轨迹描述

迭代	选取的 结点	S	DIST							
			(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
置初值	—	1	0	4	5	2	$+\infty$	$+\infty$	$+\infty$	$+\infty$



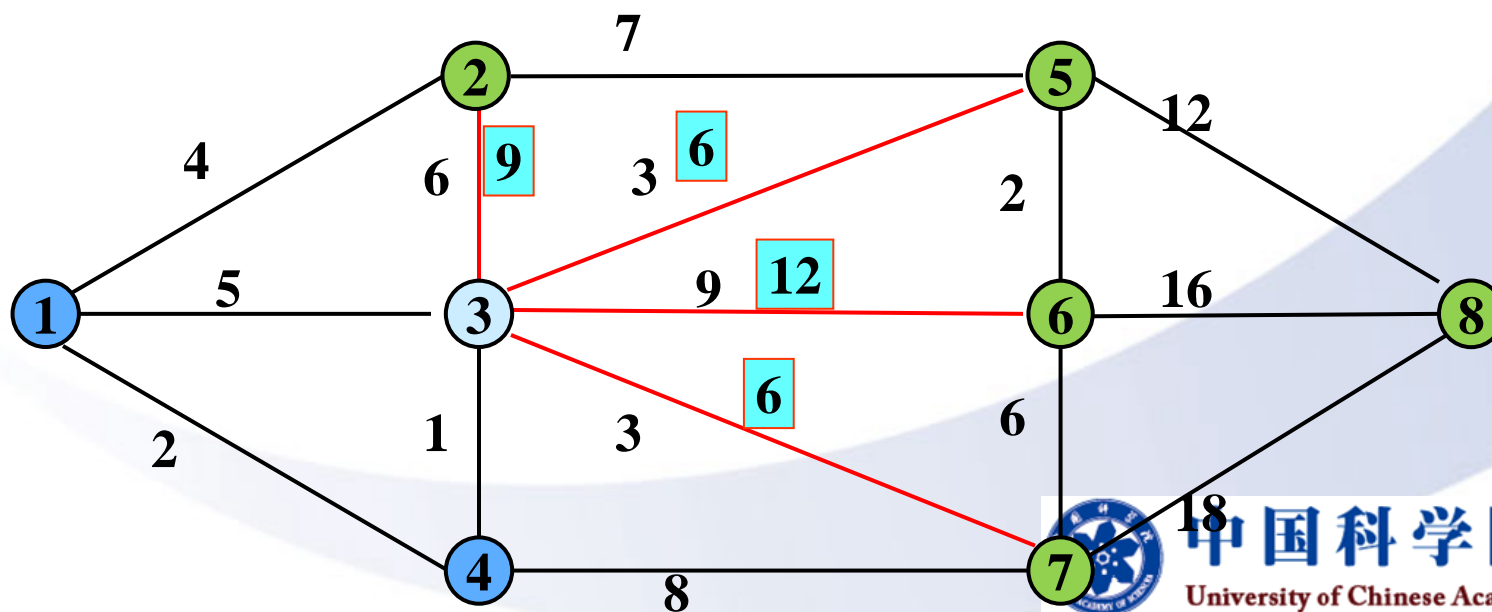
算法的执行轨迹描述

迭代	选取的 结点	S	DIST							
			(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
置初值 1	—	1	0	4	5	2	$+\infty$	$+\infty$	$+\infty$	$+\infty$
	4	1, 4	0	4	3	2	$+\infty$	$+\infty$	10	$+\infty$



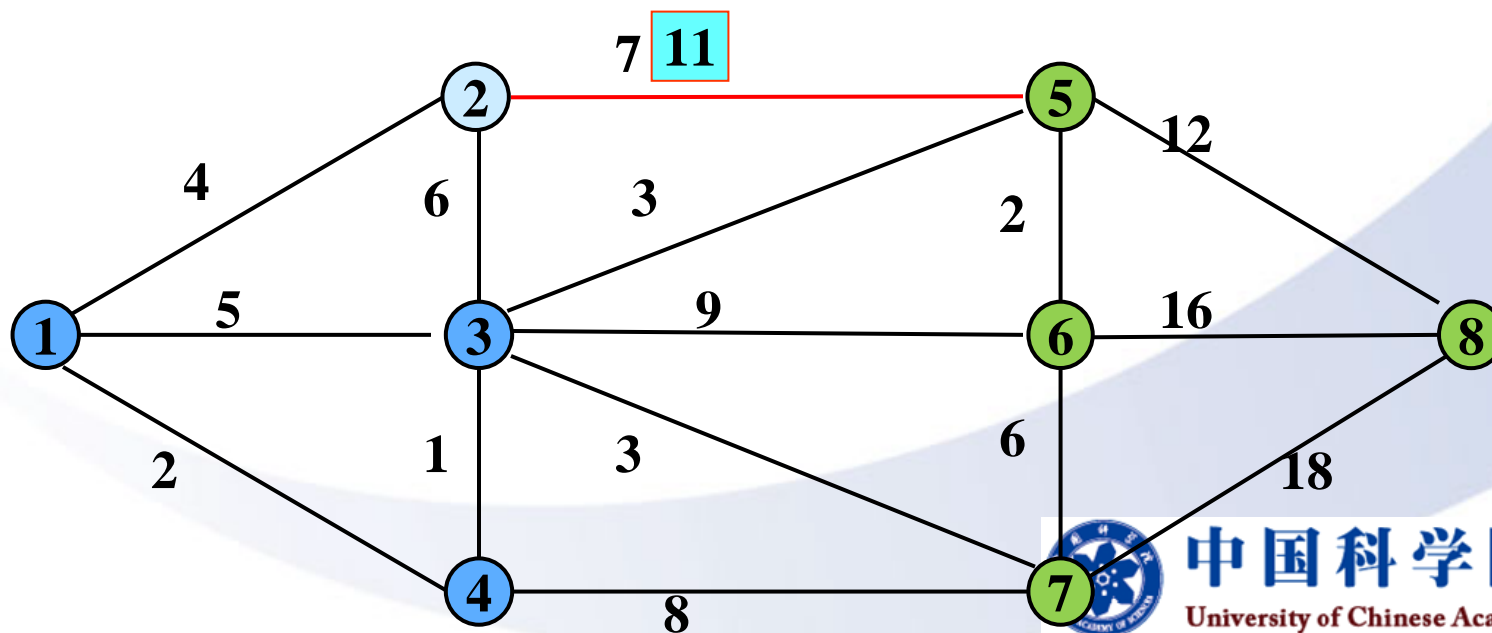
算法的执行轨迹描述

迭代	选取的 结点	S	DIST							
			(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
置初值	—	1	0	4	5	2	$+\infty$	$+\infty$	$+\infty$	$+\infty$
1	4	1, 4	0	4	3	2	$+\infty$	$+\infty$	10	$+\infty$
2	3	1, 4, 3	0	4	3	2	6	12	6	$+\infty$



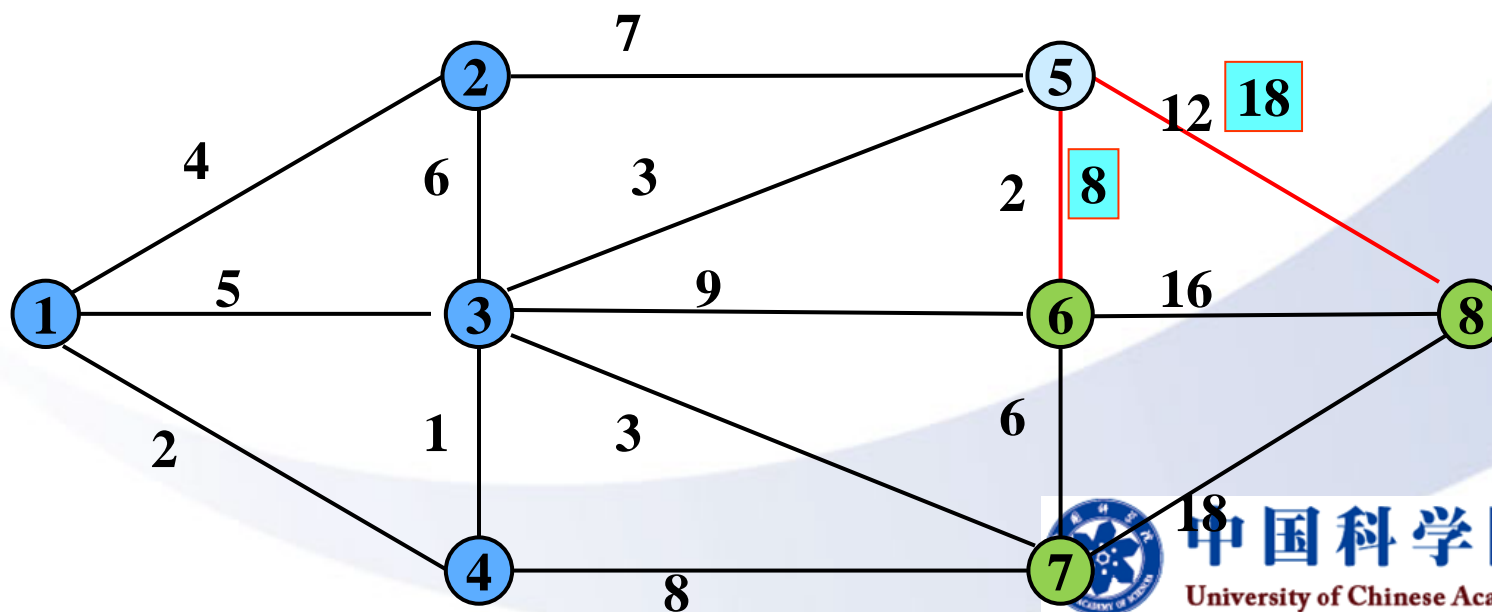
算法的执行轨迹描述

迭代	选取的 结点	S	DIST							
			(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
置初值	—	1	0	4	5	2	$+\infty$	$+\infty$	$+\infty$	$+\infty$
1	4	1, 4	0	4	3	2	$+\infty$	$+\infty$	10	$+\infty$
2	3	1, 4, 3	0	4	3	2	6	12	6	$+\infty$
		1, 4, 3, 2	0	4	3	2	6	12	6	$+\infty$



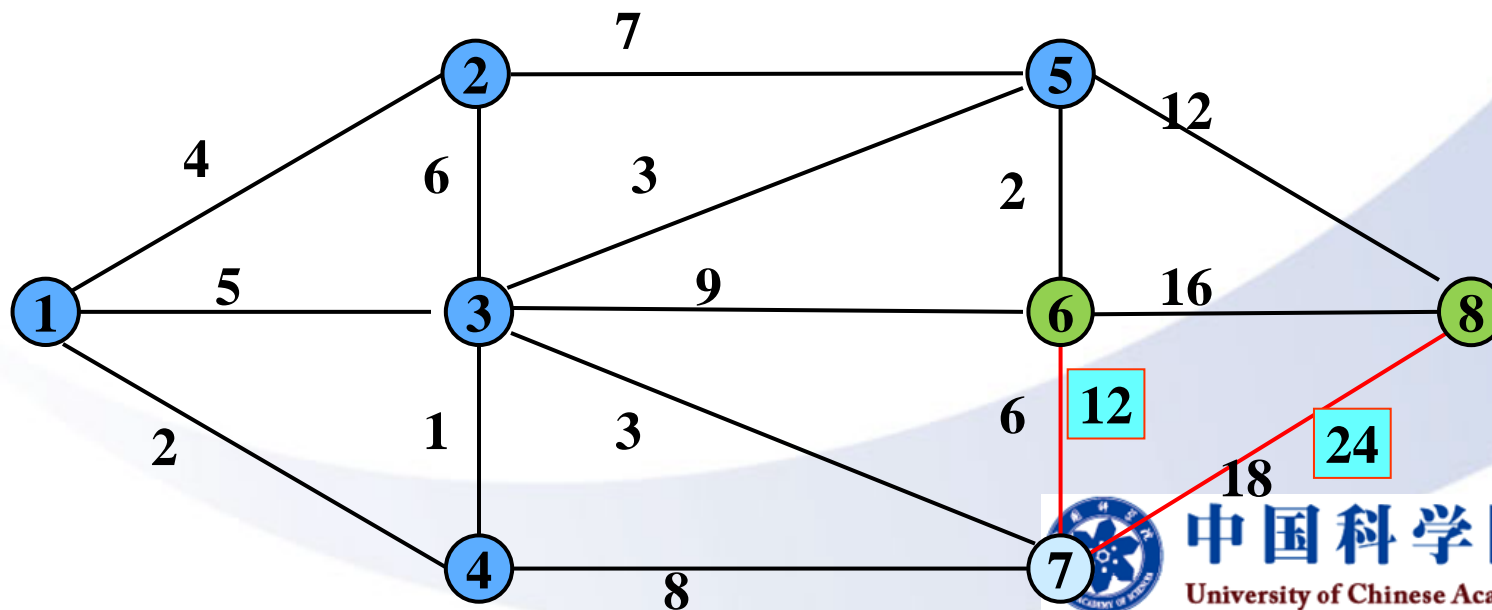
算法的执行轨迹描述

迭代	选取的 结点	S	DIST							
			(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
置初值	—	1	0	4	5	2	$+\infty$	$+\infty$	$+\infty$	$+\infty$
1	4	1, 4	0	4	3	2	$+\infty$	$+\infty$	10	$+\infty$
2	3	1, 4, 3	0	4	3	2	6	12	6	$+\infty$
3	2	1, 4, 3, 2	0	4	3	2	6	12	6	$+\infty$
4	5	1, 4, 3, 2, 5	0	4	3	2	6	8	6	18



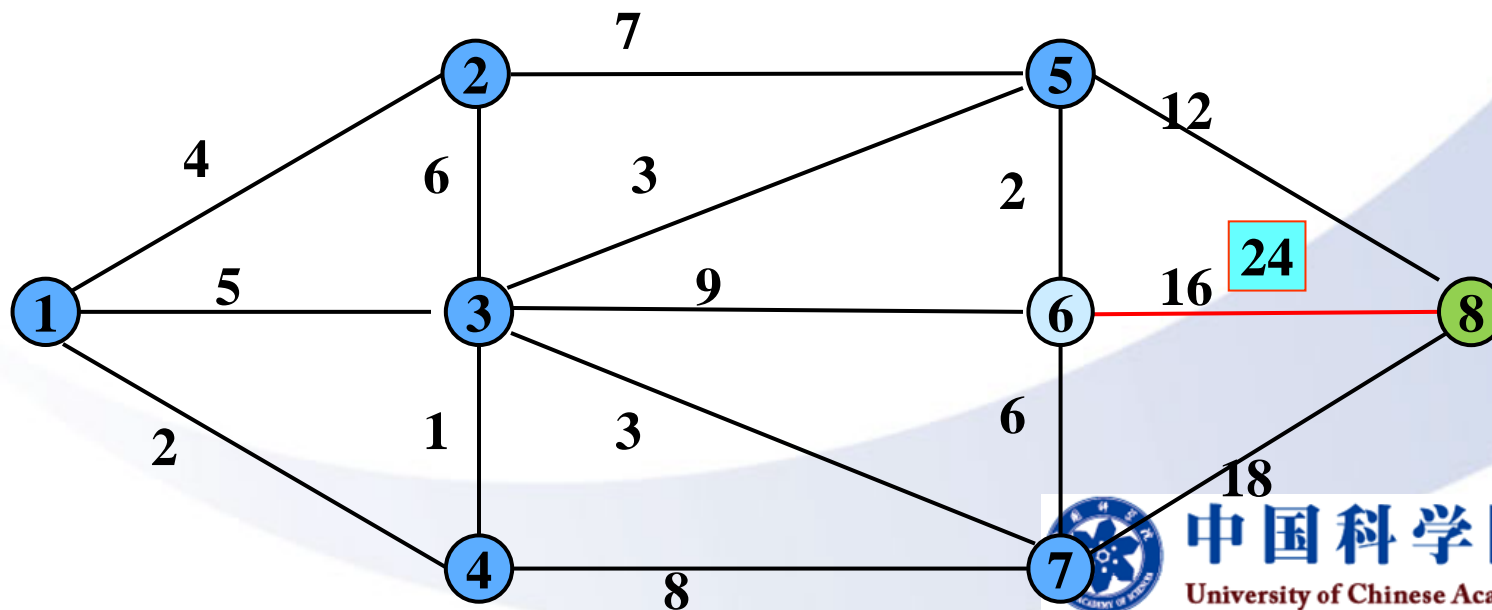
算法的执行轨迹描述

迭代	选取的 结点	S	DIST							
			(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
置初值	—	1	0	4	5	2	$+\infty$	$+\infty$	$+\infty$	$+\infty$
1	4	1, 4	0	4	3	2	$+\infty$	$+\infty$	10	$+\infty$
2	3	1, 4, 3	0	4	3	2	6	12	6	$+\infty$
3	2	1, 4, 3, 2	0	4	3	2	6	12	6	$+\infty$
4	5	1, 4, 3, 2, 5	0	4	3	2	6	8	6	18
5	7	1, 4, 3, 2, 5, 7	0	4	3	2	6	8	6	18



算法的执行轨迹描述

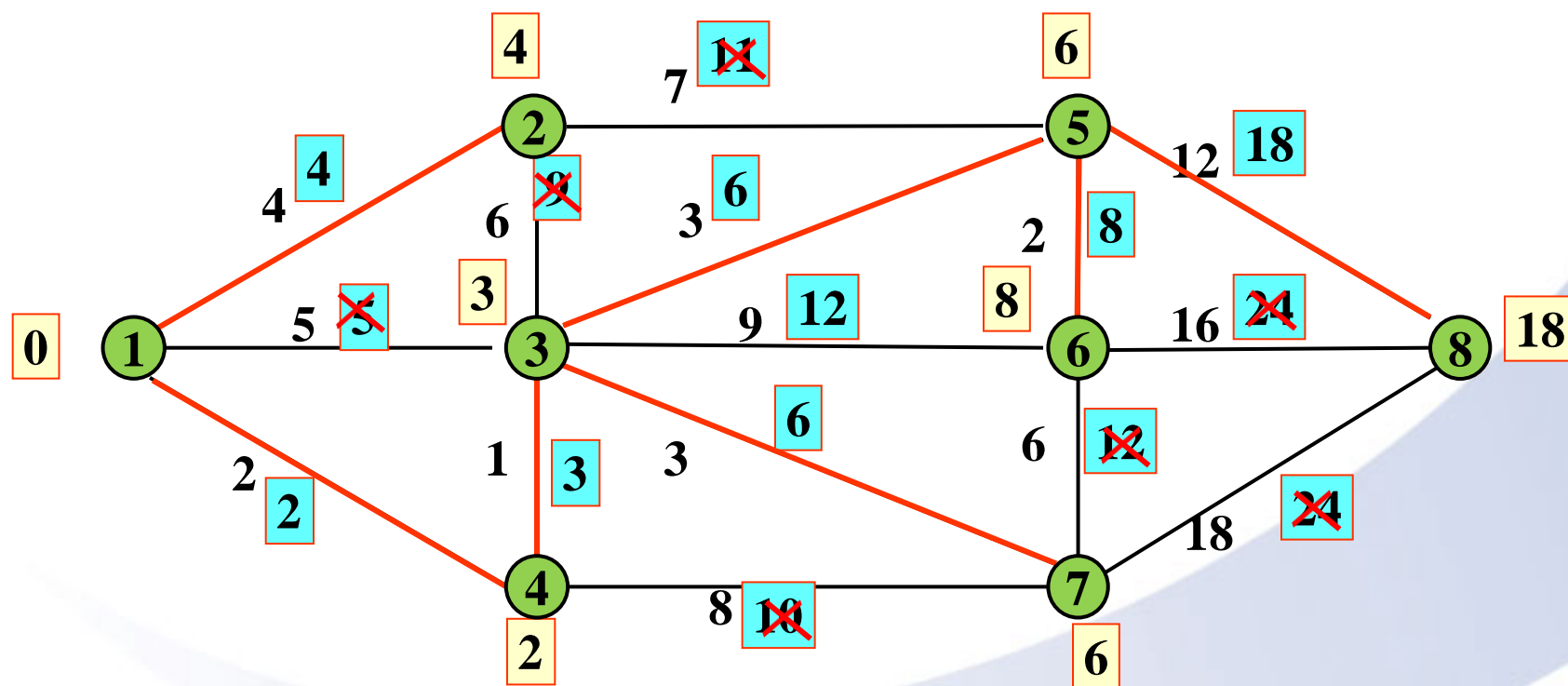
迭代	选取的 结点	S	DIST							
			(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
置初值	—	1	0	4	5	2	$+\infty$	$+\infty$	$+\infty$	$+\infty$
1	4	1, 4	0	4	3	2	$+\infty$	$+\infty$	10	$+\infty$
2	3	1, 4, 3	0	4	3	2	6	12	6	$+\infty$
3	2	1, 4, 3, 2	0	4	3	2	6	12	6	$+\infty$
4	5	1, 4, 3, 2, 5	0	4	3	2	6	8	6	18
5	7	1, 4, 3, 2, 5, 7	0	4	3	2	6	8	6	18
6	6	1, 4, 3, 2, 5, 7, 6	0	4	3	2	6	8	6	18



4.6 单源点最短路径

■ 2. 贪心策略求解

□ 求图所示 v_1 到各点的最短路及最短距离

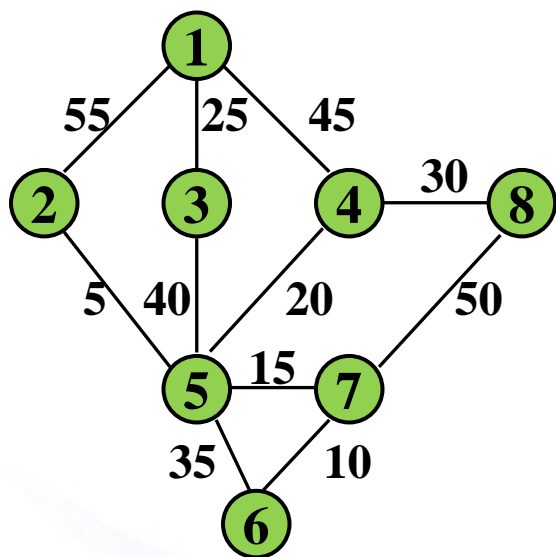


4.6 单源点最短路径

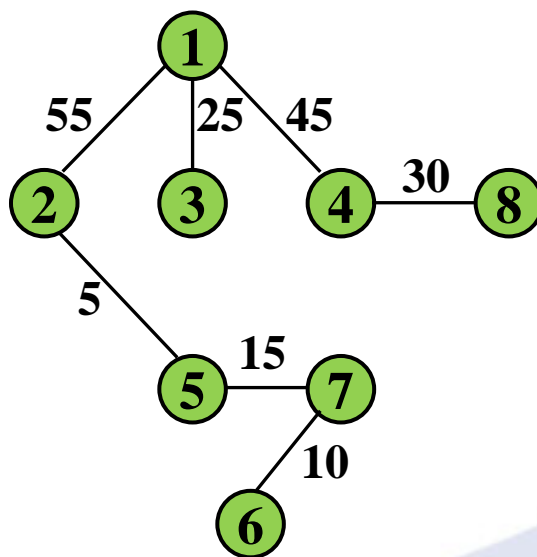
■ 3. 最短路径生成树

□ 对于无向连通图 G ，由结点 v 到其余各结点的最短路径的边构成 G 的一棵**生成树**，称为**最短路径生成树**。

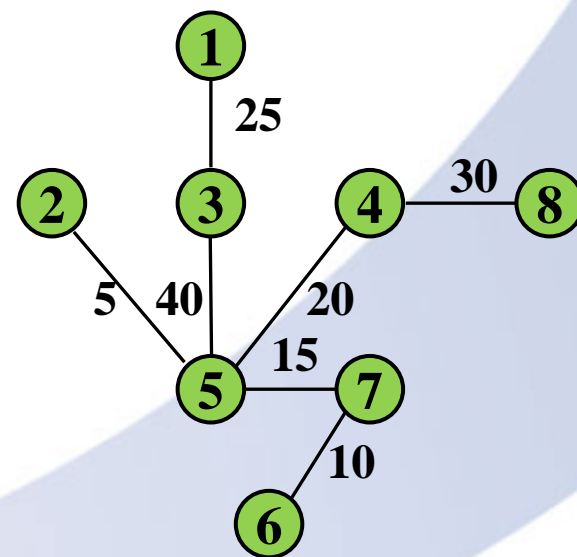
□ 注：不同起点 v 的生成树可能不同。



原始图



由结点1出发的最短路径生成树



最小成本生成树



4.6 单源点最短路径

■ 4. 贪心策略的基本要素

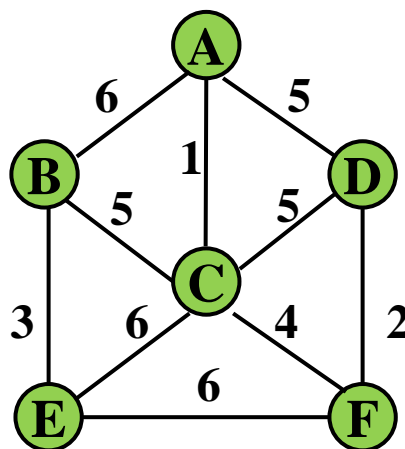
- 贪心法总是作出在**当前看来最好的选择**。也就是说贪心法**并不从整体最优考虑**，它所作出的选择只是在某种意义上的**局部最优**选择。
- 它期望通过所作的局部最优选择产生出一个全局最优解。
- 虽然贪心法不能对所有问题都得到整体最优解，但对许多问题它能产生整体最优解。如单源最短路径问题，最小生成树问题等。
- 而在一些情况下，即使贪心算法不能得到整体最优解，其最终结果却是最优解的很好近似。



作业-课后练习13

■ 问题描述

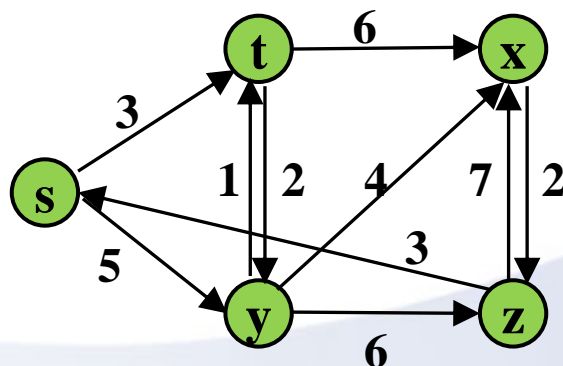
- 利用Prim算法，求下面无向图的最小生成树。
- 利用Kruskal算法，求下面无向图的最小生成树。
- 利用破圈法，求下面无向图的最小生成树。



作业-课后练习14

■ 问题描述

- 在下面的有向图中，利用算法**SHORTEST-PATHS**获取按照长度非降次序排列的由结点到其余结点的最短路径长度。
- 将结点s作为源点，计算其到其余结点的最短路径长度。
- 将结点z作为源点，计算其到其余结点的最短路径长度。



作业-算法实现3

■ 删数问题

□通过键盘输入一个高精度的正整数 n (n 的有效位数 ≤ 240)，去掉其中任意 s 个数字后，剩下的数字按原左右次序将组成一个新的正整数。编程对给定的 n 和 s ，寻找一种方案，使得剩下的数字组成的新数最小。

□输入： n, s

□输出：最后剩下的最小数

□输入示例

178543

4

□输出示例

13



作业-算法实现3

■ 删数问题

□ 要求

- 给出算法的说明性文档
- 用C语言(C++, Matlab)编写该算法的程序
- 上载到课程网站上



End

