《计算机算法设计与分析》

第四章 贪心方法

马丙鹏

2020年10月13日

中国科学院大学
University of Chinese Academy of Sciences

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。

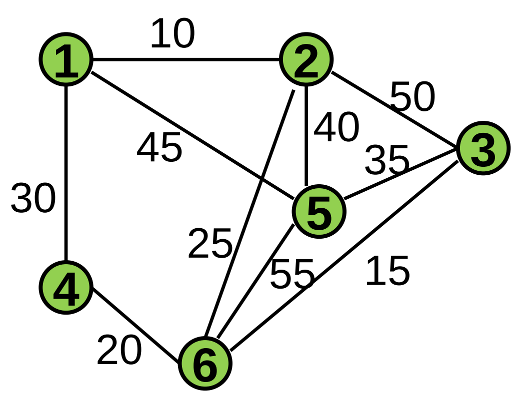
4.5 最小生成树

- 2. 贪心策略
  - 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**

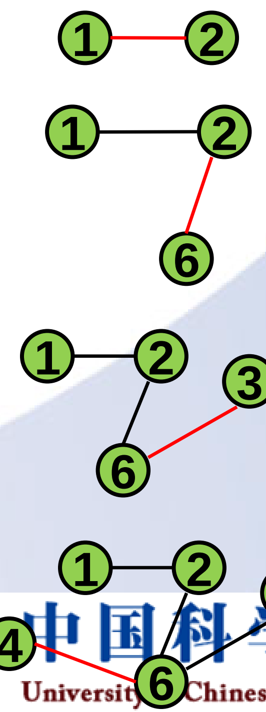
4.5 最小生成树

■ 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∪{(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

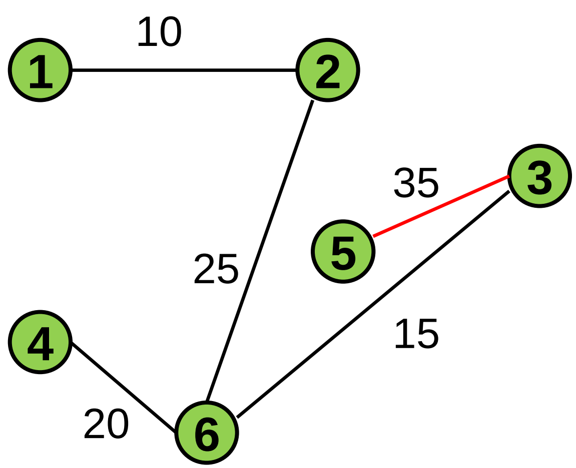


4.5 最小生成树

■ 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∪{(u, v)}也是一棵树的最小成本边。

边	成本
(3, 5)	35



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

算法4.7 Prim最小生成树算法

procedure PRIM(E, COST, n, T, mincost)

//E是G的边集。COST(n, n)是n结点图G的成本邻接矩阵，矩阵元素COST(i, j)是一个正实数，如果不存在边(i, j)，则为+∞。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//

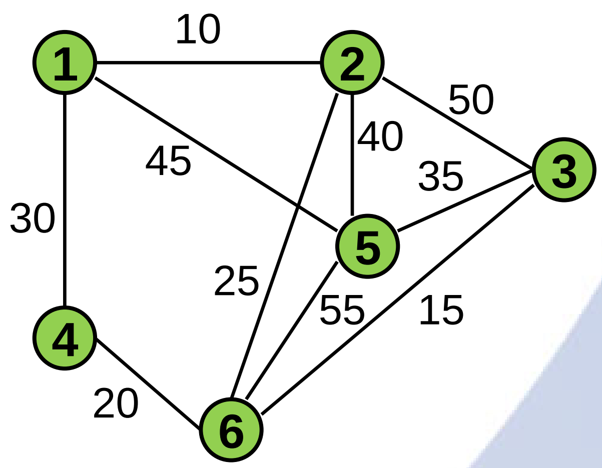
```
real COST(n, n), mincost
integer NEAR(n), n, i, k, l, T(1:n-1, 2)
(k, l)←具有最小成本的边
mincost←COST(k, l)
(T(1, 1), T(1, 2)) ←(k, l)
 for i←1 to n do //将NEAR置初值//
    if COST(i, l) <COST(i, k) then NEAR(i)←l
          else  NEAR(i) ←k
    endif
  repeat
  NEAR(k)←NEAR(l)←0
```

```
for i←2 to n-1 do //找T的其余n-2条边//
    设j是NEAR(j)≠0 且COST(j, NEAR(j))最小的下标
    (T(i, 1), T(i, 2))←(j, NEAR(j))
    mincost←mincost + COST(j, NEAR(j))
    NEAR(j)←0
    for k←1 to n do //修改NEAR//
        if NEAR(k)≠0 and COST(k, NEAR(k))>COST(k, j)
            then NEAR(k)←j
        endif
    repeat
repeat
if mincost>∞ then print('no spanning tree') endif
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

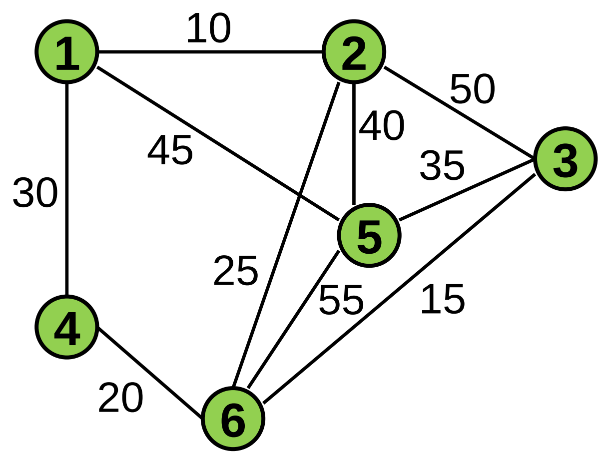
- 3. Prim算法
  - 另一种PRIM算法
    - 最小生成树中包含了与每个结点v相关的一条最小成本边。

      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

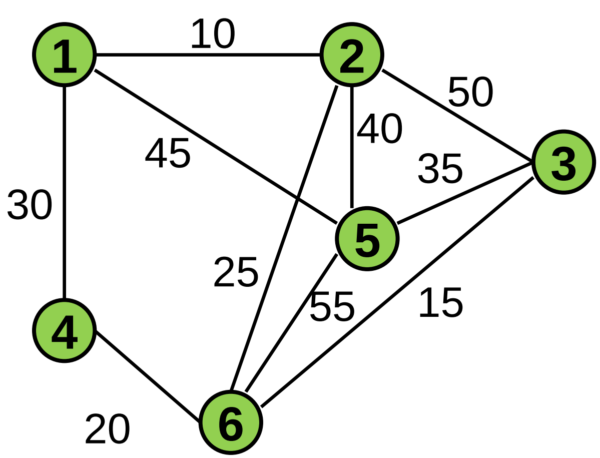
- (连通)图的边按成本的非降次序排列，下一条计入生成树T中的边是还没有计入的边中具有最小成本、且和T中现有的边不会构成环路的边。



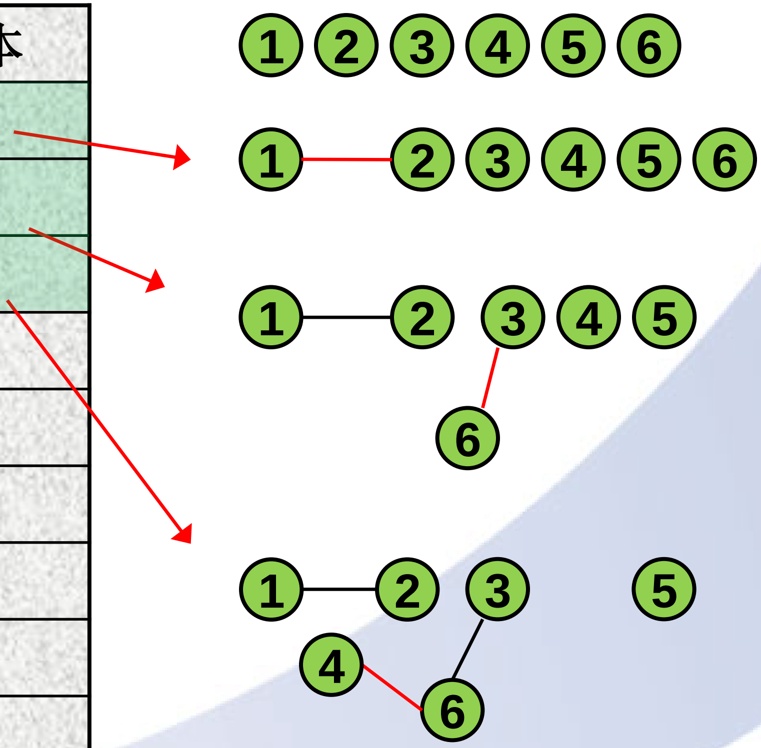
边	成本
(1, 2)	10
(3, 6)	15
(4, 6)	20
(2, 6)	25
(1, 4)	30
(3, 5)	35
(2, 5)	40
(1, 5)	45
(2, 3)	50
(5, 6)	55

4.5 最小生成树

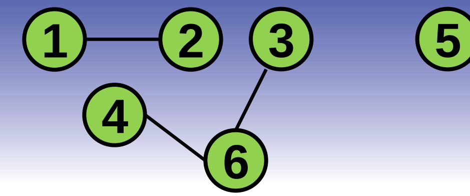
■ **4. Kruskal算法**



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

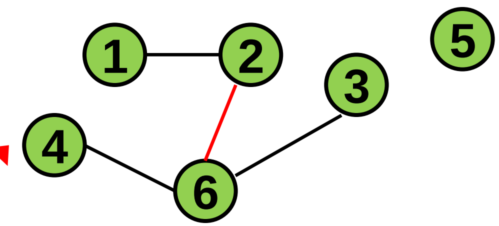


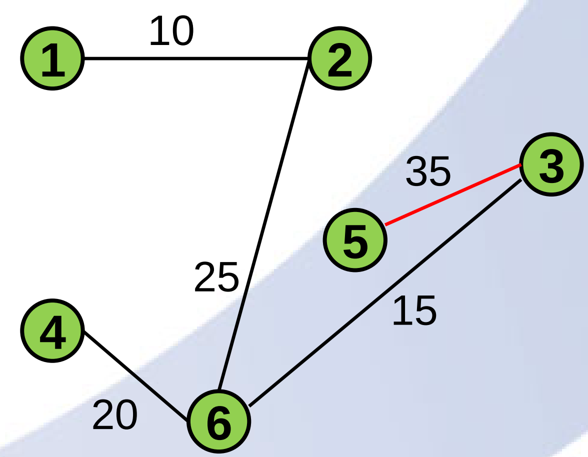
4.5 最小生成树



■ **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





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

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

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

算法4.9 Kruskal算法

```
procedure KRUSKAL(E, COST, n, T, mincost)
//G有n个结点，E是G的边集。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//
   real mincost, 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← -1 //每个结点都在不同的集合中//
   i←mincost←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←FIND(u); k←FIND(v)
       if(j≠k) then i←i+1
           T(i, 1) ←u;  T(i, 2) ←v
           mincost←mincost + COST(u, v)
           call UNION(j, k)
       endif
   repeat
    if i≠n-1 then print(‘no spanning tree’) endif
    return
 end KRUSKAL
```

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(loge)的时间内找到；
    - 算法的计算时间是O(eloge)。

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

中国科学院大学
University of Chinese Academy of Sciences

4.6 单源点最短路径

■ 1. 问题描述

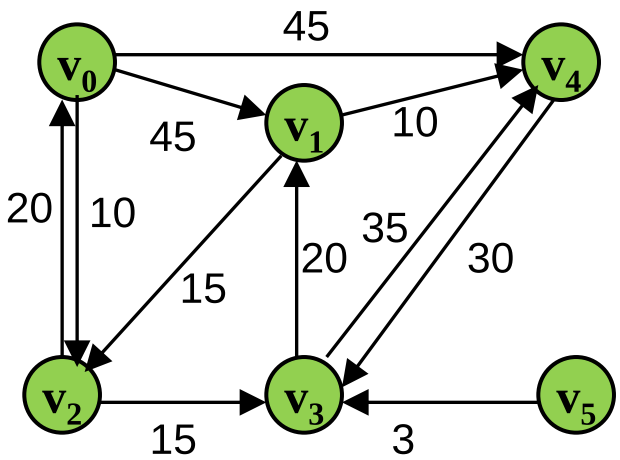
- 最短路径问题
  - 单源点最短路径问题
  - 每对结点之间的路径问题
  - 特定线路下的最短路径问题等
- 单源点最短路径问题
  - 已知一个n结点有向图G=(V, E)和边的权函数c(e)，求由G中某指定结点v_0到其它各结点的最短路径。
  - 路径长度：路径上所有边的权值之和。
  - 最短路径：具有最小长度的路径。
  - 假定边的权值为正。

中国科学院大学
University of Chinese Academy of Sciences

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

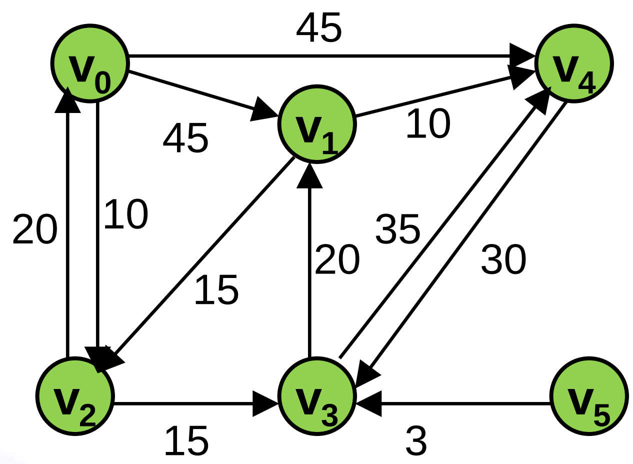
- 度量标准
  - ➢度量标准的选择：
    - ✓逐条构造最短路径，可以使用迄今已生成的所有路径长度之和作为度量，
    - ✓——为使之达到最小，其中任意一条路径都应具有最小长度。
  - ➢假定已经构造了i条最短路径，则下一条要构造的路径应是下一条最短的路径。
  - ➢处理规则：
    - ✓按照路径长度的非降次序依次生成从结点v_0到其它各结点的最短路径。

4.6 单源点最短路径

■ 2. 贪心策略求解

- 度量标准
  - 例

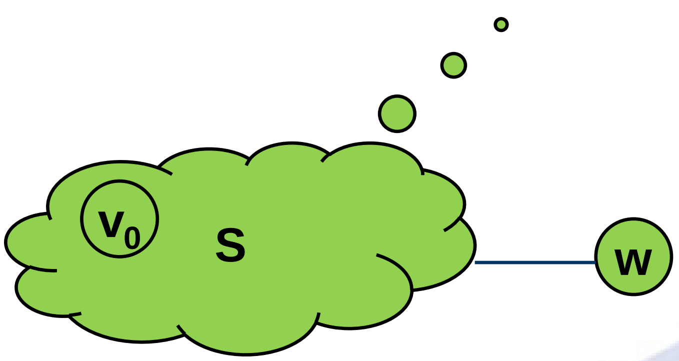
问题：如何对尚未生成的路径长度进行排序，以确定其中最短者？



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：　$v_0, \underbrace{s_1, s_2, \ldots, s_{m-1}}_{\text{均在S中}}, u$

    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。

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

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。

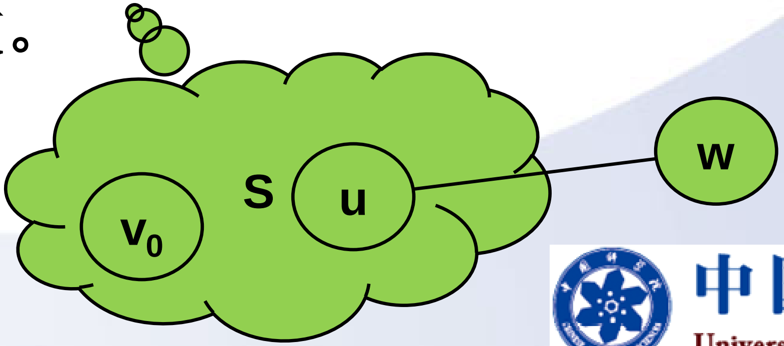
4.6 单源点最短路径

■ 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——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<u, w>∈E和<u, w> ∉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

      $$DIST(w) = DIST(u) + c(u, w)$$

    - 随着u的加入，DIST(w)调整为

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

算法4.10 生成最短路径的贪心算法

```
procedure SHORTEST-PATHS(v, COST, DIST, n)
  //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≤j≤n。DIST(v)被置成零//
  boolean S(1:n); real COST(1:n, 1:n), DIST(1:n)
  integer u, v, n, num, i, w
  for i←1 to n do //将集合S初始化为空//
      S(i) ←0; DIST(i) ←COST(v, i)
  repeat
  S(v) ←1; DIST(v) ←0  //结点v计入S//
  for num←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{DIST(w)}   (min over S(w)=0)
      S(u) ←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
```

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(n²)

(1) **for** i←1 **to** n **do** → $\Theta(n)$

S(i) ←0; DIST(i) ←COST(v,i)

repeat

(2) **for** num←2 **to** n-1 **do** → $O(n-2)$

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

S(u) ←1

for 所有S(w) = 0的结点w **do** → $O(n)$

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

repeat

repeat

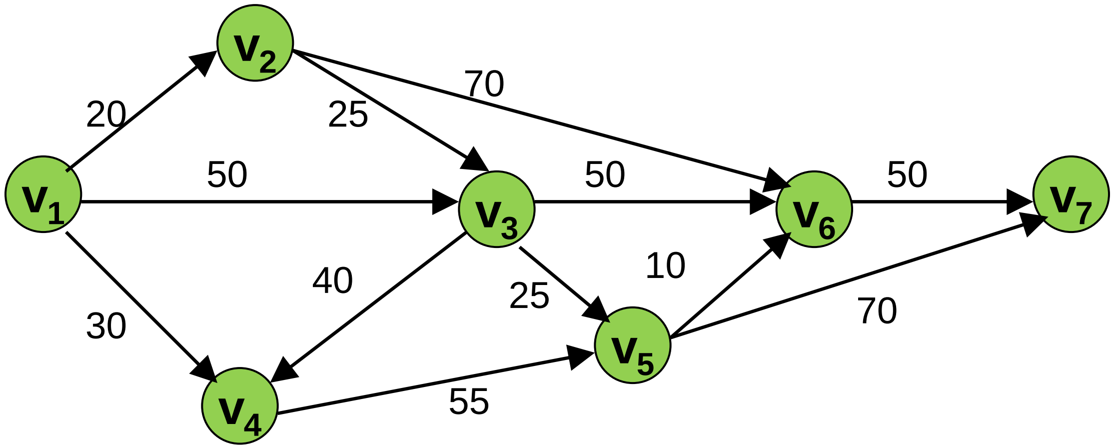
4.6 单源点最短路径

■ 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。

中国科学院大学
University of Chinese Academy of Sciences

例4.11　求下图中从v_1出发到其余各结点的最短路径

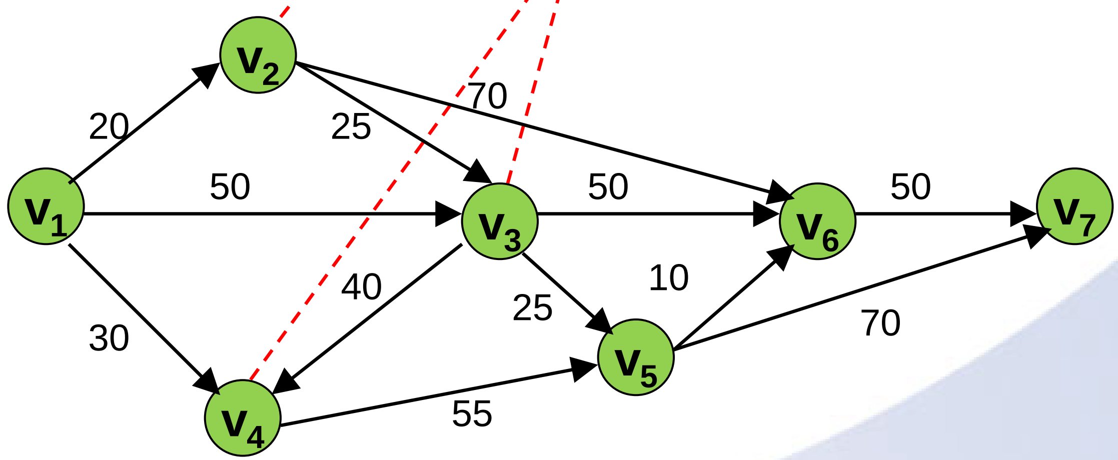


图的成本邻接矩阵：

$$\begin{pmatrix} 0 & 20 & 50 & 30 & +\infty & +\infty & +\infty \\ +\infty & 0 & 25 & +\infty & +\infty & 70 & +\infty \\ +\infty & +\infty & 0 & 40 & 25 & 50 & +\infty \\ +\infty & +\infty & +\infty & 0 & 55 & +\infty & +\infty \\ +\infty & +\infty & +\infty & +\infty & 0 & 10 & 70 \\ +\infty & +\infty & +\infty & +\infty & +\infty & 0 & 50 \\ +\infty & +\infty & +\infty & +\infty & +\infty & +\infty & 0 \end{pmatrix}$$

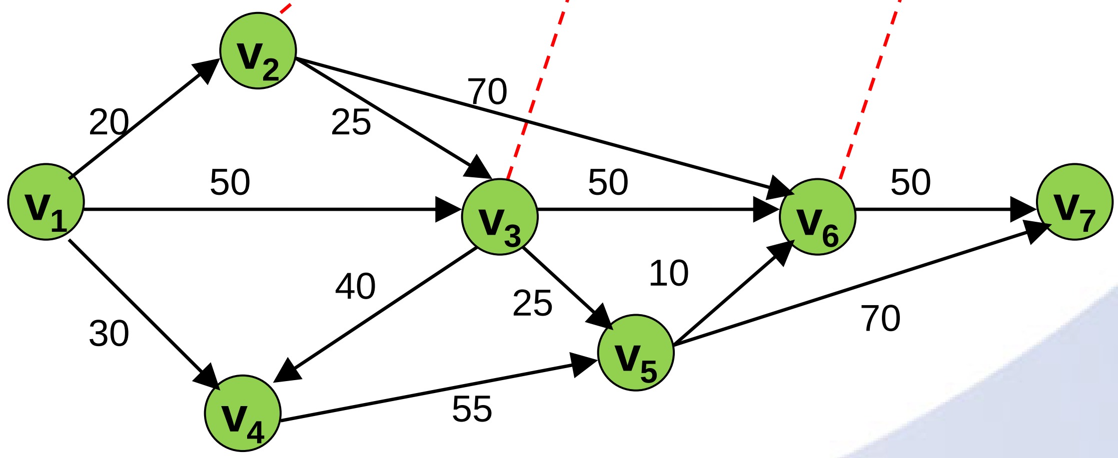
算法的执行轨迹描述

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



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	+∞	+∞	+∞
1	2	1, 2	0	20	45	30	+∞	90	+∞



DIST(3)=min(DIST(3), DIST(2)+C(2, 3))

=min(50, 20+25)

=45

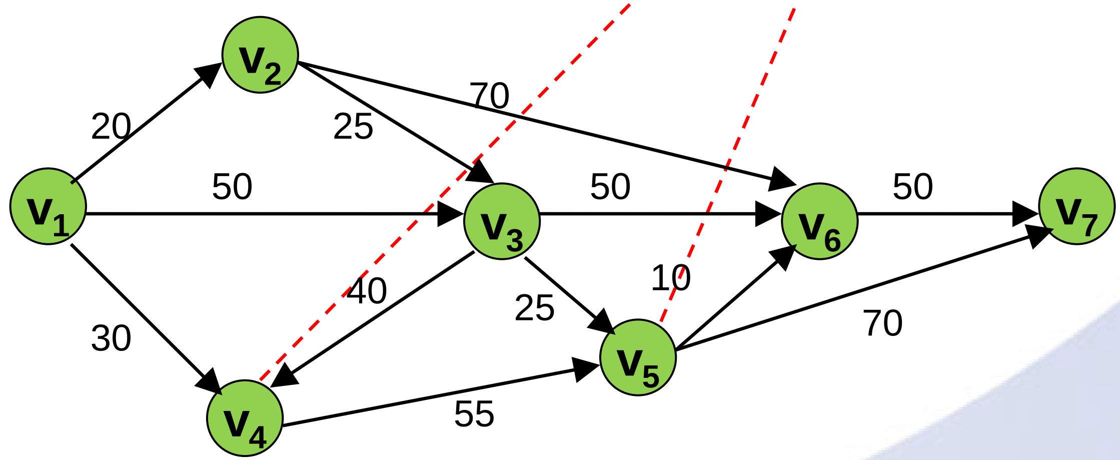
DIST(6)=min(DIST(6), DIST(2)+C(2, 6))

=min(+∞, 20+70)

=90

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	+∞	+∞	+∞
1	2	1, 2	0	20	45	30	+∞	90	+∞
2	4	1, 2, 4	0	20	45	30	85	90	+∞



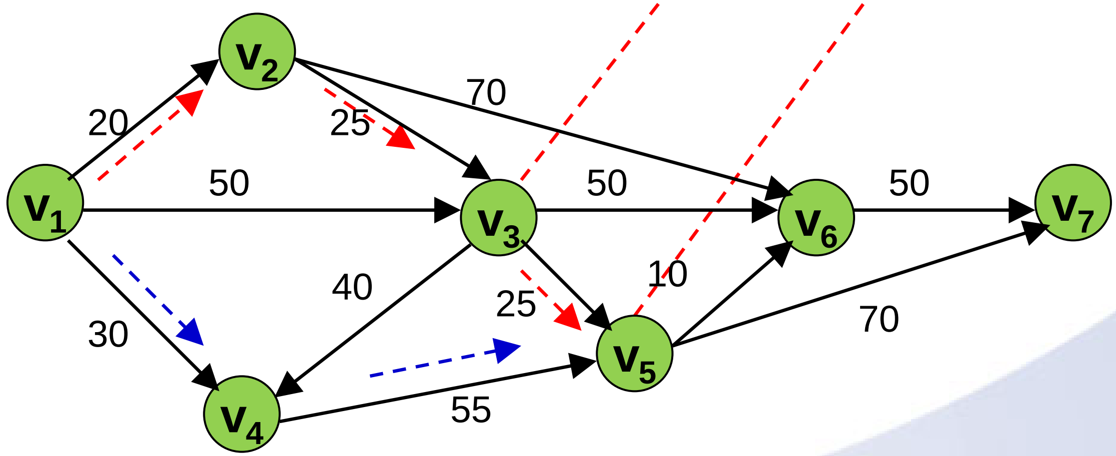
DIST(5)=min(DIST(5), DIST(4)+C(4,5))

=min(+∞, 30+55)

=85

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	+∞	+∞	+∞
1	2	1, 2	0	20	45	30	+∞	90	+∞
2	4	1, 2, 4	0	20	45	30	85	90	+∞
3	3	1, 2, 4, 3	0	20	45	30	70	90	+∞



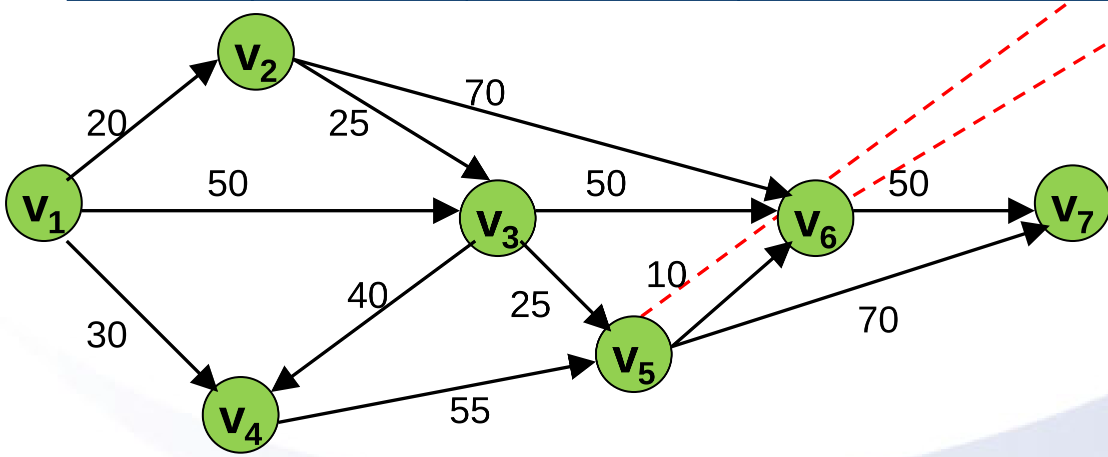
DIST(5)=min(DIST(5), DIST(3)+C(3, 5))

=min(85, 45+25)

=70

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	+∞	+∞	+∞
1	2	1, 2	0	20	45	30	+∞	90	+∞
2	4	1, 2, 4	0	20	45	30	85	90	+∞
3	3	1, 2, 4, 3	0	20	45	30	70	90	+∞
4	5	1, 2, 4, 3, 5	0	20	45	30	70	80	140

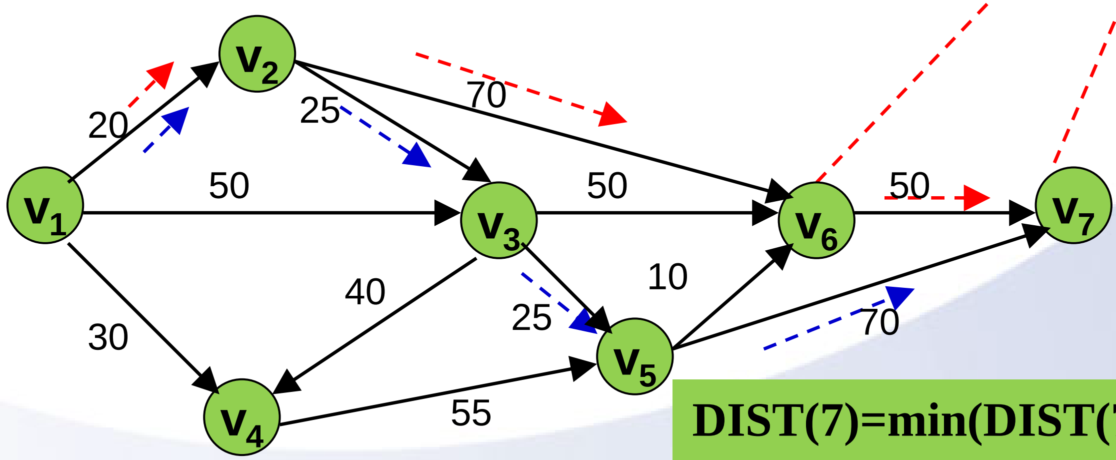


DIST(6)=min(DIST(6), DIST(5)+C(5, 6))
=min(90, 70+10) =80

DIST(7)=min(DIST(7), DIST(5)+C(5, 7))
=min(+∞, 70+70) =140

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	+∞	+∞	+∞
1	2	1, 2	0	20	45	30	+∞	90	+∞
2	4	1, 2, 4	0	20	45	30	85	90	+∞
3	3	1, 2, 4, 3	0	20	45	30	70	90	+∞
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



DIST(7)=min(DIST(7), DIST(6)+C(6, 7))

=min(140, 80+50)

=130

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	+∞	+∞	+∞
1	2	1, 2	0	20	45	30	+∞	90	+∞
2	4	1, 2, 4	0	20	45	30	85	90	+∞
3	3	1, 2, 4, 3	0	20	45	30	70	90	+∞
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。

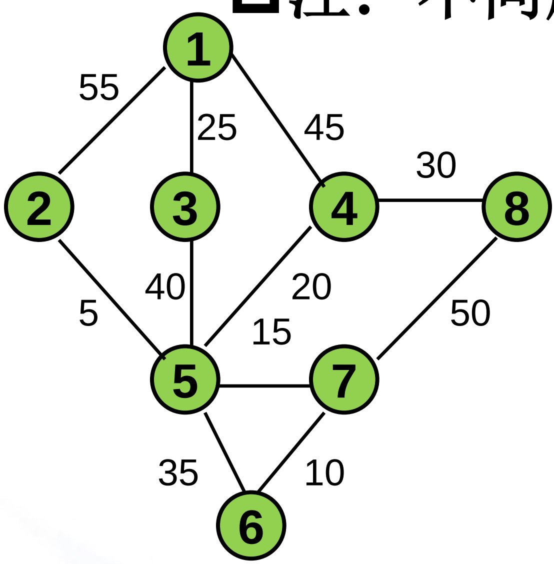
4.6 单源点最短路径

- **2. 贪心策略求解**
  - 如何求出所有这些最短路径？
    - 提示：如果DIST(w)是通过计算
      DIST(w) = min(DIST(w), DIST(u) + COST(u, w))
      且因为DIST(u) + 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？

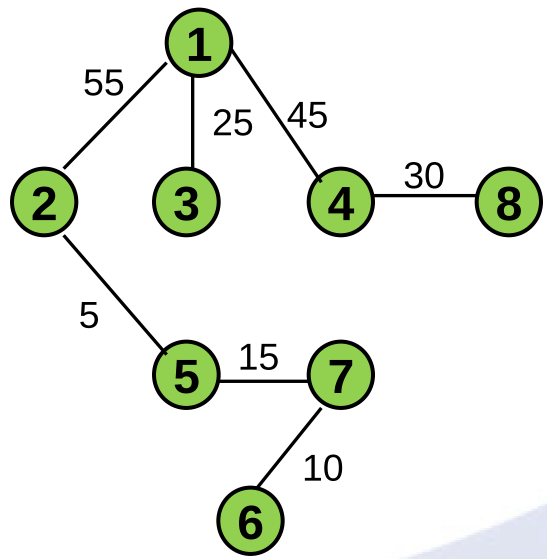
4.6 单源点最短路径

■ 3. 最短路径生成树

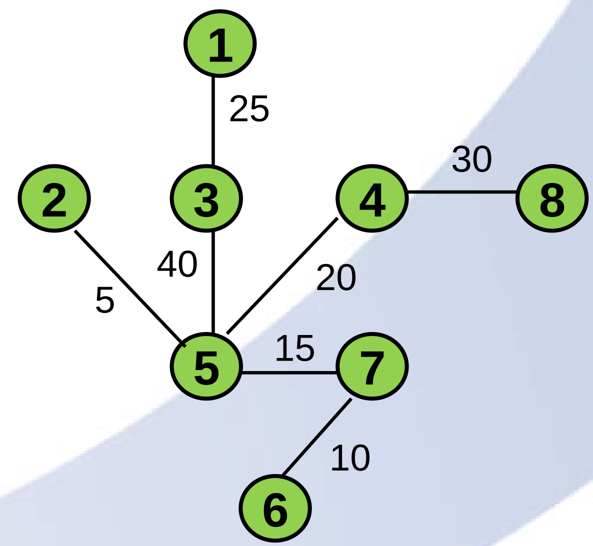
- 对于无向连通图G，由结点v到其余各结点的最短路径的边构成G的一棵生成树，称为最短路径生成树。
- 注：不同起点v的生成树可能不同。



原始图



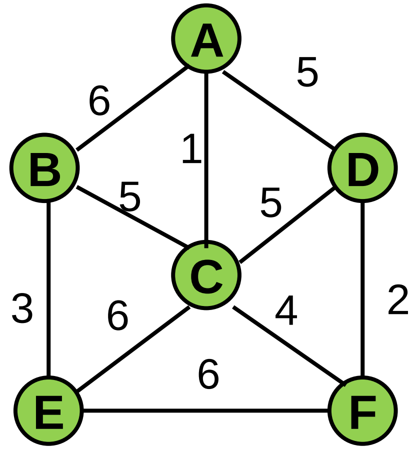
由结点1出发的最短路径生成树



最小成本生成树

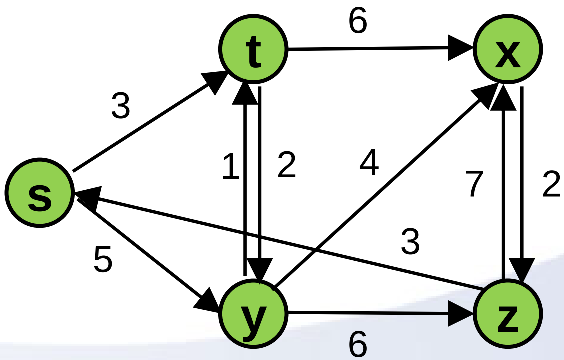
作业-课后练习11

- 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。



作业-课后练习12

- 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5

- 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5

- 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