algorithm

6.贪心法b

本节内容

- 最优前缀码
  - 哈夫曼算法
- 最小生成树
  - Prim
  - Kruskal
- 单源最短路径问题
  - Dijkstra

二元前缀码

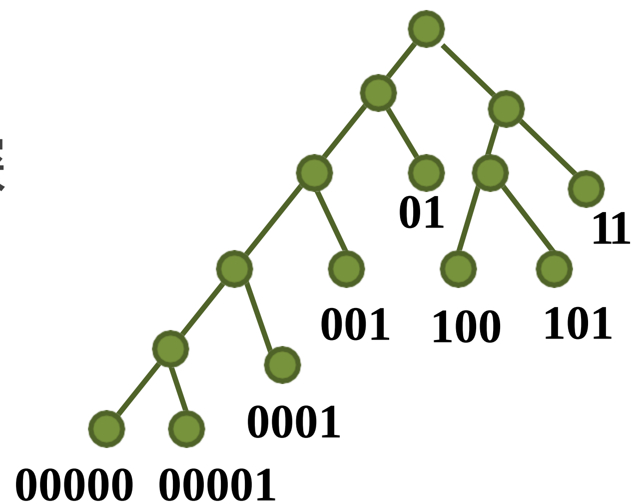
- 二元前缀码
- 用0-1字符串作为代码表示字符，要求任何字符的代码都不能作为其它字符代码的前缀
- 非前缀码的例子

  a: 001, b: 00, c: 010, d: 01

- 解码的歧义，例如字符串 0100001
- 解码1: 01, 00, 001 d, b, a
- 解码2: 010, 00, 01 c, b, d

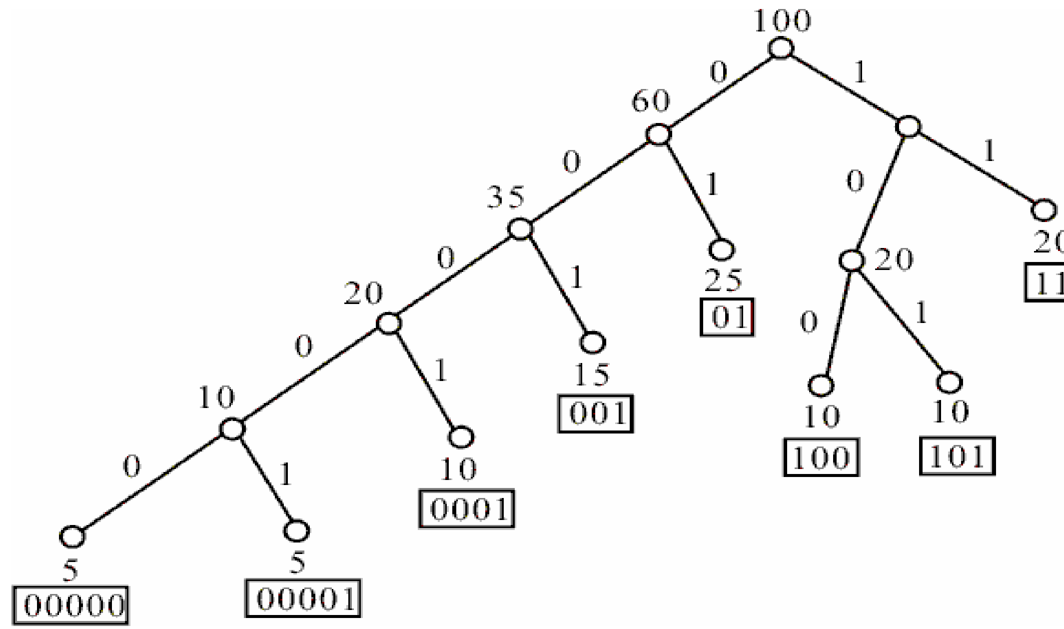
前缀码的二叉树表示

- 前缀码：
- {00000, 00001, 0001, 001, 01,100,101,11}
- 构造树：
- 0-左子树
- 1-右子树
- 码对应一片树叶最大位数为树深



平均传输位数

$$B=\sum_{i=1}^{n} f(x_i)d(x_i)$$



B=[(5+5)×5+10×4

+(15+10+10)×3

+(25+20)×2]/100

=2.85

- 问题:
  - 给定字符集 $C=\{x_1,x_2,\ldots,x_n\}$
  - 和每个字符的频率 $f(x_i)$， $i=1,2,\ldots,n$
  - 求关于C的一个最优前缀码(平均传输位数最小)

哈夫曼树算法伪码

- 算法 Huffman(C)
- 输入： $C=\{x_1, x_2,\ldots, x_n\}$, $f(x_i)$, i=1,2,…,n.
- 输出： Q / /队列

1. $n \leftarrow |C|$
2. $Q \leftarrow C$ //频率递增队列Q
3. for $i \leftarrow 1$ to $n-1$ do
4. 　　$z \leftarrow$Allocate-Node() //生成结点 z
5. 　　$z.left \leftarrow Q$中最小元 //最小作z左儿子
6. 　　$z.right \leftarrow Q$中最小元 //最小作z右儿子
7. 　　$f(z) \leftarrow f(x)+f(y)$
8. 　　Insert(Q,z) // 将 z 插入Q
9. return Q

实例

- 输入 *a*:45; *b*:13; *c*:12; *d*:16; *e*:9; *f*:5
- 编码：

  f--0000,

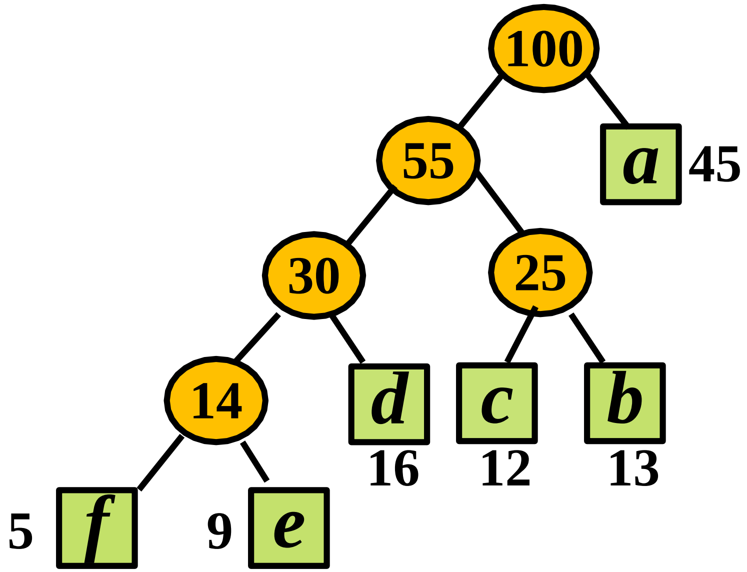
  e--0001,

  d--001,

  c--010,

  b—011,

  a--1



- 平均位数：

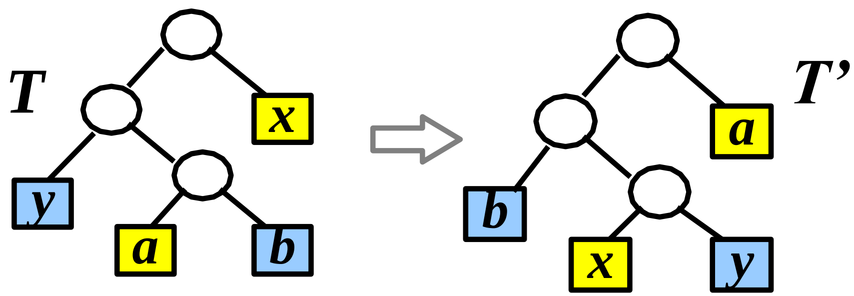
4×(0.05+0.09)+3×(0.16+0.12+0.13)+1×0.45=2.24

最优前缀码性质：引理1

- 引理1: C是字符集，$\forall c \in C$, $f(c)$为频率, $x,y \in C$, $f(x)$, $f(y)$频率最小, 那么存在最优二元前缀码使得 x, y 码字等长且仅在最后一位不同

$f(x) \le f(a)$

$f(y) \le f(b)$

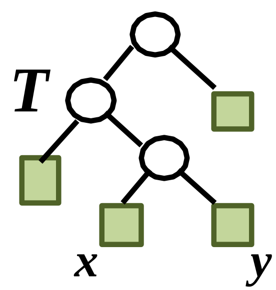


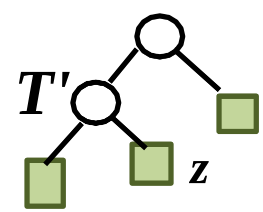
$$B(T) - B(T') = \sum_{i \in C} f[i] d_T(i) - \sum_{i \in C} f[i] d_{T'}(i) \ge 0$$

- 其中$d_T(i)$为i 在T 中的层数(i到根的距离)

引理2

- 引理 设T是二元前缀码的二叉树,$\forall x, y \in T$, x, y是树叶兄弟，z是 x, y的父亲,
- 令 $T' = T - \{x, y\}$
- 令 z 的频率 $f(z) = f(x) + f(y)$
- T'是对应二元前缀码 $C' = (C - \{ x, y \}) \cup \{ z \}$ 的二叉树
- 那么 $B(T) = B(T') + f(x) + f(y)$





引理2证明

证 $\forall c \in C-\{x,y\}$, 有

$$d_T(c) = d_{T'}(c) \Rightarrow f(c)d_T(c) = f(c)d_{T'}(c)$$

$$d_T(x) = d_T(y) = d_{T'}(z) + 1$$

$$\begin{aligned}
B(T) &= \sum_{i \in T} f(i)d_T(i) \\
&= \sum_{i \in T, i \neq x,y} f(i)d_T(i) + f(x)d_T(x) + f(y)d_T(y) \\
&= \sum_{i \in T', i \neq z} f(i)d_{T'}(i) + f(z)d_{T'}(z) + (f(x) + f(y)) \\
&= B(T') + f(x) + f(y)
\end{aligned}$$

哈夫曼算法的证明及应用

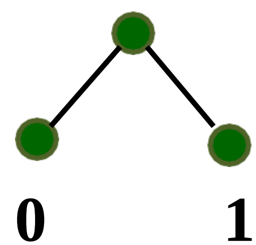
- 两个引理
- 引理1：设C是字符集，$\forall c\in C$, $f(c)$为频率，$x, y\in C$，$f(x)$，$f(y)$频率最小，那么存在最优二元前缀码使得 x, y 码字等长，且仅在最后一位不同
- 引理2：设 T 是二元前缀码所对应的二叉树，$\forall\ x, y\in T$, x, y是树叶兄弟，z 是 x, y 的父亲，令$T' = T - \{x, y\}$, 且令 z 的频率 $f(z)=f(x)+f(y)$，T'是对应于二元前缀码 $C' =(C-\{x, y\})\cup\{z\}$的二叉树，那么 $B(T)=B(T')+f(x)+f(y)$

算法正确性证明思路

- 定理 Huffman 算法对任意规模为n（$n\geq 2$）的字符集C 都得到关于C 的最优前缀码的二叉树
- 归纳基础
  - 证明：对于n=2的字符集, Huffman算法得到最优前缀码
- 归纳步骤
  - 证明：假设Huffman算法对于规模为 k 的字符集都得到最优前缀码，那么对于规模为k+1的字符集也得到最优前缀码

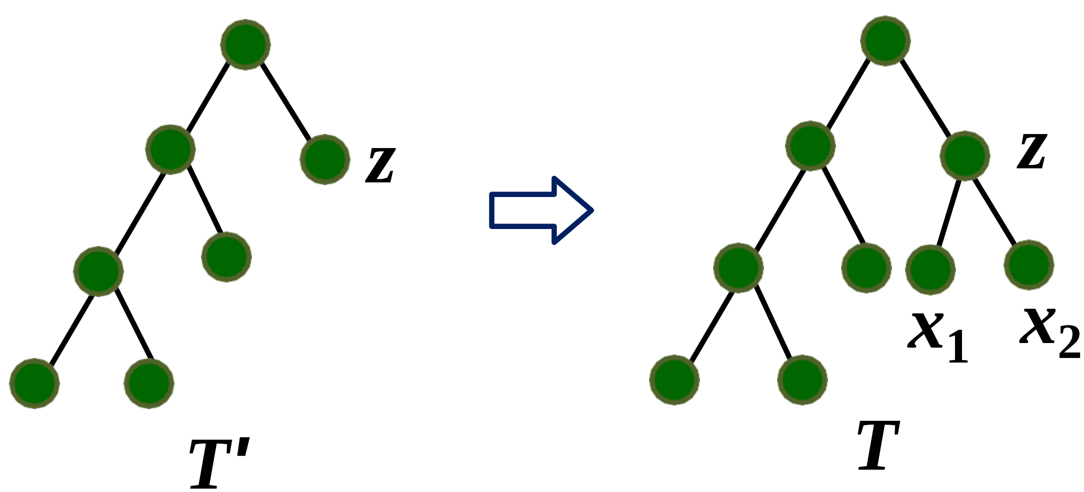
归纳基础

- n=2，字符集 C={x_1, x_2}，
- 对任何代码的字符至少都需要1位二进制数字
- Huffman算法得到的代码是 0 和 1，是最优前缀码



归纳步骤

- 假设Huffman算法对于规模为 k 的字符集都得到最优前缀码
- 考虑规模为 $k+1$ 的字符集 $C = \{x_1, x_2, \ldots, x_{k+1}\}$, 其中 $x_1, x_2 \in C$ 是频率最小的两个字符
- 令$C' = (C - \{x_1, x_2\}) \cup \{z\}$，$f(z) = f(x_1) + f(x_2)$
- 根据归纳假设，算法得到一棵关于字符集C'，频率 $f(z)$ 和 $f(x_i)$ $(i = 3, 4, \ldots, k+1)$ 的最优前缀码的二叉树T'
- 把 x_1, x_2作为 z 的儿子附到 T'上，得到树T，那么T是关于 $C = (C' - \{z\}) \cup \{x_1, x_2\}$ 的最优前缀码的二叉树



归纳步骤

- 如若不然, 存在更优树 T^*, $B(T^*)<B(T)$,
- 且由引理1，其树叶兄弟是 x_1和 x_2
- 去掉 T^*中 x_1和 x_2，得到$T^{*\prime}$
- 根据引理2

$$
\begin{aligned}
B(T^{*\prime}) &= B(T^*) - (f(x_1)+f(x_2)) \\
&< B(T) - (f(x_1)+f(x_2)) \\
&= B(T')
\end{aligned}
$$

- 与 T'是一棵关于 C'的最优前缀码的二叉树矛盾

应用：文件归并

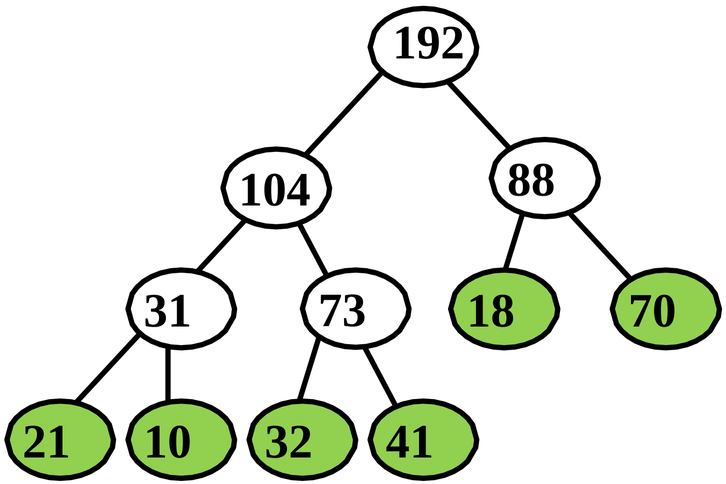
- 问题：给定一组不同长度的排好序文件构成的集合

$$S = \{ f_1, \ldots , f_n\}$$

- 其中 f_i 表示第 i 个文件含有的项数
- 使用二分归并将这些文件归并成一个有序文件
- 归并过程对应于二叉树：
- 文件为树叶，f_i 与 f_j 归并的文件是它们的父结点.

两两顺序归并

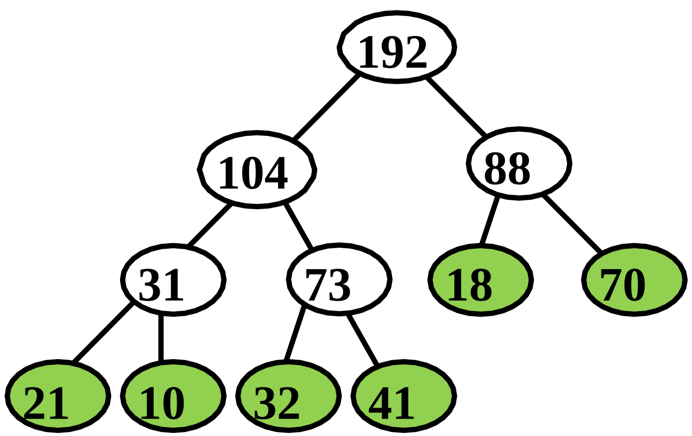
- 实例： $S = \{ 21,10,32,41,18,70 \}$



归并代价

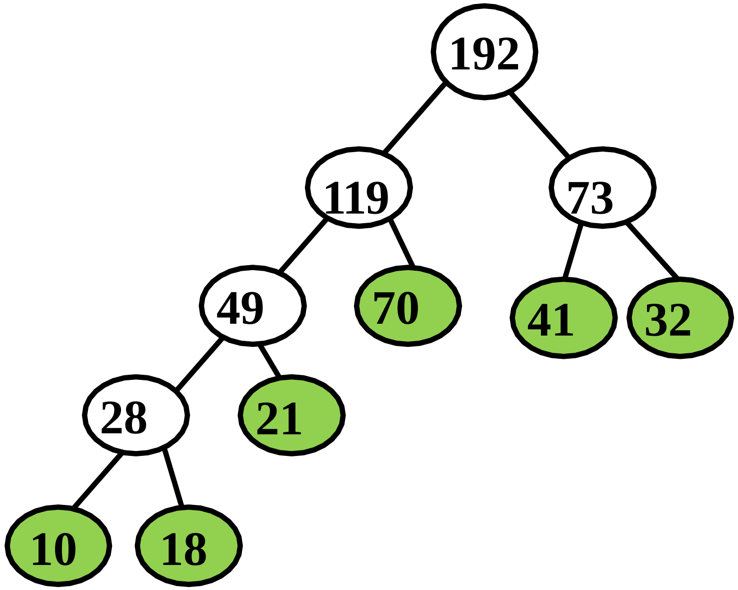
- (21+10–1)+(32+41–1)+(18+70–1)+(31+73–1)+(104+88–1) = 483
- (21+10+32+41)×3+(18+70)×2–5=483
- 代价计算公式

$$\sum_{i \in S} d(i) f_i - (n-1)$$



实例：Huffman树归并

- 输入： S={21,10,32,41,18,70}



- 代价: (10+18)×4+21×3+(70+41+32)×2–5=456

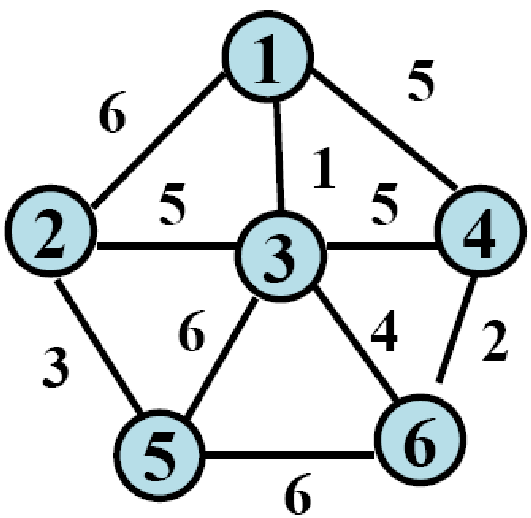
最小生成树

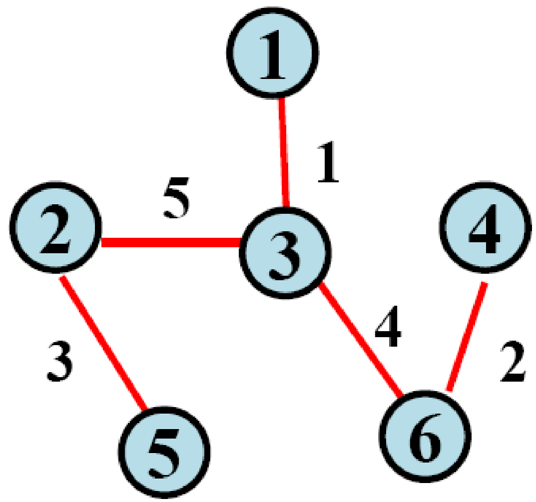
- 无向连通带权图
- $G = (V, E, W)$,
- 其中 $w(e) \in W$ 是边 e 的权
- G 的一棵生成树 T 是包含了G 的所有顶点的树,树中各边的权之和$W(T)$ 称为树的权，具有最小权的生成树称为 G 的最小生成树

最小生成树的实例

$G=(V,E,W), V=\{1,2,3,4,5,6\}, W$ 如图所示

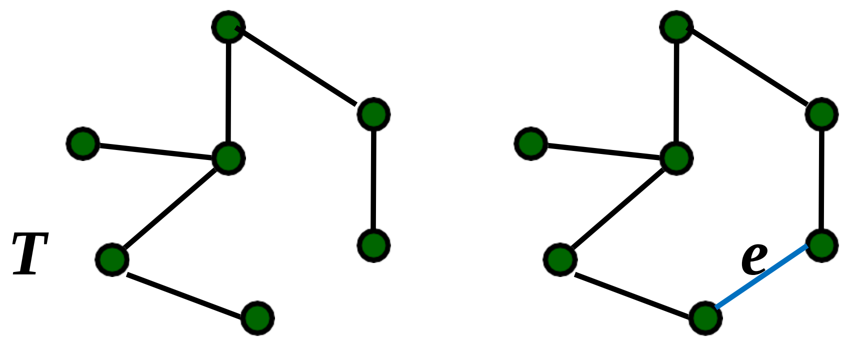
$E = \{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,5\},$
$\{3,4\},\{3,5\},\{3,6\},\{4,6\},\{5,6\}\}$



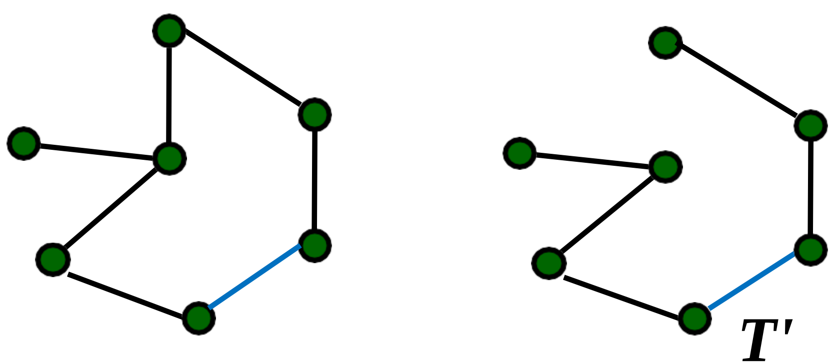


生成树的性质

- 命题1 设G是 n 阶连通图, 那么
  - (1) T 是G 的生成树当且仅当T 无圈且有 n - 1条边
  - (2) 如果 T 是 G 的生成树，$e \notin T$，那么 $T \cup \{e\}$含有一个圈C (回路)



  - (3) 去掉圈C的任意一条边，就得到G 的另外一棵生成树T'。



生成树性质的应用

- 算法步骤：选择边
- 约束条件：不形成回路
- 截止条件：边数达到 n-1
- 改进生成树 T 的方法
- 在 T 中加一条非树边 e，形成回路C，在 C 中去掉一条树边 e_i，形成一棵新的生成树 T'

$$W(T')-W(T) = W(e)-W(e_i)$$

- 若 $W(e) \leq W(e_i)$，则 $W(T') \leq W(T)$

求最小生成树

- 问题:
  - 给定连通带权图 $G = (V, E, W)$,
  - $w(e) \in W$是边 e 的权
  - 求 G 的一棵最小生成树

- 贪心法:
  - Prim 算法,
  - Kruskal 算法
- 生成树在网络中有着重要应用

Prim算法

- 输入：图 $G=(V,E,W)$, $V=\{1,2,\ldots,n\}$
- 输出：最小生成树 T

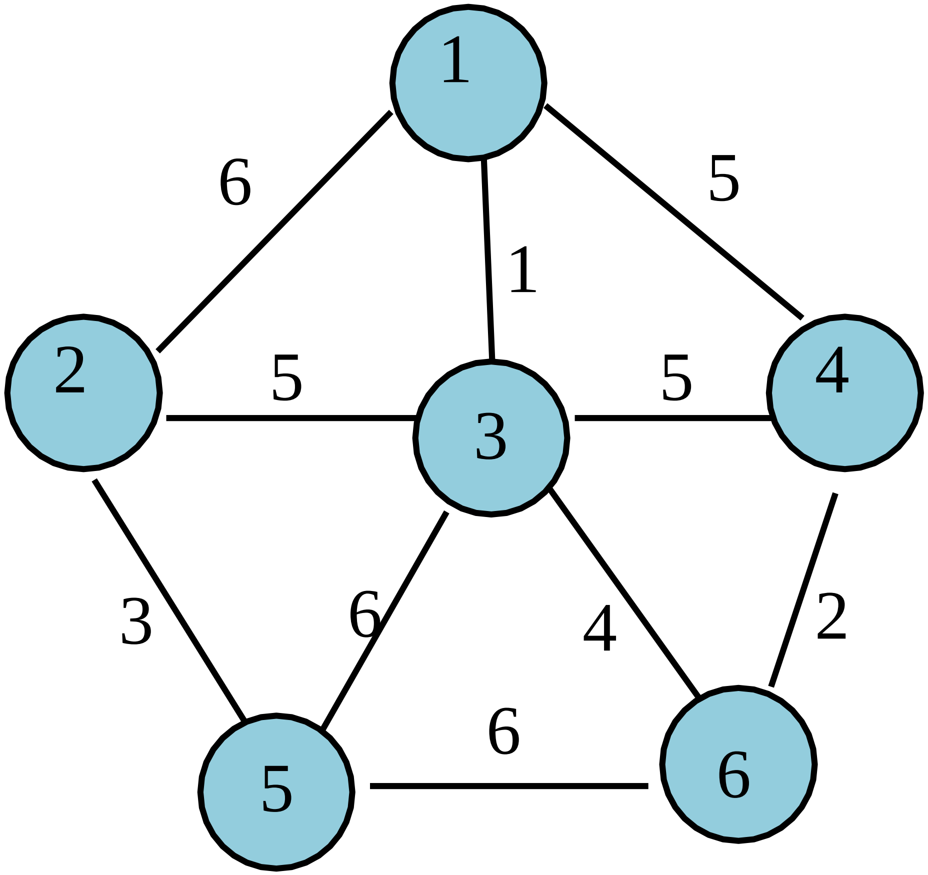
- 设计思想:
- 初始 $S = \{1\}$,
- 选择连接 S 与 $V–S$ 集合的最短边 $e =\{ i, j \}$，其中 $i\in S, j\in V–S$
- 将 e 加入树 T，j 加入 S
- 继续执行上述过程，直到 $S=V$ 为止

伪码

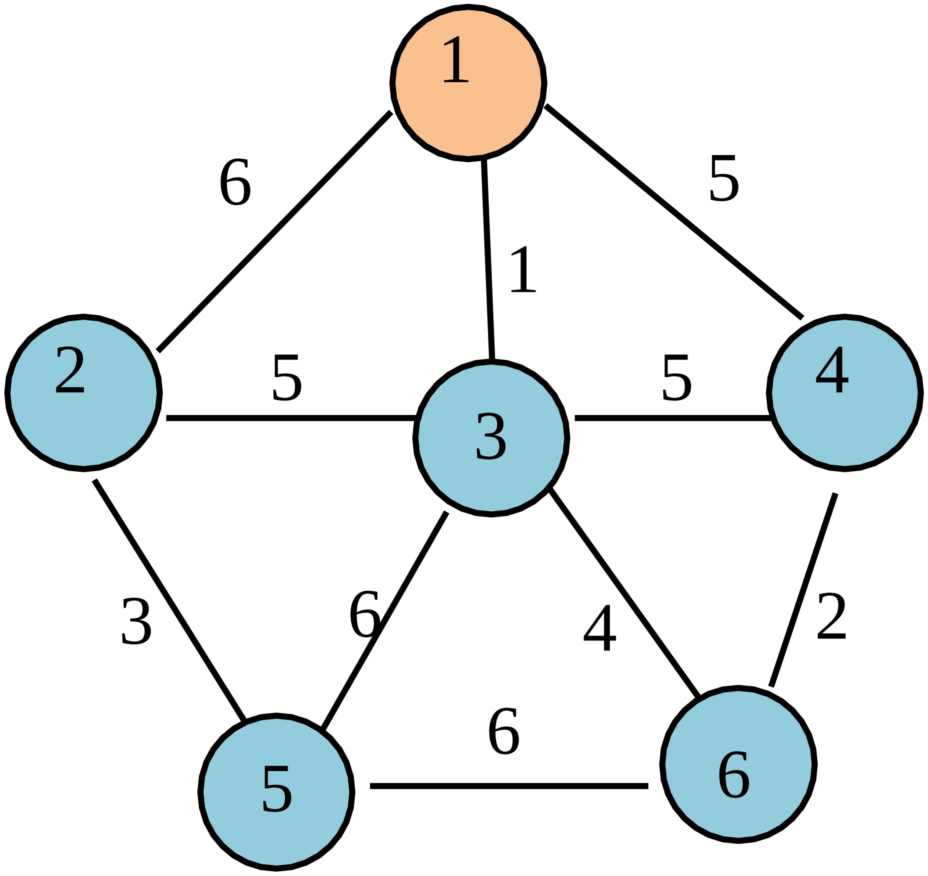
算法 Prim (G, E, W)

1. $S \leftarrow \{ 1 \}$
2. while $V - S \neq \varnothing$ do
3. 　从V–S中选择 j 使得 j 到 S中顶点的边权最小
4. 　$S \leftarrow S \cup \{ j \}$

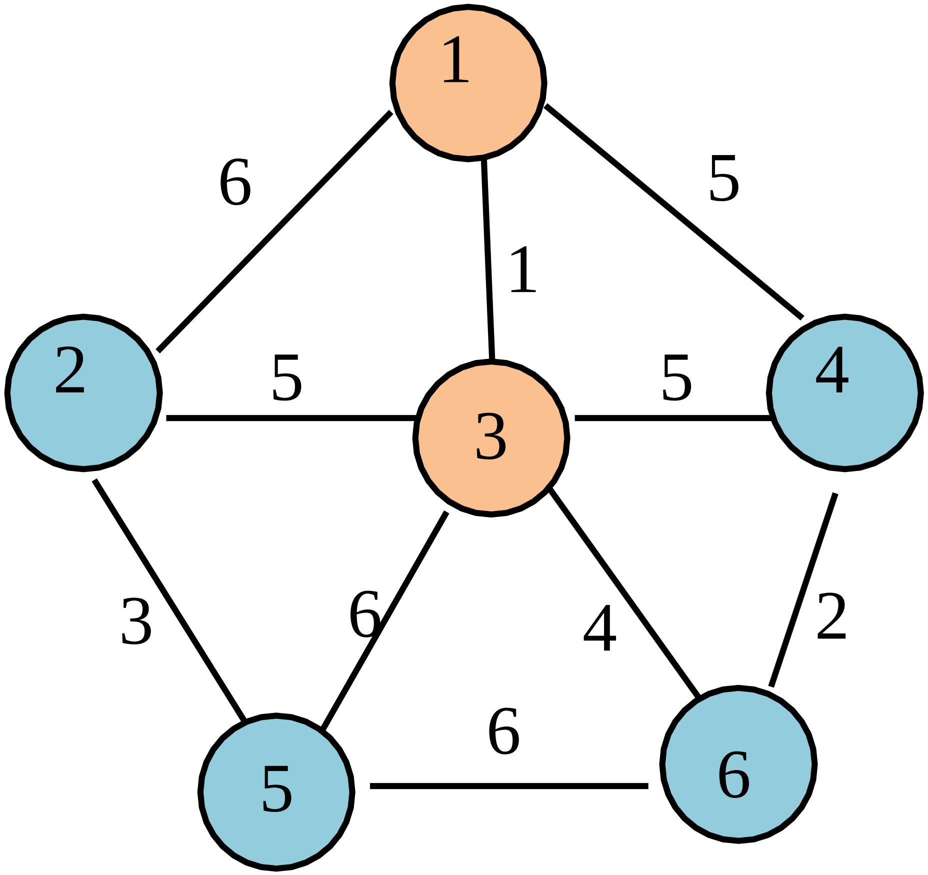
实例



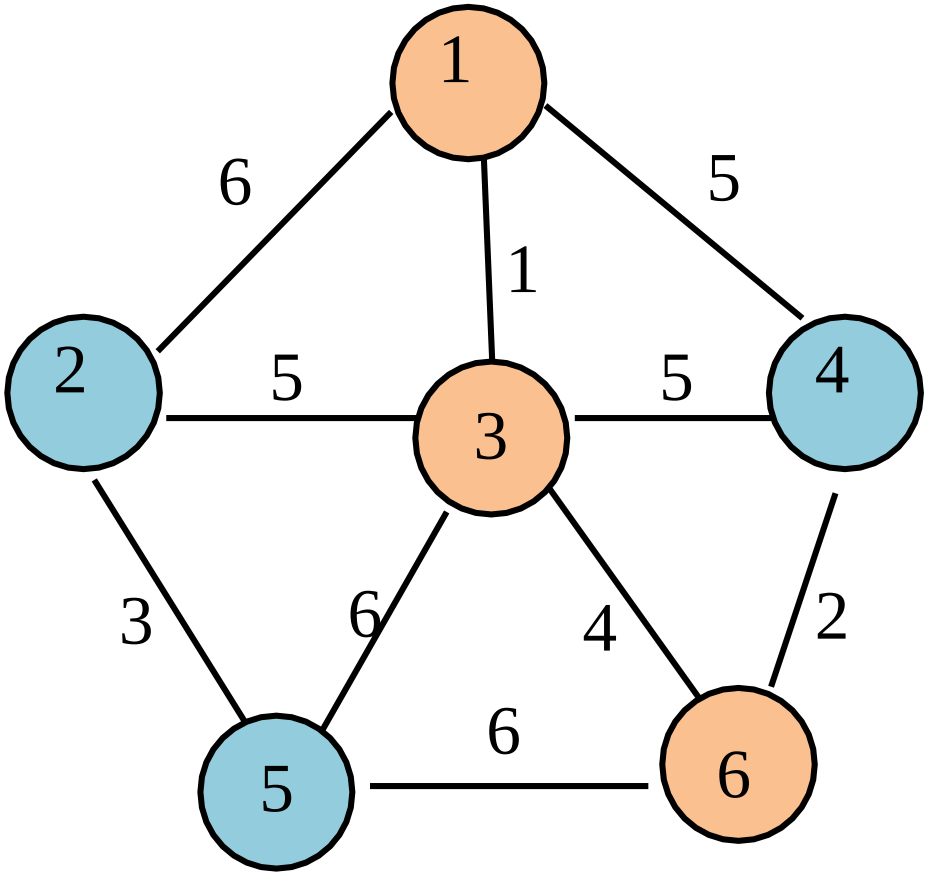
实例



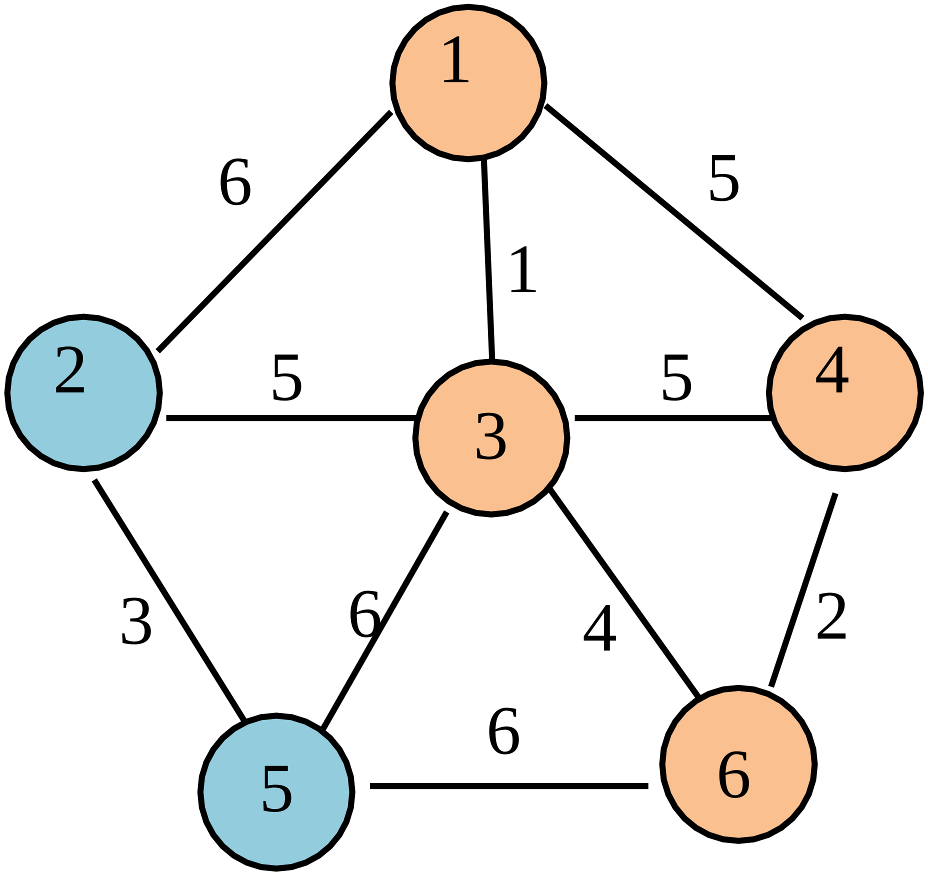
实例



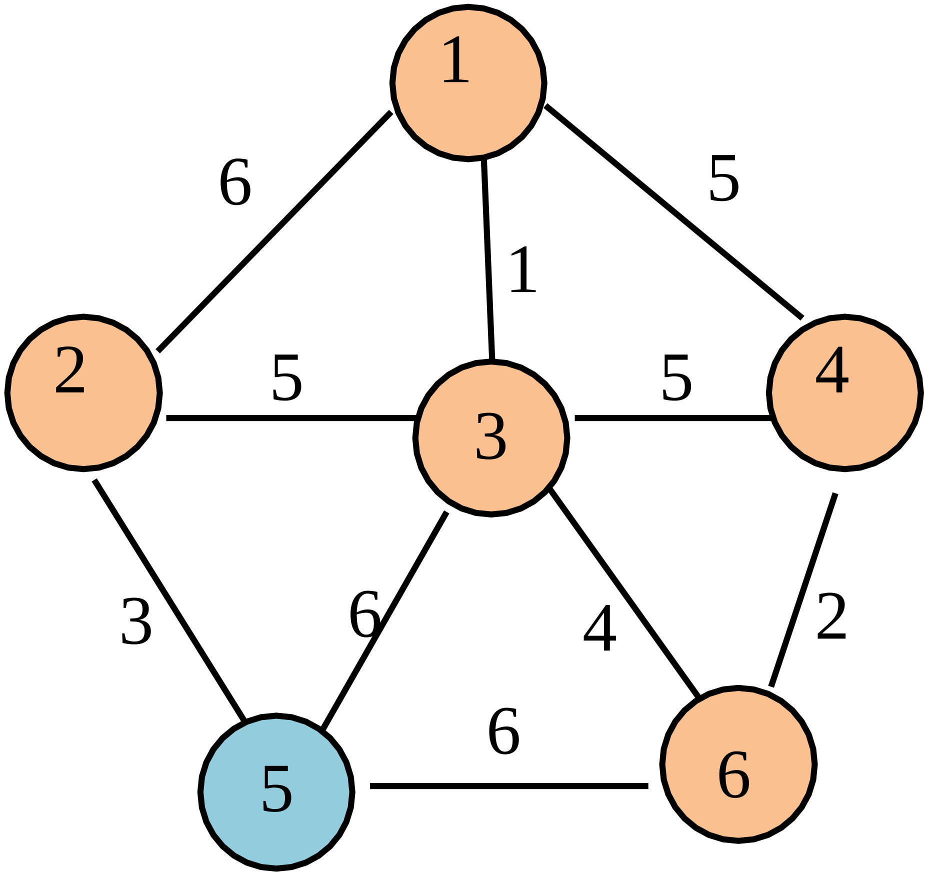
实例



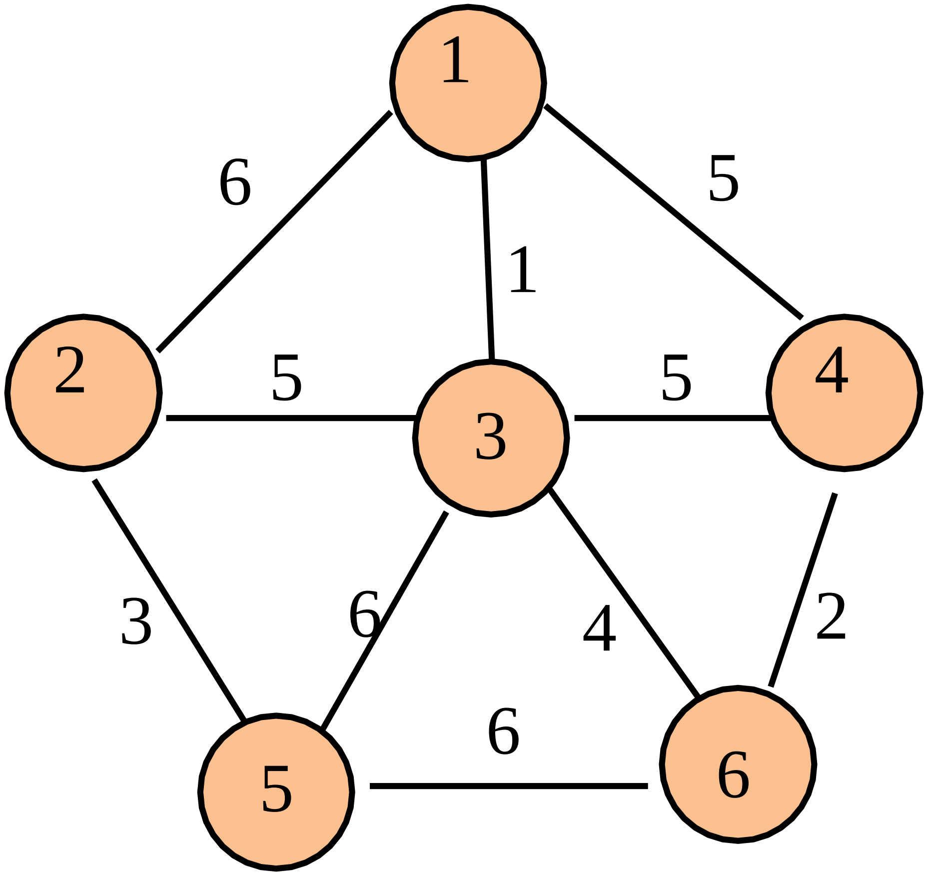
实例



实例



实例

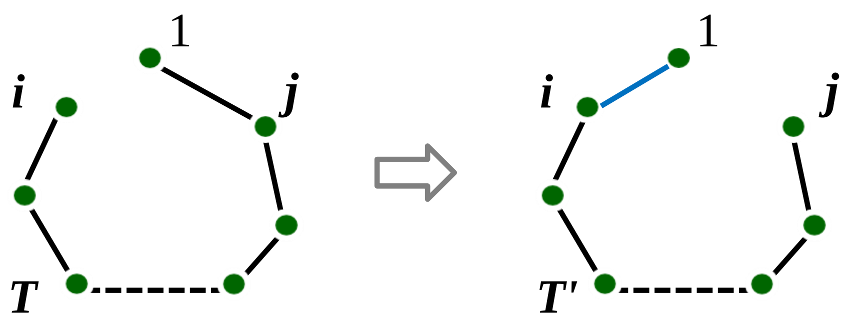


正确性证明：归纳法

- 命题：对于任意 $k < n$, 存在一棵最小生成树包含算法前 k 步选择的边
- 归纳基础： $k = 1$, 存在一棵最小生成树 T 包含边 $e = \{1, i\}$, 其中$\{1, i\}$是所有关联 1 的边中权最小的
- 归纳步骤：假设算法前 k 步选择的边构成一棵最小生成树的边，则算法前 $k+1$ 步选择的边也构成一棵最小生成树的边

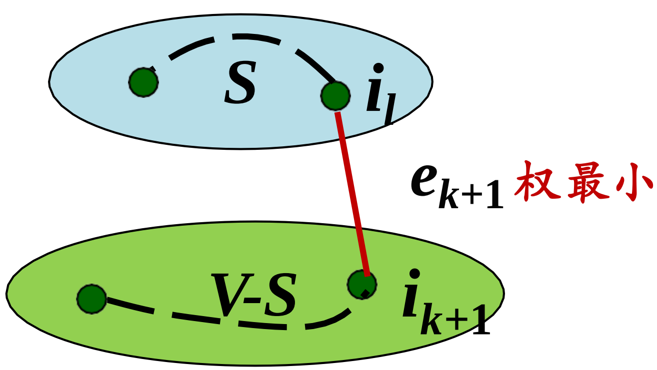
归纳基础

- 证明：存在一棵最小生成树 T 包含关联结点1的最小权的边 $e=\{1,i\}$
- 证 设 T 为一棵最小生成树，假设 T 不包含$\{1, i\}$, 则$T \cup \{\{1,i\}\}$ 含有一条回路，回路中关联1的另一条边$\{1, j\}$
- 用$\{1, i\}$替换$\{1, j\}$得到树 T'， 则 T' 也是生成树，且 $W(T') \leq W(T)$



归纳步骤

- 假设算法进行了k步，生成树的边为 $e_1, e_2, \ldots, e_k$，这些边的端点构成集合S
- 由归纳假设存在 G 的一棵最小生成树 T 包含这些边
- 算法第 $k+1$ 步选择顶点 i_{k+1}, 则 i_{k+1} 到 S 中顶点边权最小,设此边 $e_{k+1}=\{i_{k+1}, i_l\}$. 若$e_{k+1} \in T$，算法 $k+1$ 步显然正确

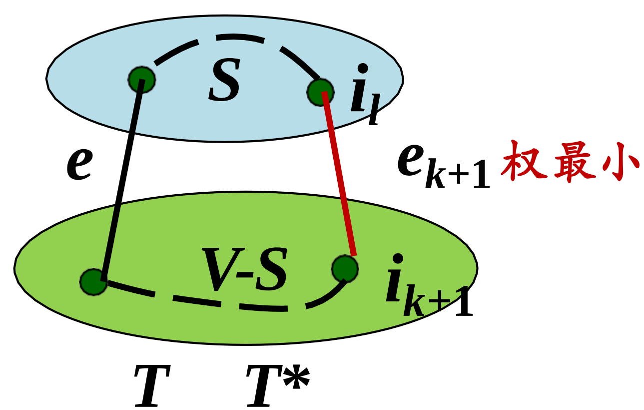


归纳步骤

- 假设 T 不含有 e_{k+1}，则将 e_{k+1}加到 T 中形成一条回路. 这条回路有另外一条连接 S 与 $V–S$中顶点的边 e,
- 令 $T^*= (T–\{e\}) \cup \{e_{k+1}\}$
- 则 T^*是 G 的一棵生成树,包含$e_1,e_2,\ldots,e_{k+1}$, 且

$$W(T^*) \le W(T)$$

- 算法到 k+1步仍得到最小生成树



时间复杂度

- 算法步骤执行$O(n)$次
- 每次执行$O(n)$时间:
- 找连接 S与V-S 的最短边
- 算法时间：$T(n) = O(n^2)$

Kruskal算法

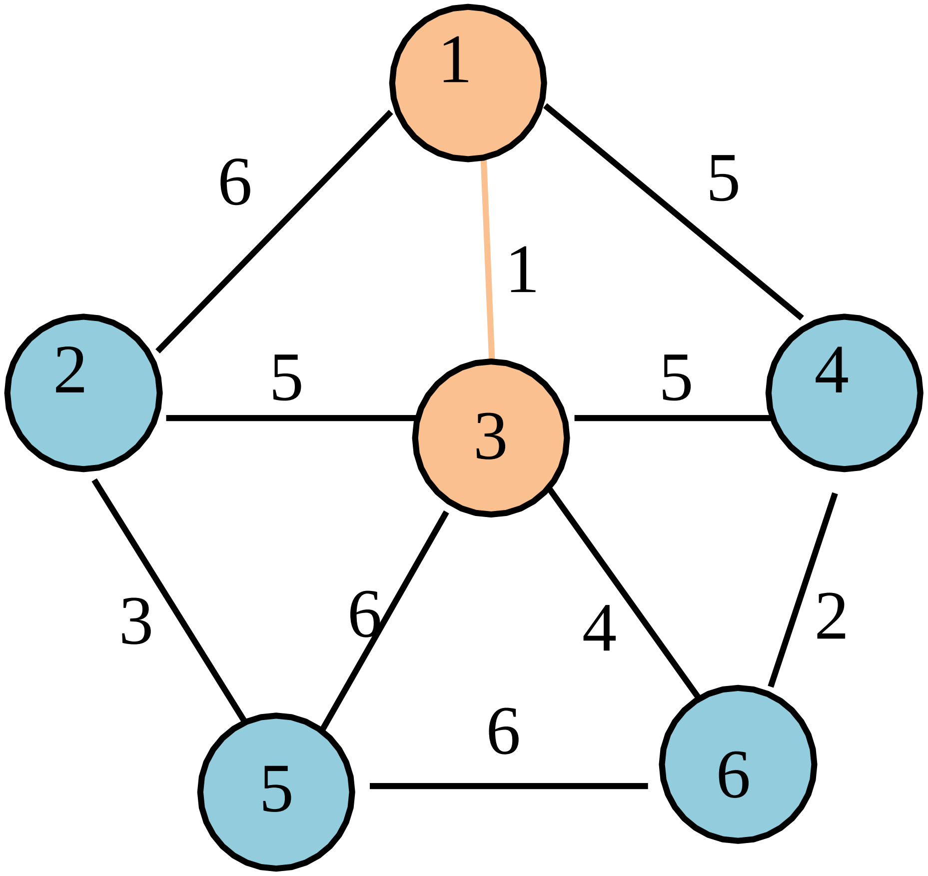
- 设计思想
- 输入：图$G=(V,E,W)$, $V=\{1,2,\ldots,n\}$
- 输出：G的最小生成树 T
- 设计思想:
  - 按照长度从小到大对边排序
  - 依次考察当前最短边 e，如果e 与T 的边不构成回路，则把 e 加入树 T，否则跳过 e，直到选择了n-1 条边为止

伪码

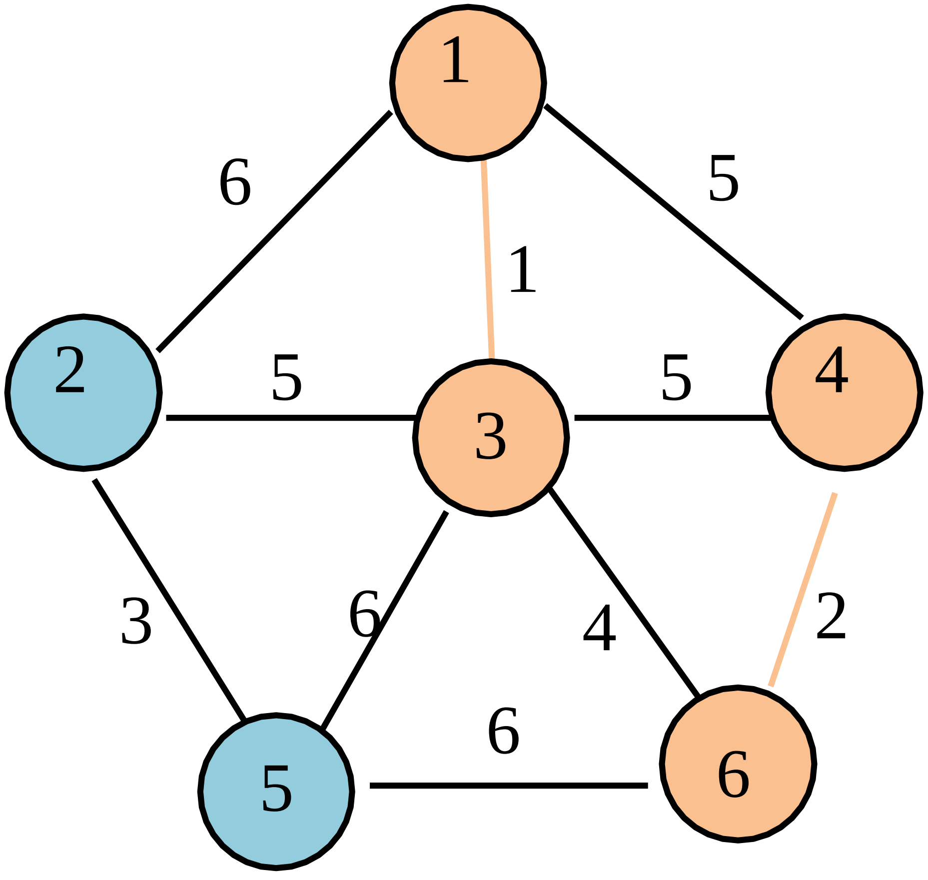
- 算法 Kruskal
- 输入：连通图G //顶点数n，边数m
- 输出：G 的最小生成树

1. 权从小到大排序E的边, $E=\{e_1,e_2,\ldots,e_m\}$
2. $T \leftarrow \varnothing$
3. repeat
4. $\quad e \leftarrow E$中的最短边
5. $\quad$ if e 的两端点不在同一连通分支
6. $\quad$ then $T \leftarrow T \cup \{e\}$
7. $\quad E \leftarrow E - \{e\}$
8. $\quad$ until T 包含了$n - 1$条边

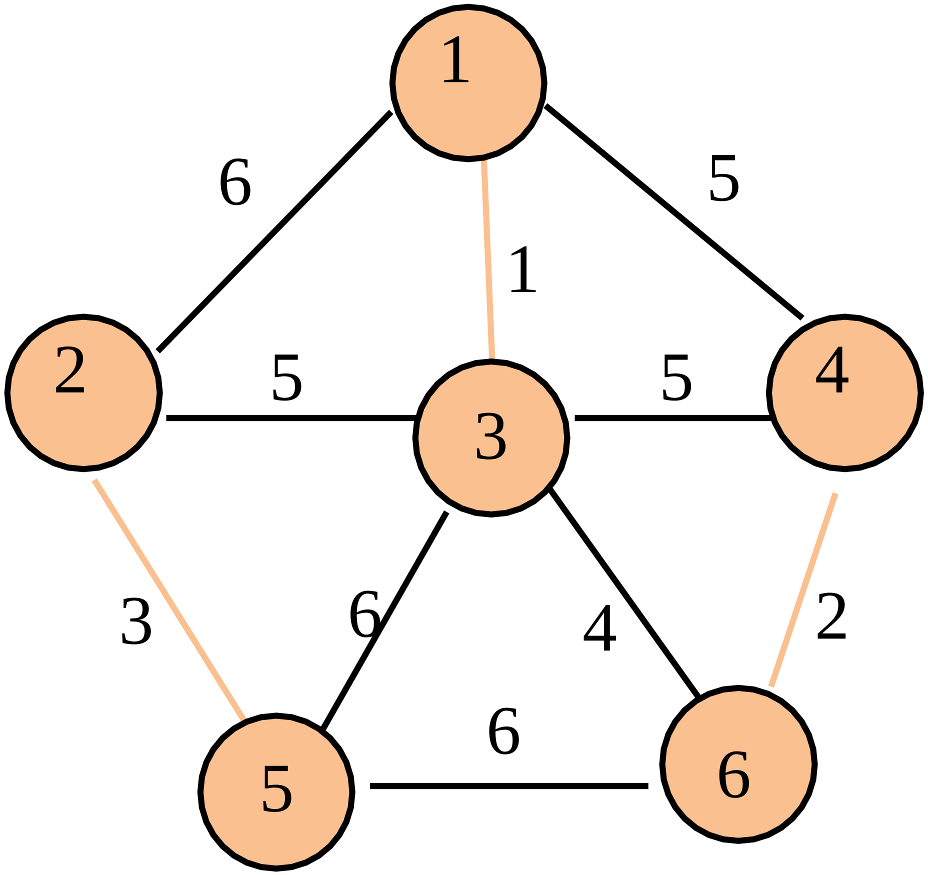
实例



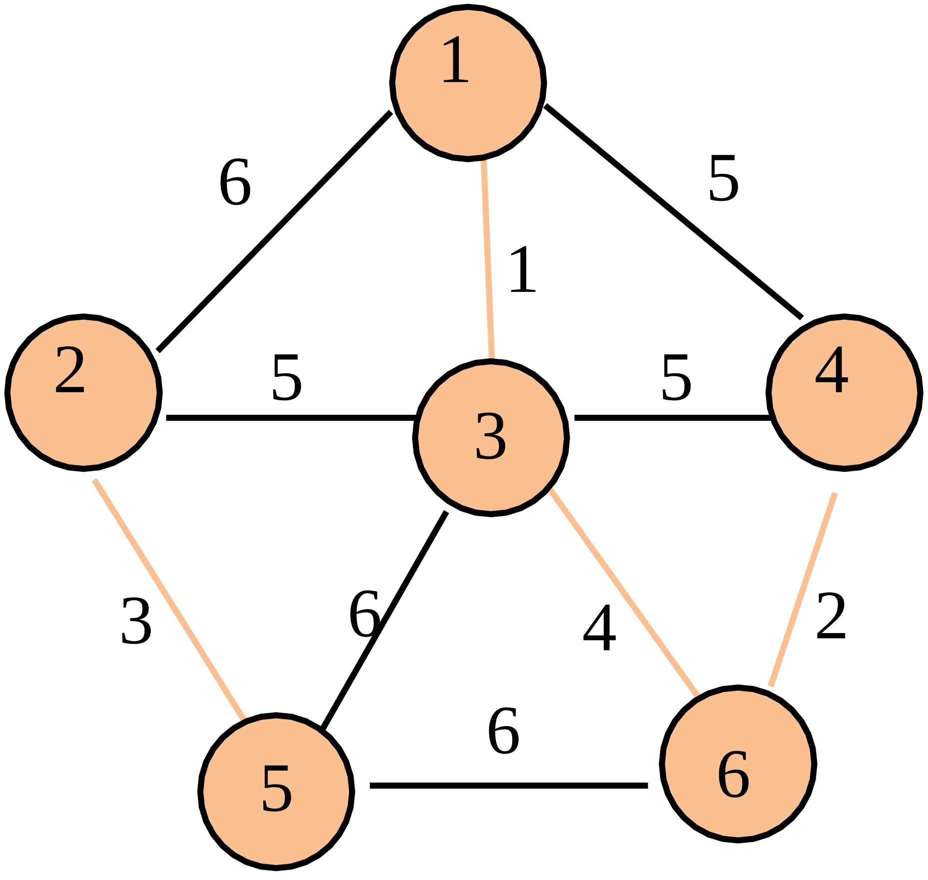
实例



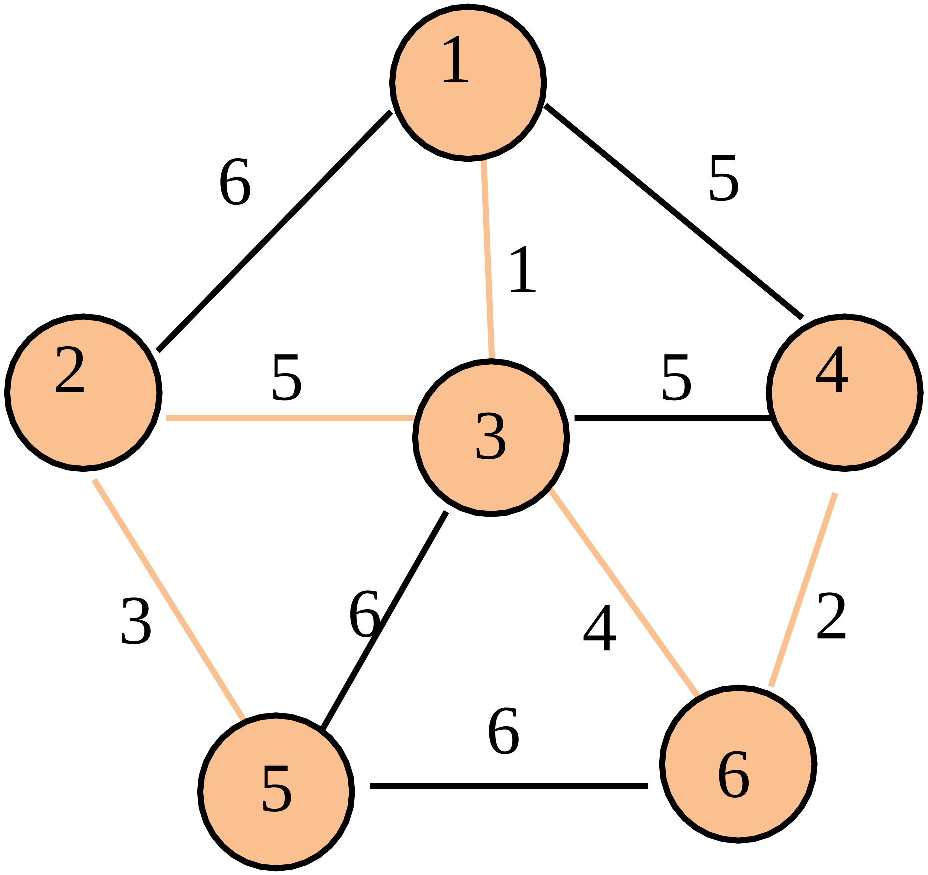
实例



实例



实例

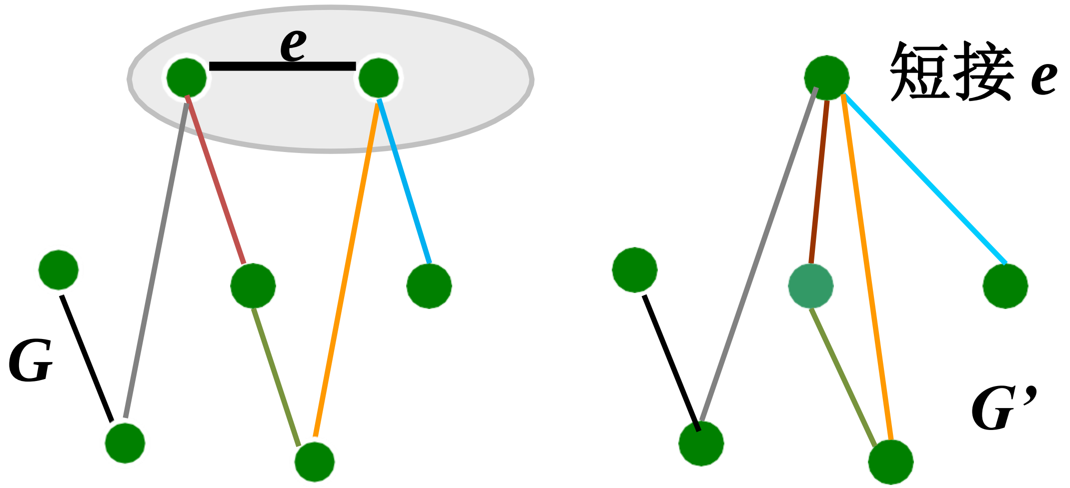


正确性证明思路

- 命题：对于任意 n, 算法对 n 阶图找到一棵最小生成树
- 证明思路：
- 归纳基础
  - 证明：$n = 2$, 算法正确
  - G只有一条边，最小生成树就是 G
- 归纳步骤
  - 证明：假设算法对于n 阶图是正确的，其中$n>1$
  - 则对于任何$n+1$阶图算法也得到一棵最小生成树

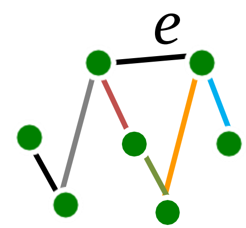
短接操作

- 任给 n+1个顶点的图 G, G中最小权边 e={i, j}，从G 中短接i 和 j, 得到图 G’



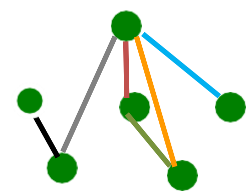
归纳步骤证明

- 对于任意 n+1阶图G短接最短边e，得到 n 阶图 G'



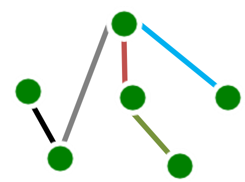
G

- 根据归纳假设算法得到G'的最小生成树 T'



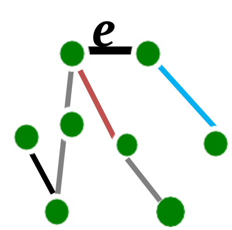
G'

- 将被短接的边 e “拉伸”回到原来长度，得到树 T



T'

- 证明 T 是 G 的最小生成树



T

T是G的最小生成树

- $T=T' \cup \{e\}$是关于G 的最小生成树
- 否则存在G 的含边e 的最小生成树T^*, $W(T^*)<W(T)$
  - 如果 $e \notin T^*$，在T^* 中加边e，形成回路
  - 去掉回路中任意别的边所得生成树的权仍旧最小
- 在T^*短接e得到G'的生成树$T^*-\{e\}$

$$W(T^*-\{e\}) = W(T^*) - w(e) < W(T) - w(e) = W(T')$$

- 与T'的最优性矛盾.

算法实现与时间复杂度

- 建立FIND数组，FIND[i] 是结点 i 的连通分支标记，TCOUNT[i] 是连通分支对应节点数量

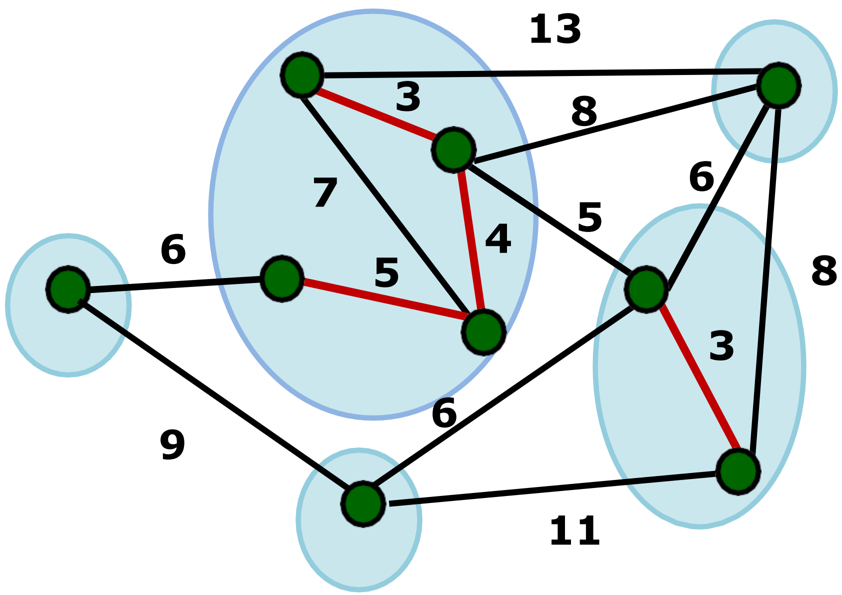
(1) 初始 FIND[i] = i，TCOUNT[i]=0

(2) 连通分支合并，较小分支标记更新为较大分支标记

- 每个结点至多更新 log$n$次，
- 建立和更新 FIND数组: $O(n\log n)$
- 时间：$O(m\log m)+O(n\log n)+O(m) = O(m\log m)$ ($m>n-1$)

应用：数据分组问题

- 一组数据(照片,文件,生物标本)要把它们按照相关性进行分类
- 用相似度函数或“距离”来描述个体之间的差距
- 如果分成5类，使得每类内部的个体尽可能相近，不同类之间的个体尽可能地“远离”，如何划分?



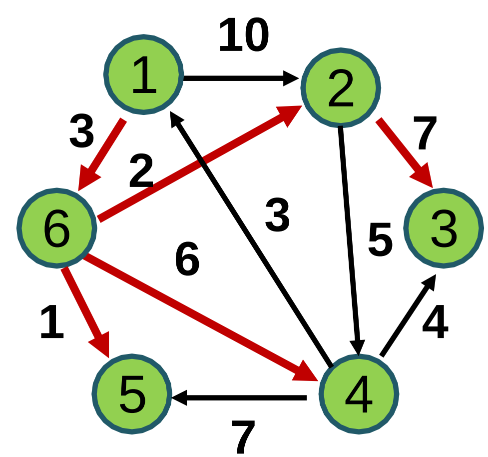
单链聚类

- 类似于Kruskal算法
  - 按照边长从小到大对边排序
  - 依次考察当前最短边 e ，如果 e 与已经选中的边不构成回路，则把 e 加入集合，否则跳过 e，计数图的连通分支个数
  - 直到保留了k个连通分支为止

单源最短路径

- 给定带权有向网络 $G=(V,E,W)$，每条边$e=<i,j>$的权$w(e)$为非负实数，表示从 i 到 j 的距离，源点 $s\in V$
- 求: 从 s 出发到达其它结点的最短路径

- 源点：1
- $1\to6\to2$：$short\ [2] = 5$
- $1\to6\to2\to3$：$short\ [3] = 12$
- $1\to6\to4$：$short\ [4] = 9$
- $1\to6\to5$：$short\ [5] = 4$
- $1\to6$：$short\ [6] = 3$



Dijkstra算法有关概念

- $x \in S \Leftrightarrow x \in V$ 且从 s 到 x 的最短路径已经找到
- 初始：$S = \{ s \}$，$S = V$ 时算法结束
- 从s到u相对于S 的最短路径：从s到u且仅经过 S 中顶点的最短路径
- $dist\ [u]$：从s到u相对S最短路径的长度
- $short\ [u]$：从s到u的最短路径的长度
- $dist\ [u] \geq short[u]$

算法设计思想

- 输入：有向图 $G = (V, E, W)$, $V = \{ 1, 2, \ldots , n \}$， $s = 1$
- 输出：从s到每个顶点的最短路径

1. 初始 $S=\{1\}$
2. 对于$i \in V-S$，计算1到 i 的相对 S 的最短路，长度 $dist[i]$
3. 选择$V-S$中 $dist$ 值最小的 j，将 j 加入 S，修改$V-S$中顶点的 $dist$ 值
4. 继续上述过程，直到 $S=V$ 为止

伪码

算法 Dijkstra

$S \leftarrow \{ s \}$

$dist[s] \leftarrow 0$

for $i \in V - \{ s \}$ do

    $dist[i] \leftarrow w(s,i)$ //s到i没边, $w(s,i)=\infty$

while $V - S \neq \varnothing$ do

    从$V - S$取相对S的最短路径顶点 j

    $S \leftarrow S \cup \{ j \}$

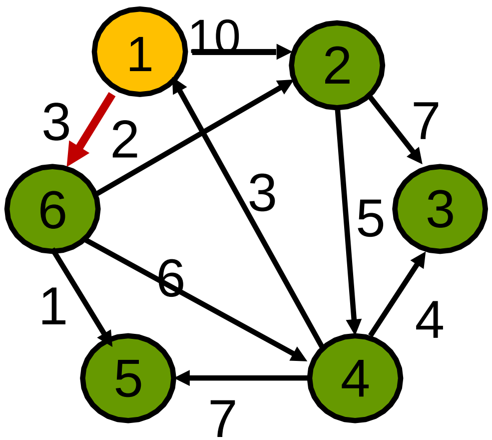
    for $i \in V - S$ do

        if $dist[j] + w(j,i) < dist[i]$

        then $dist[i] \leftarrow dist[j] + w(j,i)$

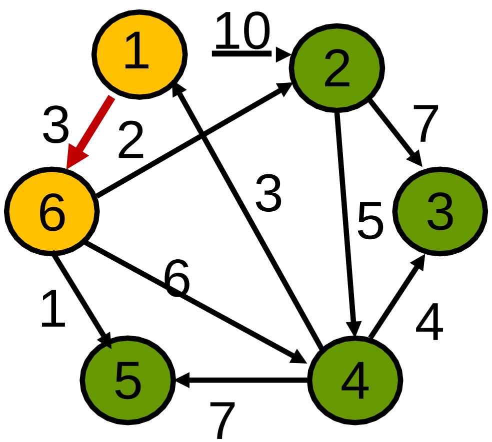
运行实例

- 输入：G=<V,E,W>，源点 1，V={1,2,3,4,5,6}
- S={1},
- $dist$[1]=0
- $dist$[2]=10
- $dist$[3]=∞
- $dist$[4]=∞
- $dist$[5]=∞
- $dist$[6]=3



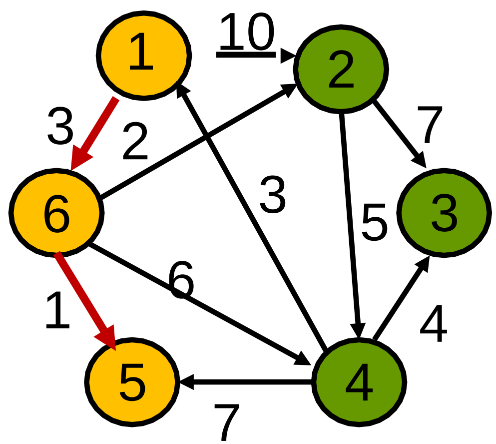
运行实例

- $S=\{1,6\}$
- $dist\,[1] = 0$
- $dist\,[2] = 5$
- $dist\,[3] = \infty$
- $dist\,[4] = 9$
- $dist\,[5] = 4$
- $dist\,[6] = 3$



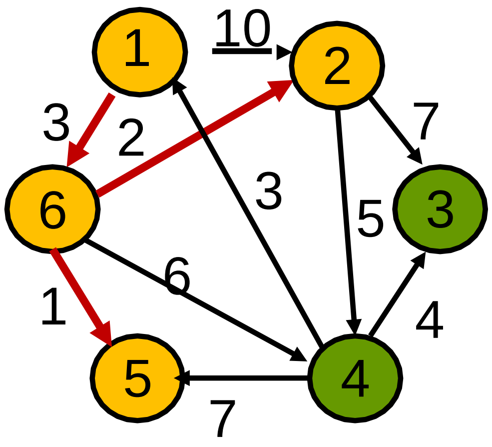
运行实例

- S={1,6,5}
- *dist* [1]=0
- *dist* [2]=5
- *dist* [3]=∞
- *dist* [4]=9
- *dist* [5]=4
- *dist* [6]=3



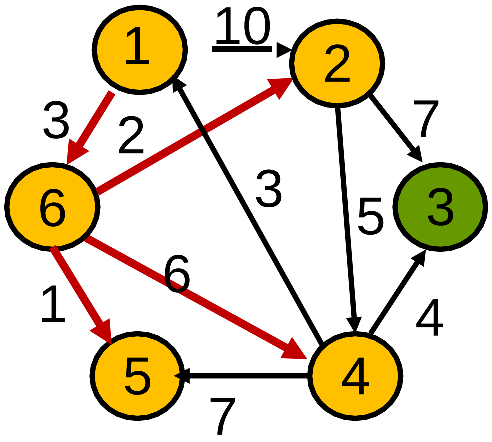
运行实例

- $S=\{1,6,5,2\}$
- $dist[1]=0$
- $dist[2]=5$
- $dist[3]=12$
- $dist[4]=9$
- $dist[5]=4$
- $dist[6]=3$



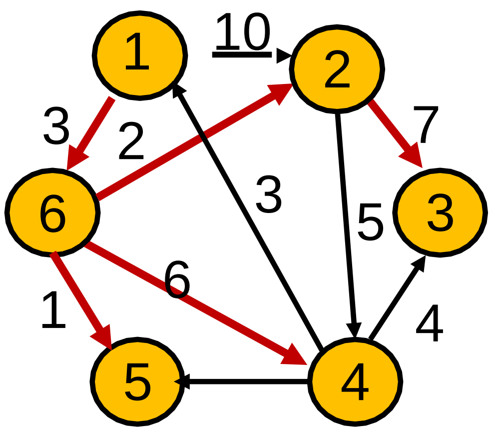
运行实例

- S={1,6,5,2,4}
- *dist*[1]=0
- *dist*[6]=3
- *dist*[5]=4
- *dist*[2]=5
- *dist*[4]=9
- *dist*[3]=12



运行实例

- S={1,6,5,2,4,3}
- *dist* [1]=0
- *dist* [2]=5
- *dist* [3]=12
- *dist* [4]=9
- *dist* [5]=4
- *dist* [6]=3



- *short*[1]=0, *short*[2]=5, *short*[3]=12,
- *short*[4]=9, *short*[5]=4, *short*[6]=3

Dijkstra算法的正确性

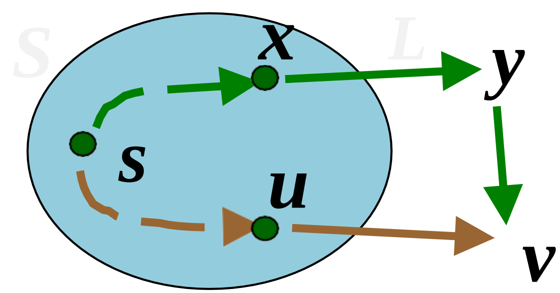
- 归纳证明思路
- 命题：当算法进行到第 k 步时，对于S 中每个结点 i，

$$dist\,[i] = short\,[i]$$

- 归纳基础
  - $k = 1, S = \{s\}, dist\,[s] = short\,[s] = 0$
- 归纳步骤
  - 证明：假设命题对 k 为真，则对 k+1 命题也为真

归纳步骤证明

- 假设命题对*k*为真，考虑 *k*+1步算法
- 选择顶点*v* (边<*u*,*v*>)，需要证明 *dist*[*v*]=*short*[*v*]
- 若存在另一条 *s*-*v* 路径 *L* (绿色)，最后一次出 *S* 的顶点为 *x*, 经过 *V*–*S* 的第一个顶点 *y*，再由 *y* 经过一段在*V*– *S* 中的路径到达 *v*



- 在 *k*+1步算法选择顶点 *v*，而不是 *y* ,

$$dist\,[v] \le dist\,[y]$$

- 令 *y* 到 *v* 的路径长度为 *d*(*y*,*v*)

$$dist\,[y]+d(y,v) \le L$$

- 于是 *dist* [*v*]≤*L*，即 *dist*[*v*]= *short* [*v*]

时间复杂度

- 时间复杂度：$O(nm)$ 算法进行n-1步
- 每步挑选1个具有最小$dist$函数值的结点进入S
    - 需要 $O(m)$时间

贪心法小结

- 贪心法正确性证明方法：
  - 直接计算优化函数，贪心法的解恰好取得最优值
  - 数学归纳法（对算法步数或者问题规模归纳）
  - 交换论证
- 证明贪心策略不对：举反例
- 对于某些不能保证对所有的实例都得到最优解的贪心算法（近似算法），可做参数化分析或者误差分析
- 贪心法的优势：算法简单，时间和空间复杂性低
- 几个著名的贪心算法
  - 最小生成树的Prim算法
  - 最小生成树的Kruskal算法
  - 单源最短路的Dijkstra算法