应用：最优前缀码

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 *a*---001, *b*---00, *c*---010, *d*---01

实例 0100001: 解码1. 01, 00, 001 *d, b, a*

解码2. 010, 00, 01 *c, b, d*

前缀码的存储采用二叉树的结构，每个字符作为树叶，每个前缀码看作根到树叶的路径

输入：n个字符 $x_1, x_2, \ldots, x_n$,

每个字符传输概率 $f(x_i)$, $i=1, 2, \ldots, n$.

求：前缀码，使得平均传输一个字符的位数达到最小

算法：Huffman树得到最优解

Huffman算法

算法 Huffman(C)

1. $n \leftarrow |C|$;
2. $Q \leftarrow C$; // 按频率递增构成队列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 // 取出Q中最小元作为z的左儿子
6. $z.right \leftarrow Q$中最小元 // 取出Q中最小元作为z的右儿子
7. $f[z] \leftarrow f[x]+f[y]$
8. Insert(Q,z); // 将 z 插入Q , $O(\log n)$
9. Return Q

时间：$O(n\log n)$

实例

例如 ***a*:45, *b*:13; *c*:12;**

***d*:16; *e*:9; *f*:5**

编码:

***f*--0000, *e*--0001,**

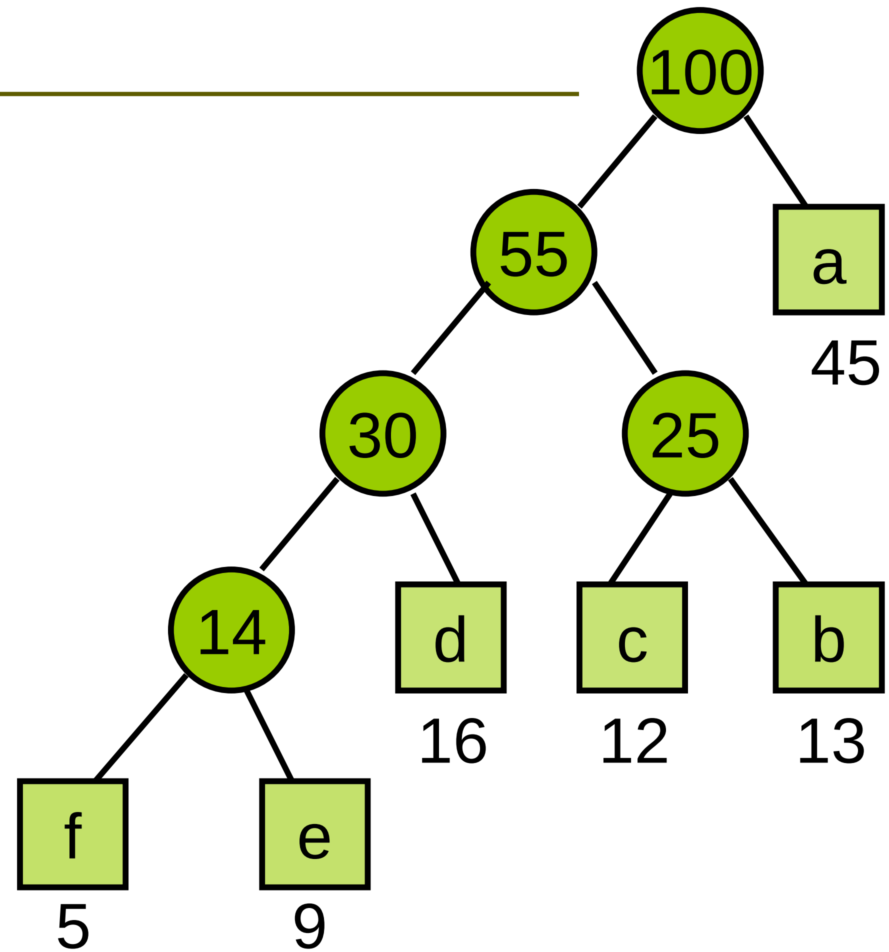
***d*--001, *c*--010,**

***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



证明：引理1

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 的码字等长，且仅在最后一位不同.

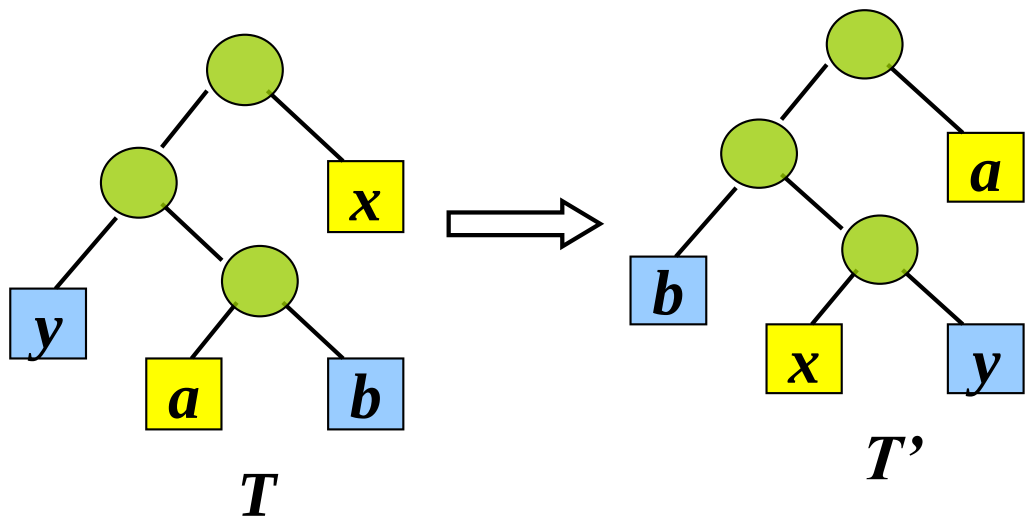
$T \rightarrow T'$

$f[x] \le f[a]$

$f[y] \le f[b]$

a与 x 交换

b与 y 交换



则T与T'的权之差为

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

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

$$B(T)=B(T')+f(x)+f(y).$$

证 $\forall c \in C-\{x, y\}$,有 $d_T(c) = d_{T'}(c) \Rightarrow f(c)d_T(c)=f(c)d_{T'}(c)$

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

$$B(T) = \sum_{i \in T} f(i) d_T(i) = \sum_{i \in T, i \neq x, y} f(i) d_T(i) + f(x) d_T(x) + f(y) d_T(y)$$

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

证明：归纳法

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 $n=2$，字符集 $C=\{x_1,x_2\}$，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$C=\{x_1, x_2, ..., x_{k+1}\}$, 其中$x_1, x_2 \in C$是频率最小的两个字符. 令

$$C' = (C - \{x_1,x_2\}) \cup \{z\}, \quad f(z)=f(x)+f(y)$$

根据归纳假设，Huffman算法得到一棵关于字符集C'、频率$f(z)$和$f(x_i)$（$i=3,4,...,k+1$）的最优前缀码的二叉树T'.

证明：归纳法(续)

把x_1和 x_2作为 z 的儿子附加到T'上，得到树T，那么T是关于字符集$C=(C'-\{z\})\cup\{x_1,x_2\}$ 的最优前缀码的二叉树.

如若不然，假如T 不是关于C 的最优二元前缀码对应的二叉树，那么存在更优的树. 根据引理1，存在最优树T^*，其最深层树叶是 x_1,x_2，且 $B(T^*)< B(T)$.

去掉T^*中的x_1和x_2，根据引理2，所得二叉树$T^{*\prime}$满足

$$B(T^{*\prime})=B(T^*)-(f(x)+f(y))<B(T)-(f(x)+f(y)) =B(T')$$

与T'是一棵关于C'的最优前缀码的二叉树矛盾.

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

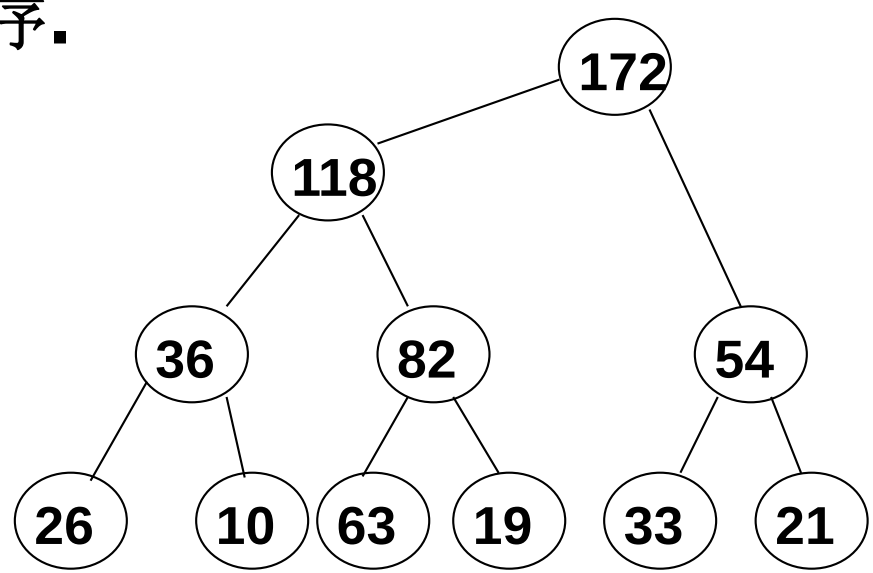
$$S = \{ f_1, \ldots , f_n\}.$$

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的文件．找到一个比较次数最少的归并次序.

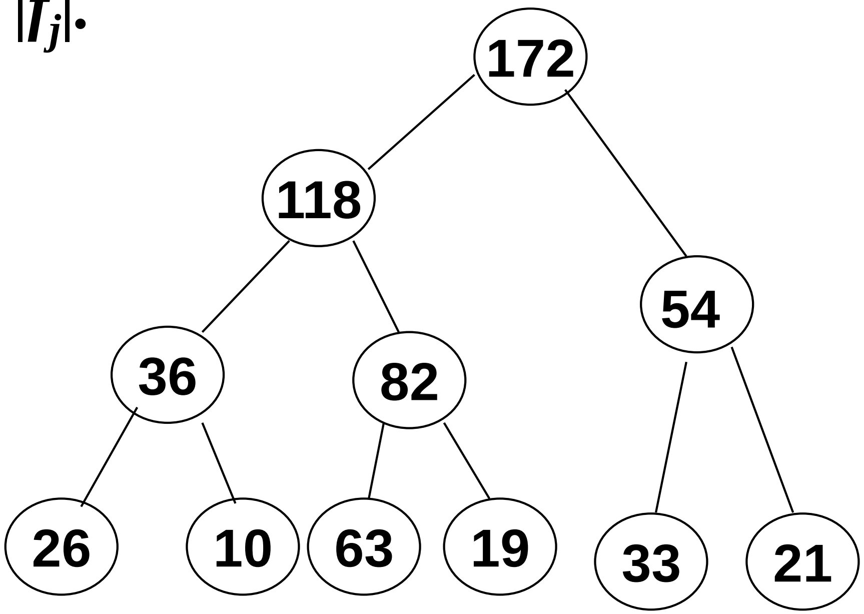
归并过程对应于二叉树

实例： 26,10,63,19,33,21



归并的代价

归并树叶 f_i 和 f_j, 代价是 $|f_i| + |f_j|$.
$C(T)$ 是树的内结点的权之和



实例:

$C(T) = 36+82+54+118+172$

$= (26+10+63+19)\times 3$

$+ (33+21)\times 2$

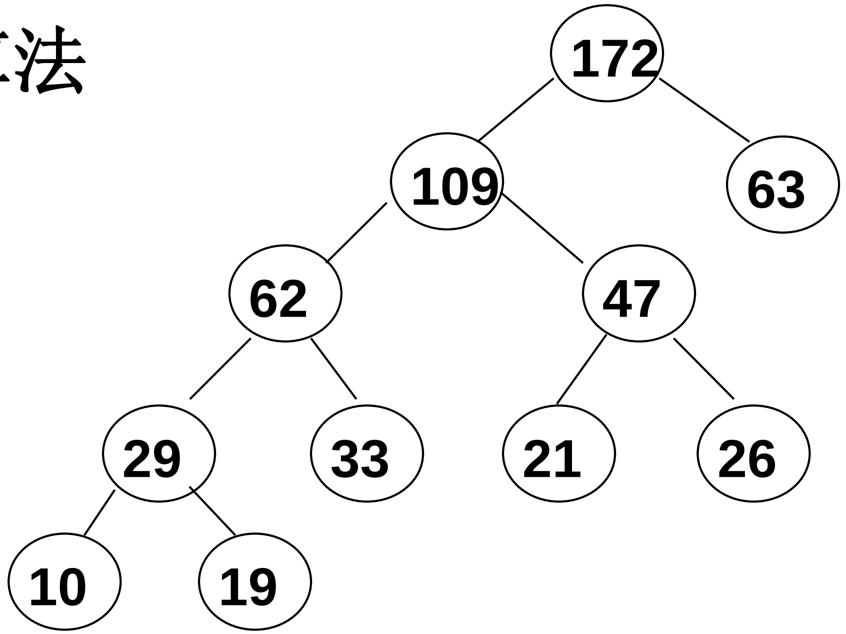
$= 462$

$$\sum_{k=1}^{n} | f_k | \times depth(f_k) - (n-1)$$

更好的归并方法

Huffman树的算法

时间$O(n\log n)$



$$(10+19)\times 4+(33+21+26)\times 3+63$$
$$=116+240+63=419$$

4.4.2 最小生成树

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

命题4.1 设 G 是 n 阶连通图，那么

(1) T 是 G 的生成树当且仅当 T 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(回路).

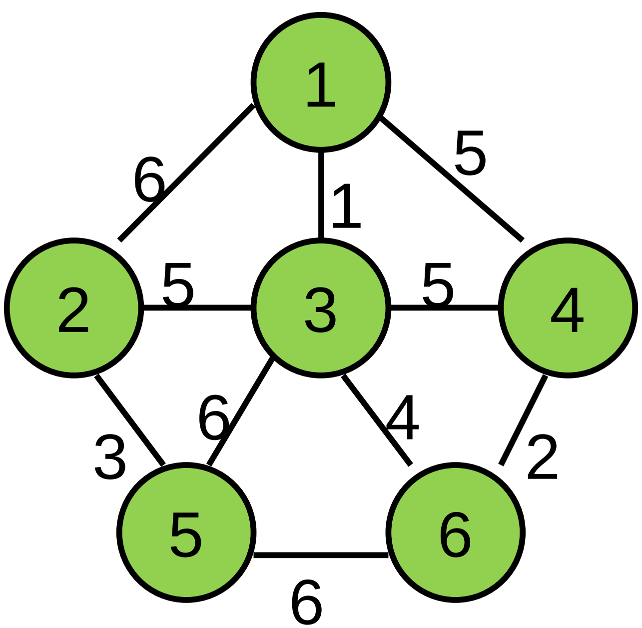
问题：给定连通带权图G，求G的一棵最小生成树.

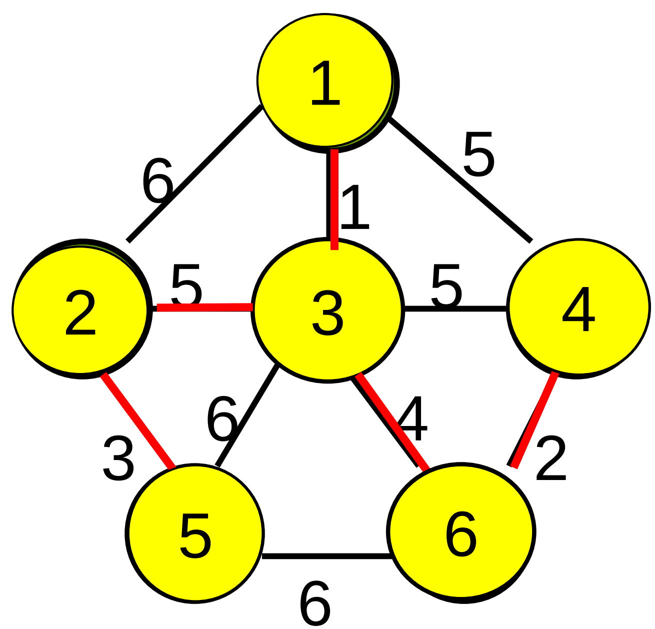
算法：**Prim算法**和**Kruskal算法**

Prim算法

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





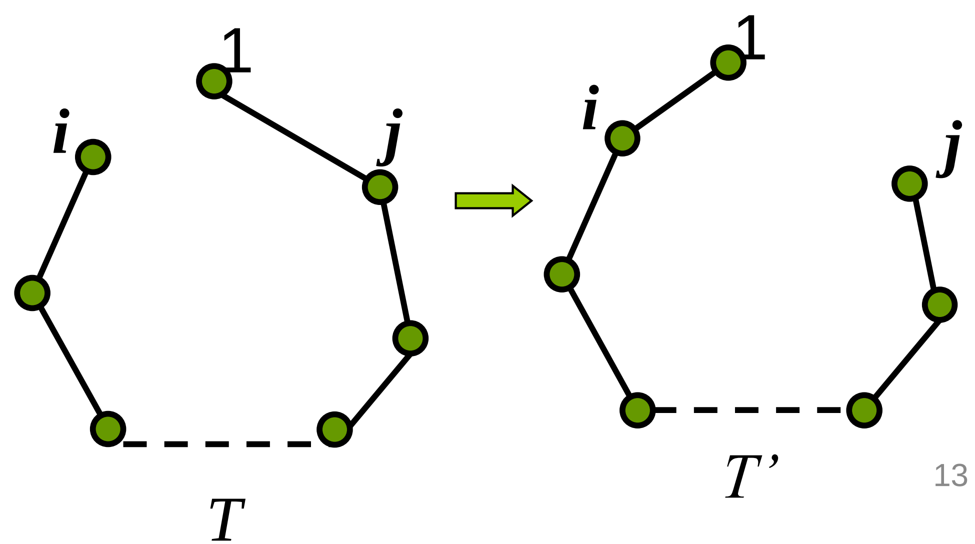
正确性证明

对步数归纳

定理：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.

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的另一条边为(1,j)，令 $T' = (T-\{(1,j)\})\cup\{(1,i)\}$，则 T'也是生成树，且 $W(T') \le W(T)$.



正确性证明(续)

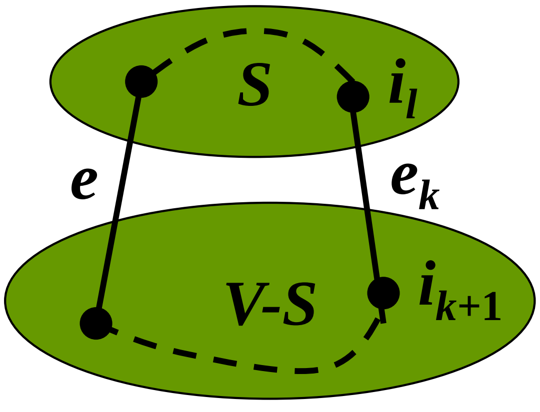
归纳步骤：

假设算法进行了$k-1$步，生成树的边为$e_1,e_2,\ldots,e_{k-1}$，这些边的 k 个端点构成集合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 步选择了顶点 i_{k+1}, 则 i_{k+1}到S中顶点的边权最小, 设这条边为 $e_k=(i_{k+1},i_l)$. 假设T不含有e_k，则将 e_k 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

$$T^*=(T-\{e\})\cup\{e_k\},$$

则T^*是G 的一棵生成树，包含$e_1,e_2,\ldots,e_k$, $W(T^*)\le W(T)$.



算法时间：$T(n)=O(n^2)$

Kruskal算法

算法4.6 Kruskal

输入：连通图*G*　　　　// 顶点数*n*，边数*m*

输出：*G*的最小生成树

1. 按权从小到大排序*G*中的边，使得$E=\{e_1, e_2, \ldots, e_m\}$

2. $T \leftarrow \varnothing$

3．repeat

4.　　$e \leftarrow E$中的最短边

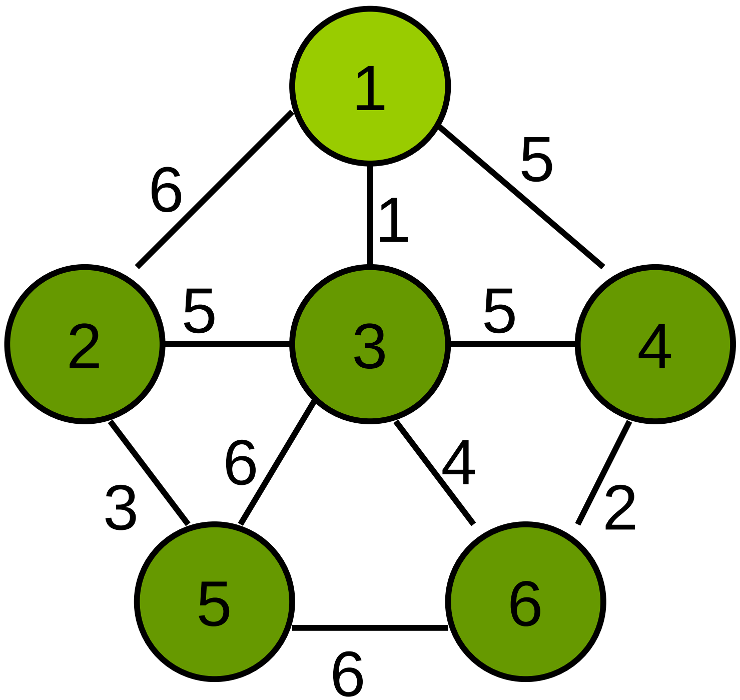
5．　　if　*e*的两端点不在同一个连通分支

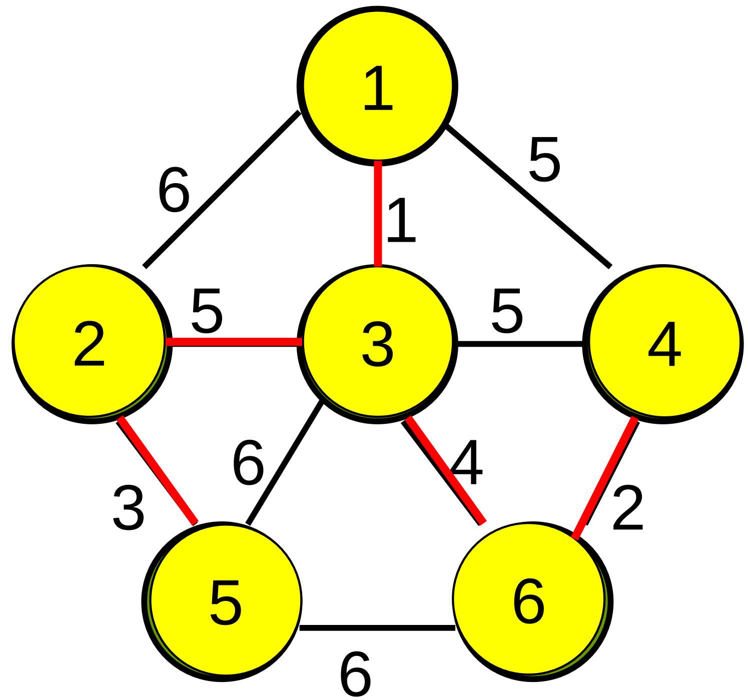
6.　　then　$T \leftarrow T \cup \{e\}$

7.　　$E \leftarrow E - \{e\}$

8. until *T*包含了$n-1$条边

实例





Kruskal算法正确性证明

命题：对于任意 $n>1$, 算法对 n 阶图得到一棵最小生成树.

证明 $n=2$, 只有一条边，命题显然为真.
假设对于n个顶点的图算法正确，考虑 $n+1$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'. 根据归纳假设，由算法存在G'的最小生成树T'.令$T=T'\cup\{e\}$, 则T 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^*中短接 e 得到G'的生成树$T^*-\{e\}$, 且

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

与T'的最优性矛盾.

算法的实现与时间复杂度

数据结构：

建立FIND数组，FIND[i] 是结点 i 的连通分支标记.

(1) 初始FIND[i]=i.

(2) 两个连通分支合并，则将较小分支结点的FIND值更新为较大分支的标记

时间复杂度：

(1) 每个结点至多更新log$n$次，建立和更新FIND数组的总时间为$O(n\log n)$

(2) 算法时间为

$$O(m\log m) + O(n\log n) + O(m) = O(m\log n)$$

边排序　　FIND数组　　其他

4.4.3 单源最短路径

给定带权有向网络$G=(V,E,W)$,每条边$e=\langle i,j\rangle$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.

Dijkstra算法：

$x\in S \Leftrightarrow x\in V$ 且从 s 到 x 的最短路径长度已知

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 的相对于S 的最短路径的长度

$short[u]$：从 s 到 u 的最短路径的长

$dist[u] \geq short[u]$

Dijkstra算法

算法 Dijkstra

1. $S \leftarrow \{s\}$
2. $dist[s] \leftarrow 0$
3. for $i \in V - \{s\}$ do
4. $\quad dist[i] \leftarrow w(s,i)$ // 如果 s 到 i 没有边, $w(s,i)=\infty$
5. while $V - S \neq \varnothing$ do
6. $\quad$ 从$V-S$中取出具有相对S的最短路径的顶点 j
7. $\quad S \leftarrow S \cup \{j\}$;
8. $\quad$ for $i \in V - S$ do
9. $\quad\quad$ if $dist[j]+w(j,i) < dist[i]$
10. $\quad\quad$ then $dist[i] \leftarrow dist[j]+w(j,i)$ // 更新$dist[i]$

实例

输入：$G=<V,E,W>$，源点 1

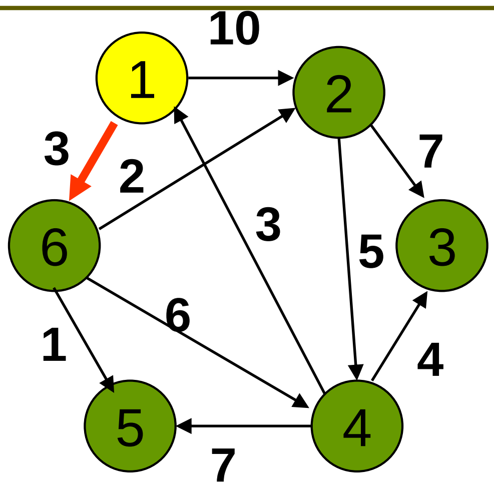
$V=\{1, 2, 3, 4, 5, 6\}$

$S=\{1\}$,

$dist[1]=0$

$dist[2]=10$, $dist[6]=3$

$dist[3]=dist[4]=dist[5]=\infty$

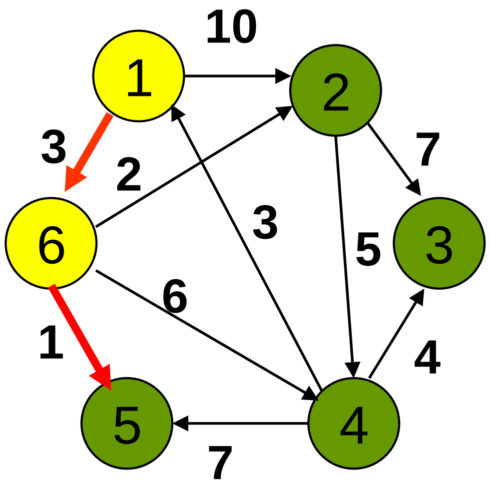


$S=\{1,6\}$,

$dist[1]=0$, $dist[6]=3$

$dist[2]=5$, $dist[4]=9$, $dist[5]=4$

$dist[3]=\infty$



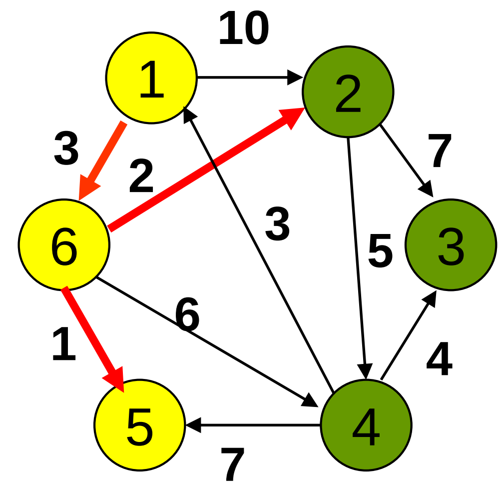
实例（续）

S={1,6,5},

dist[1]=0, *dist*[6]=3, *dist*[5]=4

dist[2]=5, *dist*[4]=9,

dist[3]=∞



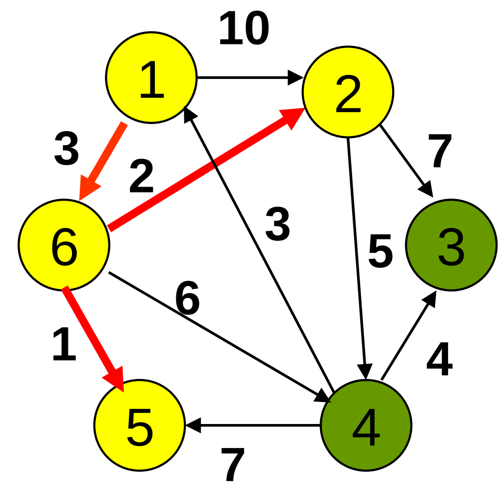
S={1,6,5,2},

dist[1]=0, *dist*[6]=3, *dist*[5]=4

dist[2]=5

dist[3]=12

dist[4]=9



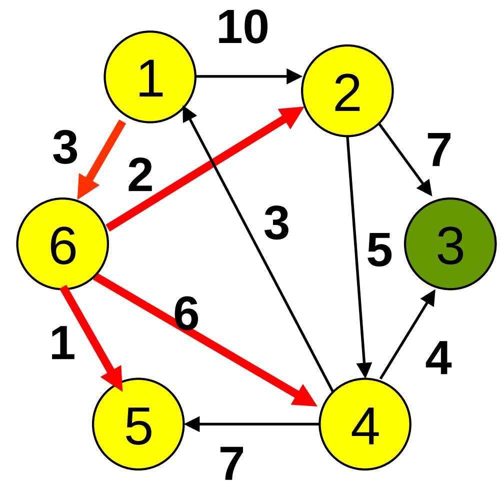
实 例（续）

S={1,6,5,2,4},

dist[1]=0, *dist*[6]=3, *dist*[5]=4

dist[2]=5, *dist*[4]=9,

dist[3]=12

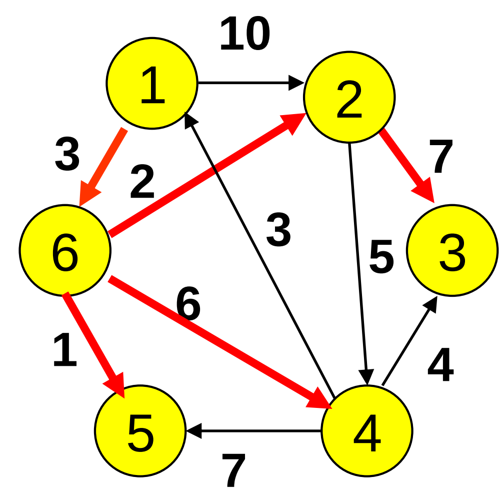


S={1,6,5,2,4,3},

dist[1]=0, *dist*[6]=3, *dist*[5]=4

dist[2]=5, *dist*[4]=9

dist[3]=12



解：

short[1]=0, short[2]=5,

short[3]=12, short[4]=9,

short[5]=4, short[6]=3.

算法正确性证明

命题：当算法进行到第 k 步时，对于S 中每个结点 i，

$$dist[i] = short[i]$$

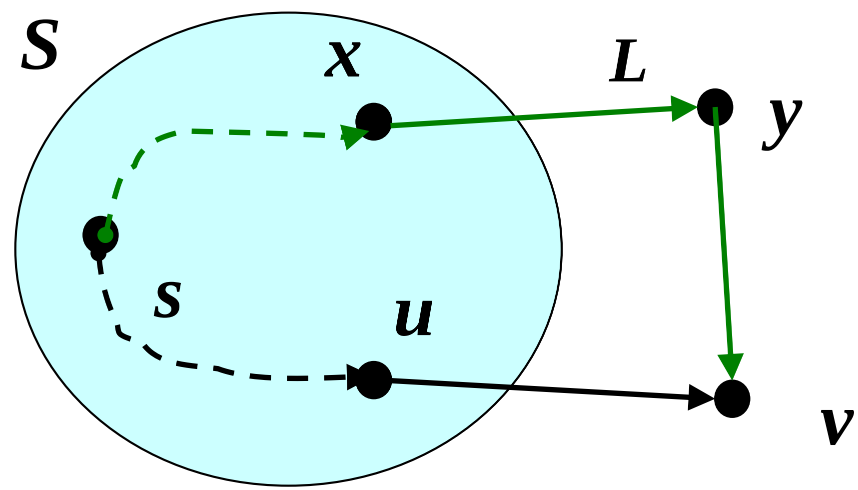
归纳基础　k=1, S={s}, $dist[s]=short[s]=0$, 命题为真.

归纳步骤　假设命题对于k 为真. 考虑 k+1步, 选择顶点v (边(u,v)). 假若存在另一条 s-v 路径 L (绿色)，最后一次出S 的顶点为 x, 在这次从S 中出来后经过V–S 的第一个顶点为 y.

$$dist[v] \le dist[y] \quad //v\text{先被选}$$
$$\le dist[y]+d(y,v) \le L$$

$$dist[v]=short[v]$$

时间复杂度 $T(n)=O(n^2)$



贪心法小结

1. 适用于组合优化问题. 求解过程是多步判断。判断的依据是局部最优策略，使目标值达到最大(或最小)，与前面的子问题计算结果无关。
2. 局部最优策略的选择是算法正确性的关键。
3. 正确性证明方法：数学归纳法、交换论证。
   - 使用数学归纳法主要通过对算法步数或者问题规模进行归纳。
   - 如果要证明贪心策略是错误的，只需举出反例。
4. 自顶向下求解，通过选择将问题归约为小的子问题。
5. 如果贪心法得不到最优解，可以对问题的输入进行分析或者估计算法的近似比。
6. 如果对原始数据排序之后，贪心法往往是一轮处理，时间复杂度和空间复杂度低。