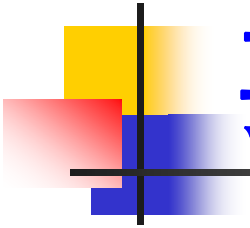
算法分析与设计

Analysis and Design of Algorithm

Lesson 11



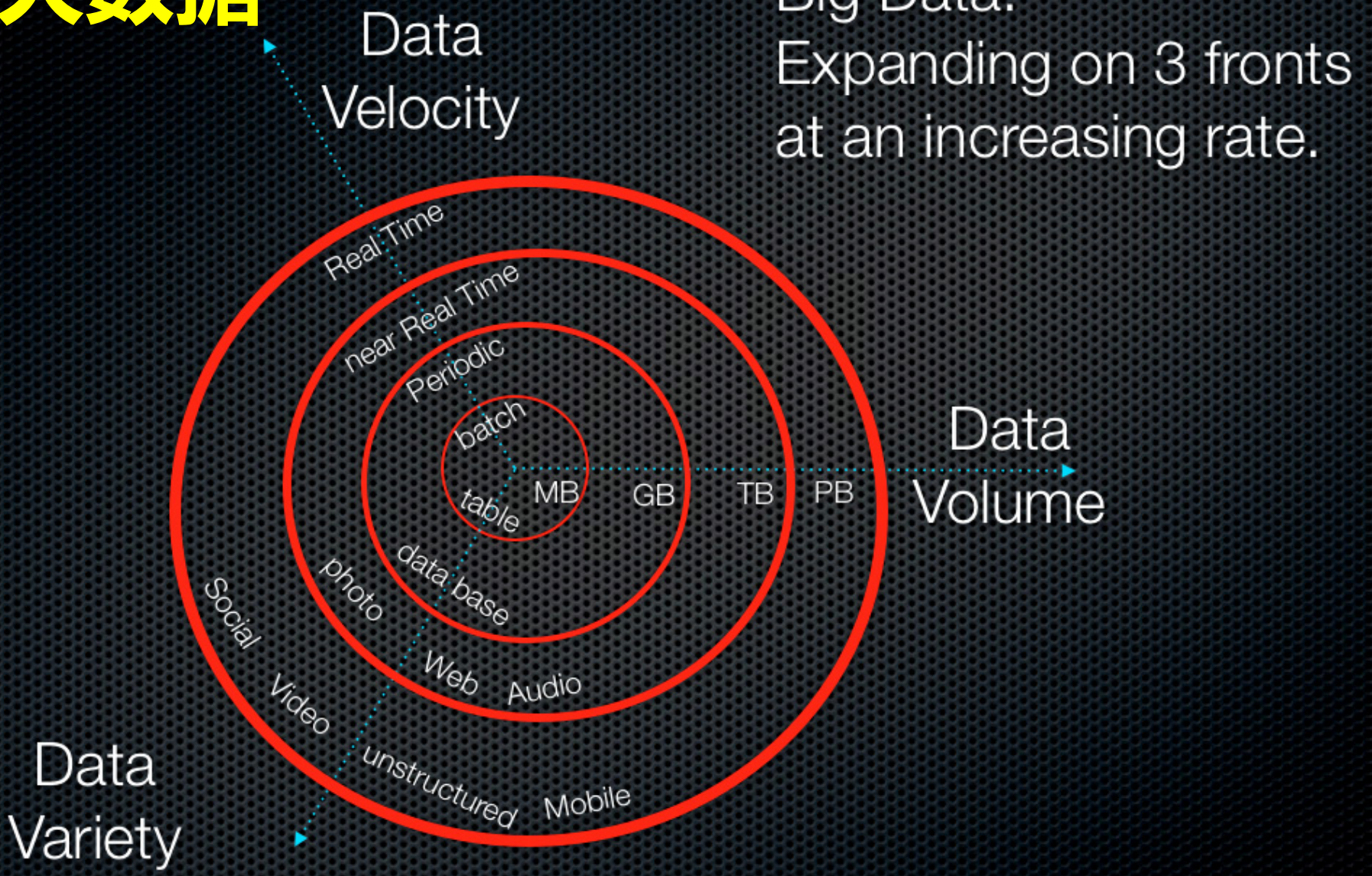
要点回顾

- 贪心算法的基本思想
  - 求解最优化问题的算法包含一系列步骤
  - 每一步都有一组选择
  - 作出在当前看来最好的选择
  - 希望通过作出局部最优选择达到全局最优选择
  - 贪心算法不一定总产生最优解
  - 贪心算法是否产生优化解，需严格证明
- 贪心算法产生最优解的条件
  - 最优子结构性质
  - 贪心选择性性质

要点回顾(cont.)

- 贪心算法正确性归纳证明
  - 叙述一个有关自然数*n*的命题，该命题断定该贪心策略的执行最终将导致最优解。其中自然数*n*可以代表步数或问题规模
  - 证明命题对所有的自然数为真
    - 归纳基础（从最小实例规模开始）
    - 归纳步骤（第一或第二数学归纳法）
- 几个实例：
  - 背包问题
  - 活动选择问题
  - 最优装载（0-1背包问题的子问题）

大数据


Big Data:
Expanding on 3 fronts
at an increasing rate.
Data
Velocity
Real Time
near Real Time
Periodic
batch
table
MB
GB
TB
PB
Data
Volume
data base
photo
Social
Video
Web
Audio
unstructured
Mobile
Data
Variety

www.wipro.com

BIG DATA

Big Data is data that is too large, complex and dynamic for any conventional data tools to capture, store, manage and analyze.

The right use of Big Data allows analysts to spot trends and gives niche insights that help create value and innovation much faster than conventional methods.

The "three V's", i.e the Volume, Variety and Velocity of the data coming in is what creates the challenge.

VOLUME

Amount of Big Data stored across the world (in petabytes)

VARIETY

PEOPLE TO PEOPLE
NETIZENS, VIRTUAL COMMUNITIES, SOCIAL NETWORKS, WEB LOGS...

PEOPLE TO MACHINE
ARCHIVES, MEDICAL DEVICES, DIGITAL TV, E-COMMERCE, SMART CARDS, BANK CARDS, COMPUTERS, MOBILES...

MACHINE TO MACHINE
SENSORS, GPS DEVICES, BAR CODE SCANNERS, SURVEILLANCE CAMERAS, SCIENTIFIC RESEARCH...

VELOCITY

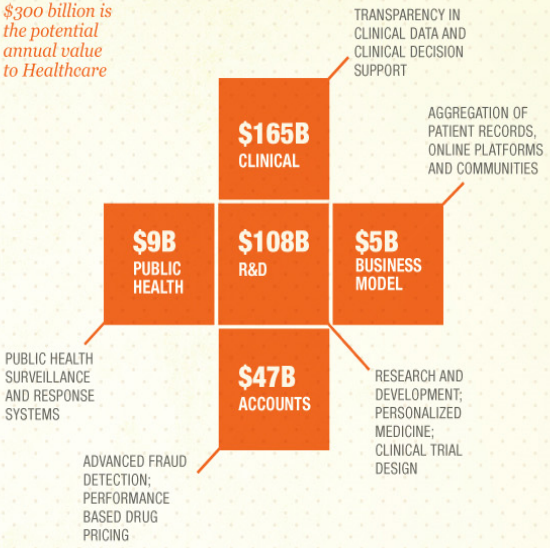
2.9 MILLION
EMAILS SENT EVERY SECOND

20 HOURS
OF VIDEO UPLOADED EVERY MIN

50 MILLION
TWEETS PER DAY

CASE STUDY - Healthcare

$300 billion is the potential annual value to Healthcare

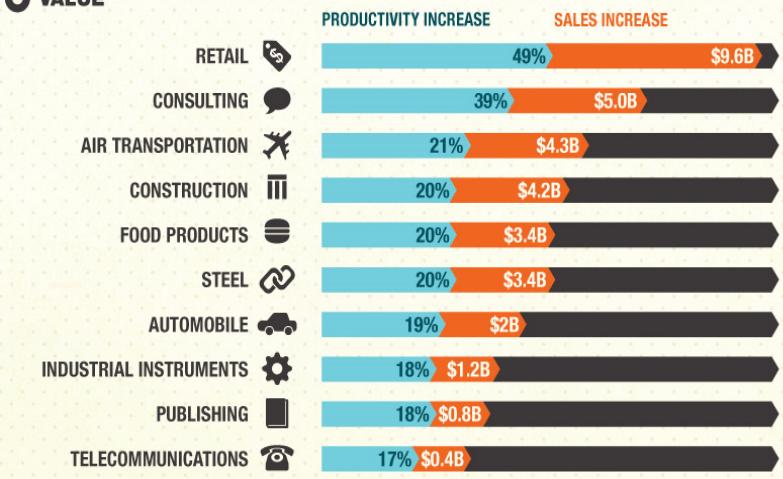


57.6% OF ORGANIZATIONS SURVEYED SAY THAT BIG DATA IS A CHALLENGE

72.7% CONSIDER DRIVING OPERATIONAL EFFICIENCIES TO BE THE BIGGEST BENEFIT OF A BIG DATA STRATEGY

50% SAY THAT BIG DATA HELPS IN BETTER MEETING CONSUMER DEMAND AND FACILITATING GROWTH

VALUE



40% PROJECTED GROWTH IN GLOBAL DATA CREATED PER YEAR

5% PROJECTED GROWTH IN GLOBAL IT SPENDING PER YEAR

The estimated size of the digital universe in 2011 was 1.8 zettabytes. It is predicted that between 2009 and 2020, this will grow 44 fold to 35 zettabytes per year. A well defined data management strategy is essential to successfully utilize Big Data.

Sources - 1 Reaping the Rewards of Big Data - Wipro Report 2 Big Data: The Next Frontier for Innovation, Competition and Productivity - McKinsey Global Institute Report 3 comScore, Radicati Group 4 Measuring the Business Impacts of Effective Data - study by University of Texas, Austin 5 US Department of Labour.

DO BUSINESS BETTER

NYSE:WIT | OVER 130,000 EMPLOYEES | 54 COUNTRIES | CONSULTING | SYSTEM INTEGRATION | OUTSOURCING

WIPRO Applying Thought

Visualization: Seventhinc

二元前缀码及其应用

- 二元前缀码是广泛用于数据文件压缩的编码方法，其使用字符在文件中出现的频率表来建立一个用0,1串表示各个字符的最优表示方式。

Google

Draco 3D compressor

facebook

Zstandard

pied piper

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他字符代码的前缀
- 非前缀码的例子

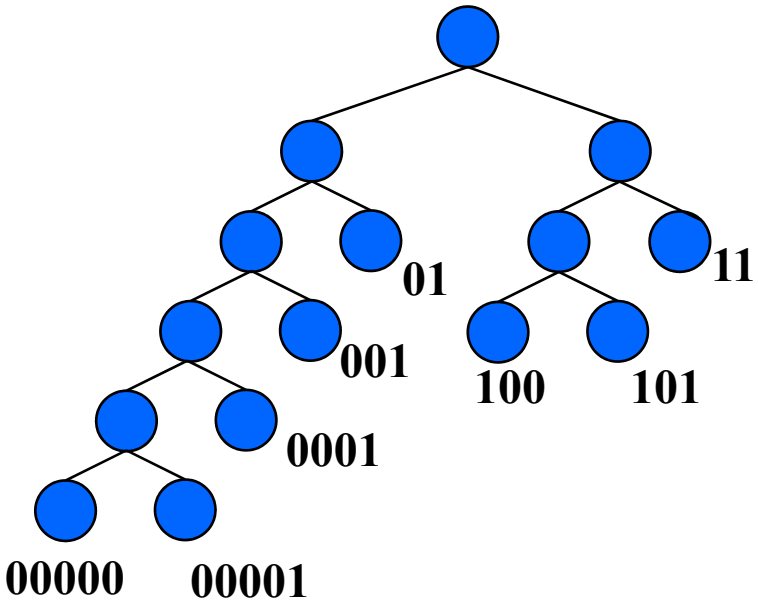
  a: 001, b: 00, c: 010, d: 01
- 解码的歧义，例如字符串0100001
  - 解码1： 01, 00, 001　d, b, a
  - 解码2： 010, 00, 01　c, b, d

前缀码的二叉树表示

- 前缀码：

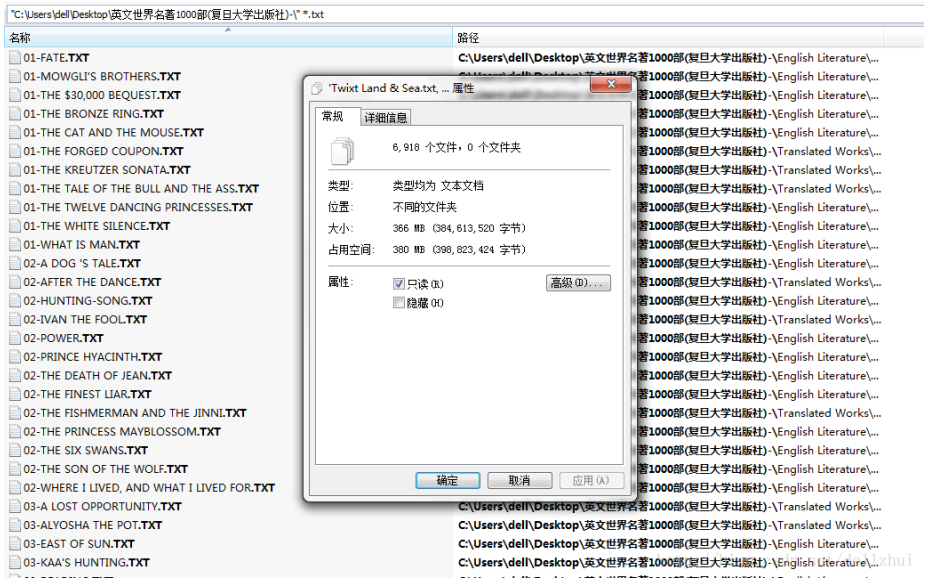
  {00000, 00001, 0001, 001, 01, 100, 101, 11}

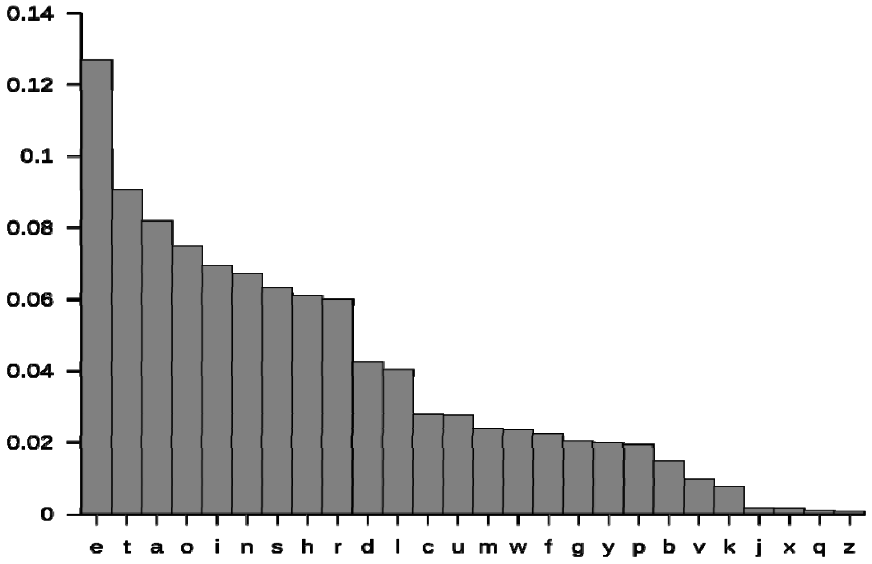
- 构造树：
  - 0-左子树
  - 1-右子树
  - 码对应一片树叶
  - 最大位数为树的层数



一个有趣的大数据统计

- 26英文字母使用频率的大样本分析
  - 样本：英文世界名著（1000部）
  - 样本容量：1.4亿（包含6918个文件，366M）

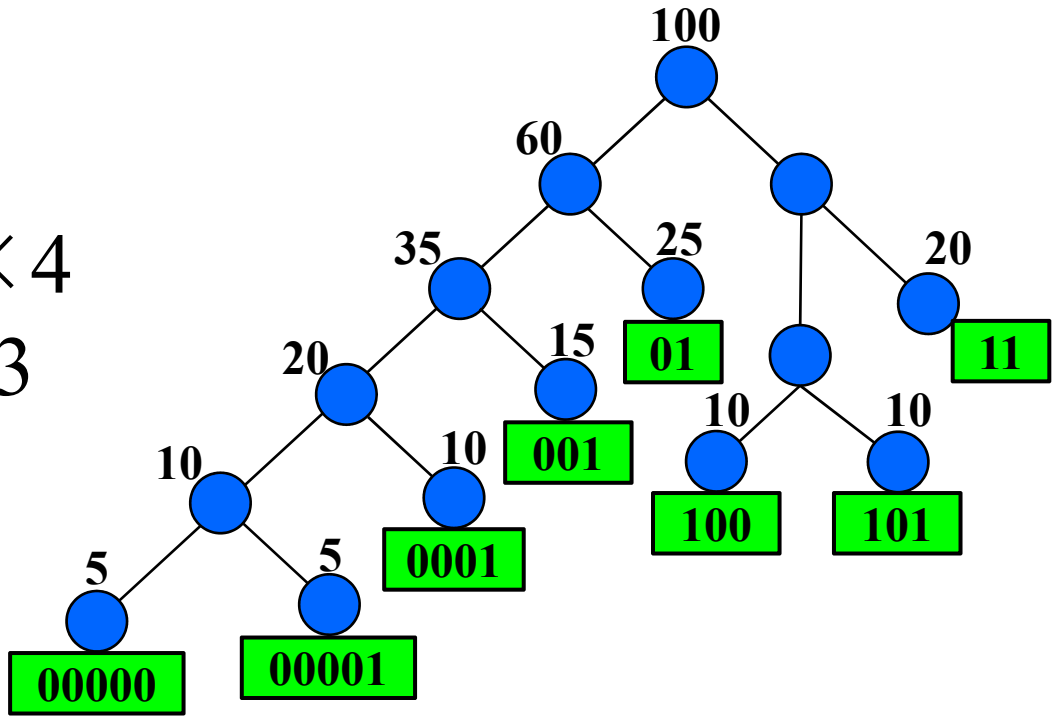




平均传输位数

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

$$\begin{aligned} B=&[(5+5)\times 5+10\times 4\\ &+(15+10+10)\times 3\\ &+(25+20)\times 2]\\ &\div 100\\ =&\ \mathbf{2.85} \end{aligned}$$



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,...,x_n}, $f(x_i)$, i=1,2,...,n
- 输出：Q //队列

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

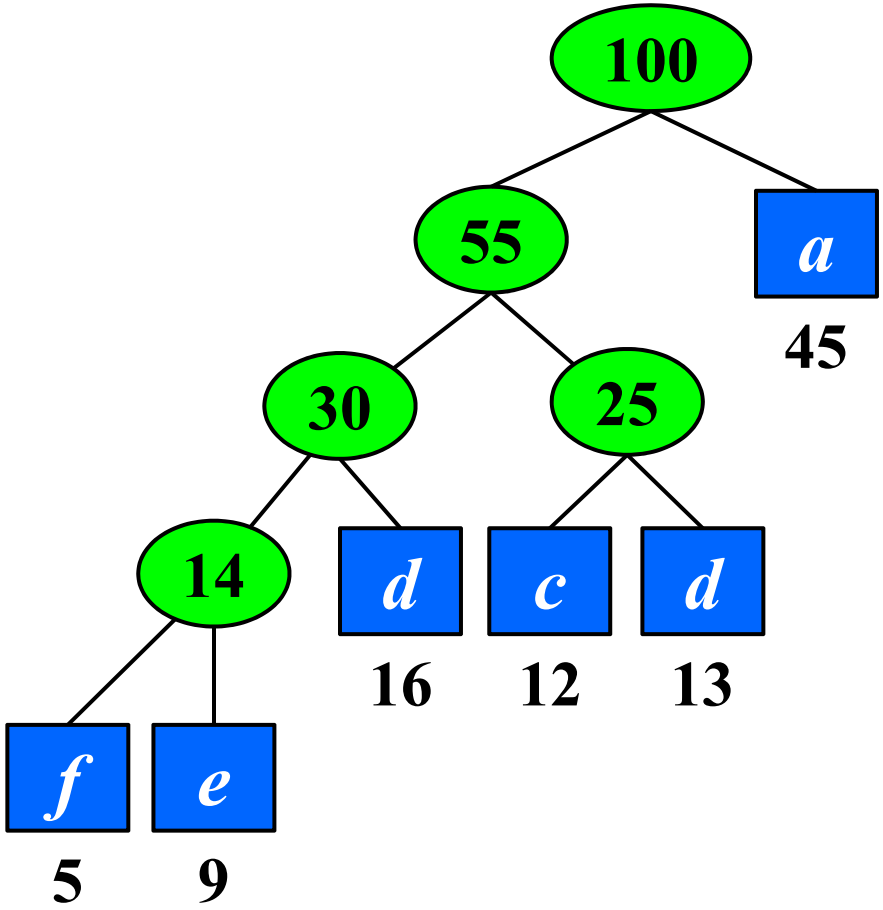
实例

输入： *a*:45, *b*:13, *c*:12, *d*:16, *e*:9, *f*:5

编码：

- $f \rightarrow 0000$
- $e \rightarrow 0001$
- $d \rightarrow 001$
- $c \rightarrow 010$
- $b \rightarrow 011$
- $a \rightarrow 1$

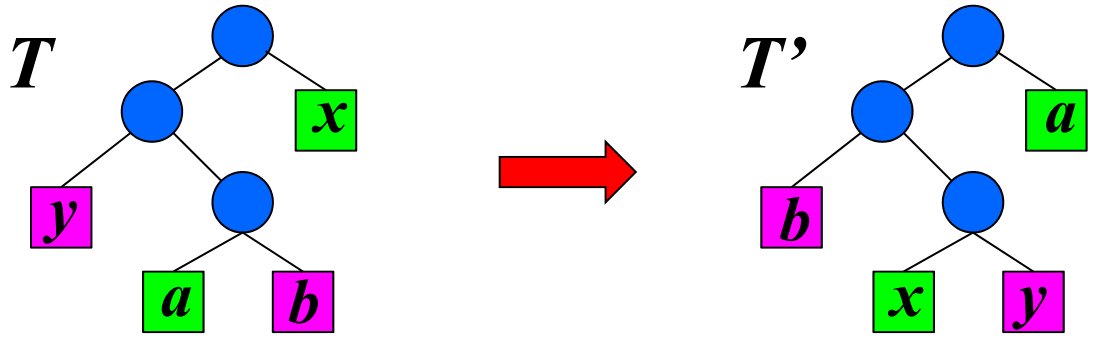
平均位数？



最优前缀码性质

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。

$f(x) \leqslant f(a)$
$f(y) \leqslant f(b)$



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

其中$d_T(i)$为i在T中的层数(i到根的距离)

最优前缀码性质

引理2： 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

$$T'=T-\{x, y\}$$

且令 z 的频率

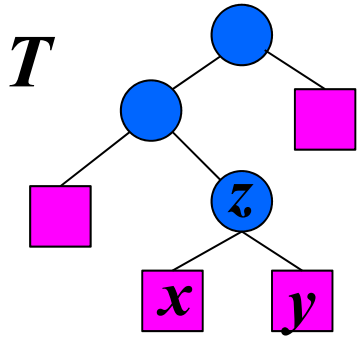
$$f(z)=f(x)+f(y)$$

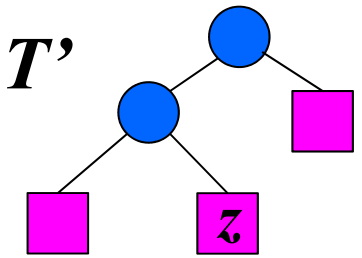
T'是对应二元前缀码

$$C'=(C-\{x, y\})\cup\{z\}$$

的二叉树，那么

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





引理2证明

证明： $\forall c \in C-\{x, y\}$，有

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

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

$$B(T) = \sum_{i \in T} f(i)d_T(i)$$

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

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

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

算法正确性证明思路

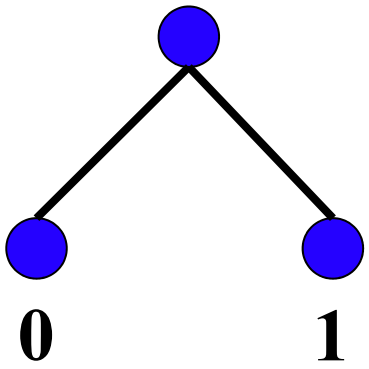
定理 Huffman算法对任意规模为$n(n \geqslant 2)$的字符集C都得到关于C的最优前缀码的二叉树。

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

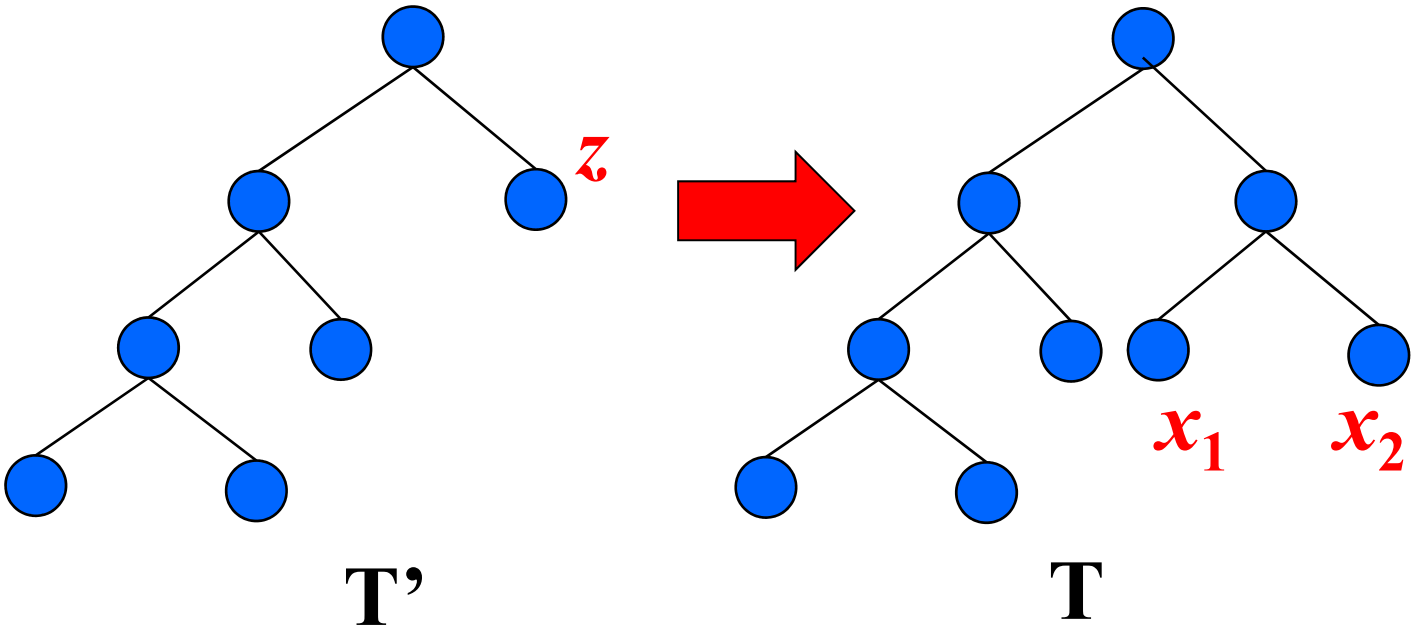
- 假设Huffman算法对于规模为k的字符集都得到最优前缀码。考虑规模为$k+1$的字符集

$$C=\{x_1, x_2, \ldots, x_{k+1}\}$$

- 其中x_1, x_2是C中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，算法可以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'

归纳步骤(cont.)

- 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(cont.)

- 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*中的x_1和x_2，得到T*’。根据引理2

$$B(T^{*'})=B(T^*)-(f(x_1)+f(x_2))$$
$$<B(T)-(f(x_1)+f(x_2))$$
$$=B(T')$$

- 与T’是一棵关于C’的最优前缀码的二叉树矛盾。

图问题中的贪心算法

最小生成树

无向连通带权图

$$G=(V,E,W)$$

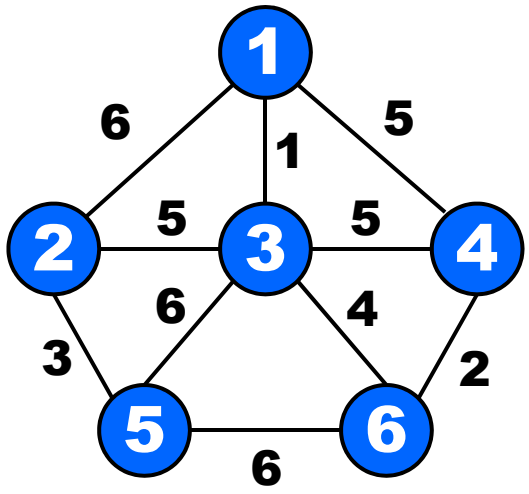
其中$w(e)\in W$是边e的权值。

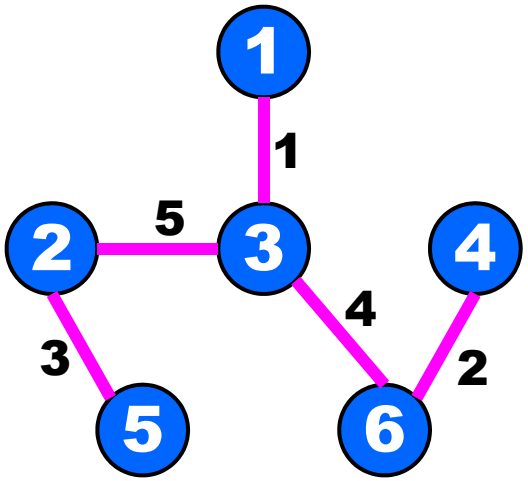
G的一棵生成树T是包含了G所有顶点的树，树中各边的权值之和W(T)称为树的权，具有最小权的生成树称为G的最小生成树。

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}}

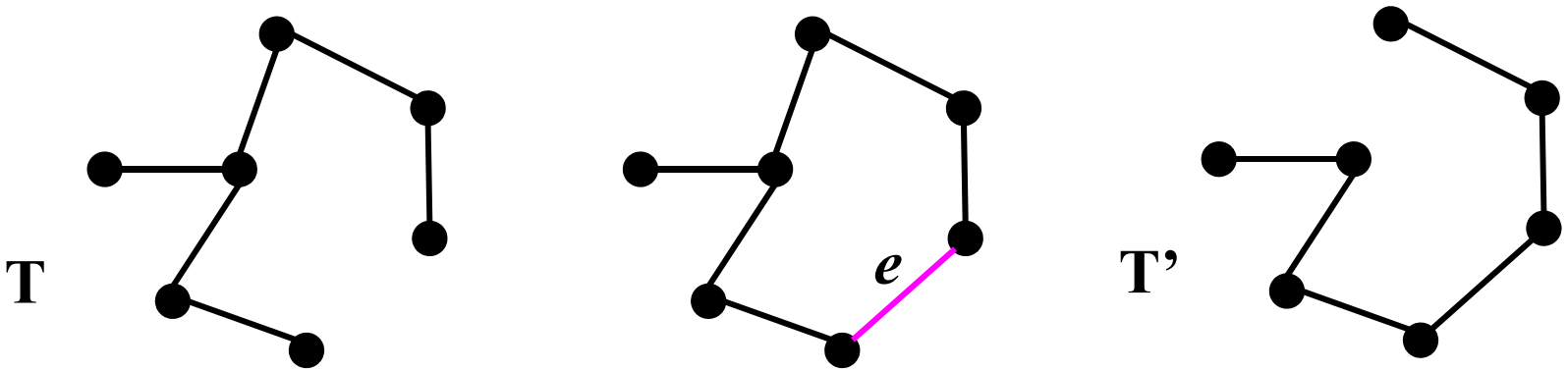




生成树的性质

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'，形成一棵新的生成树T'

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

若$W(e)\leqslant W(e')$，则$W(T')\leqslant W(T)$

求最小生成树

- 问题：
  给定连通带权图G=(V,E,W), W(*e*)∈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！

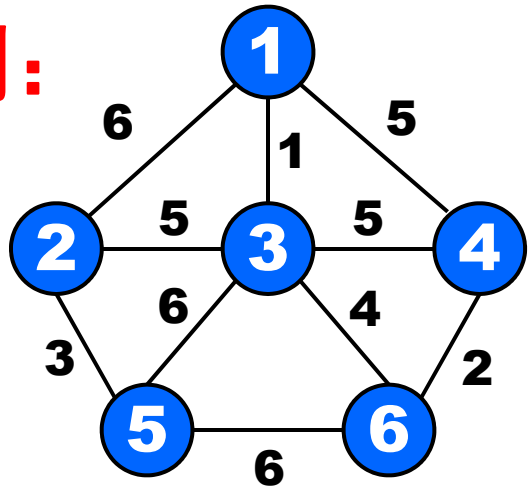
Prim算法

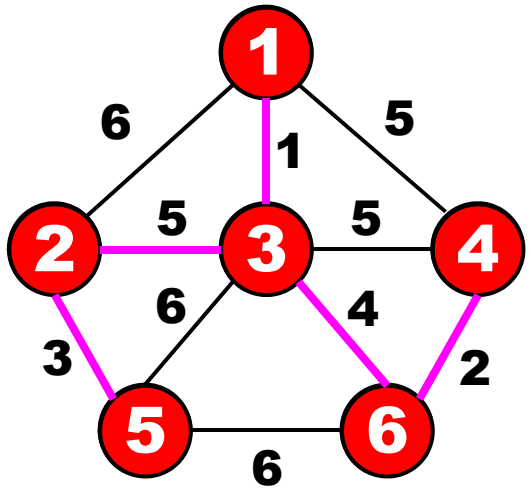
- 设计思想
  - 输入：图G=(V,E,W), V={1,2,...,*n*}
  - 输出：最小生成树T
  - 步骤：
    - 初始S={1};
    - 选择连接S与V-S集合的最短边*e*={*i*, *j*}，其中*i*∈S, *j*∈V-S。将*e*加入树T，*j*加入S；
    - 继续执行上述过程，直到S=V为止。

Prim算法伪码

- 算法Prim(G,E,W)
  1. S ← {1}
  2. while V–S≠ ∅ do
  3. 从V-S中选择*j*使得*j*到S中顶点的边权最小
  4. S ← S∪{ *j* }

- 实例：





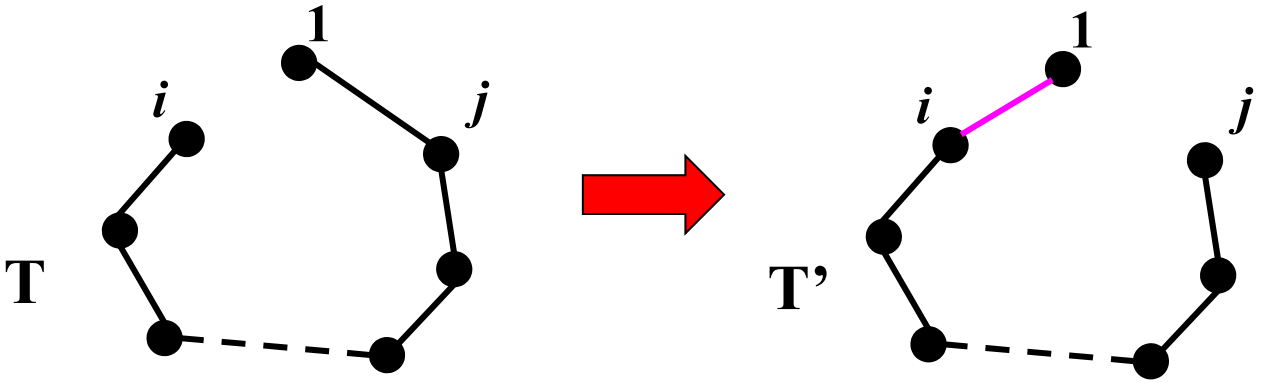
正确性证明：归纳法

- **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$e=\{1,i\}$。

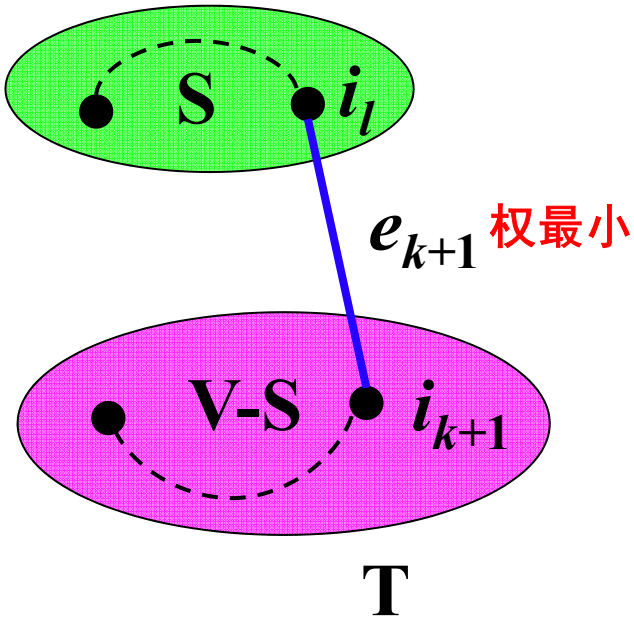
证 设T为一棵最小生成树，假设T不包含$\{1, i\}$，则T∪$\{\{1, i\}\}$含有一条回路，回路中关联1的另一条边 $\{1, j\}$ 。用$\{1, i\}$替换$\{1, j\}$得到树T’，则T’也是生成树，且W(T’)≤W(T)。



归纳步骤

假设算法进行了k步，生成树的边为$e_1,e_2,...,e_k$，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。



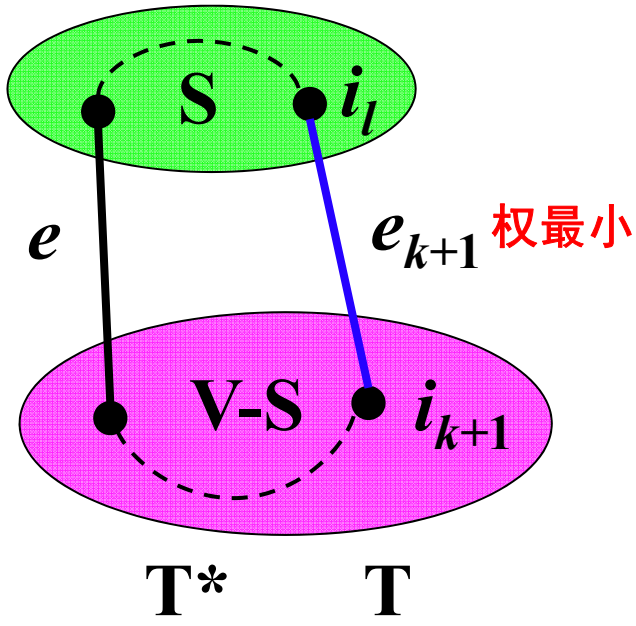
归纳步骤(cont.)

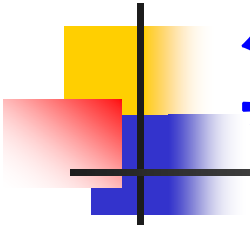
若e_{k+1} ∉T，则将加到T形成一条回路。这条回路有另一条连接S与V-S中顶点的边e

令T*=(T-{e})∪{e_{k+1}}，则T*是G的一棵生成树，包含$e_1,e_2,\ldots,e_{k+1}$，且

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

算法到第k+1步仍得到最小生成树。





算法复杂度

- 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,...,*n*}
  - 输出：最小生成树T
  - 步骤：
    - 按照长度从小到大对边进行排序；
    - 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*加入到树T，否则跳过*e*。直到选择了*n*-1条边为止。

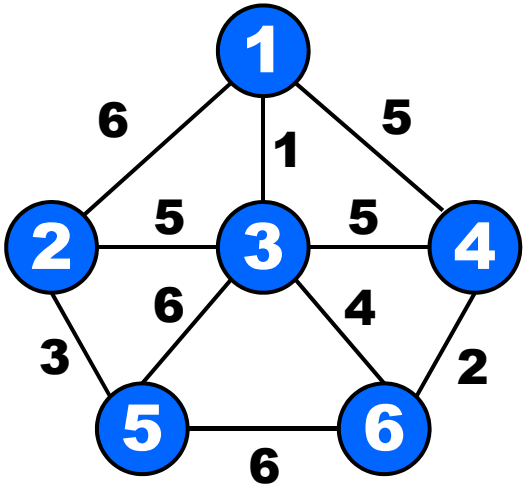
Kruskal算法伪码

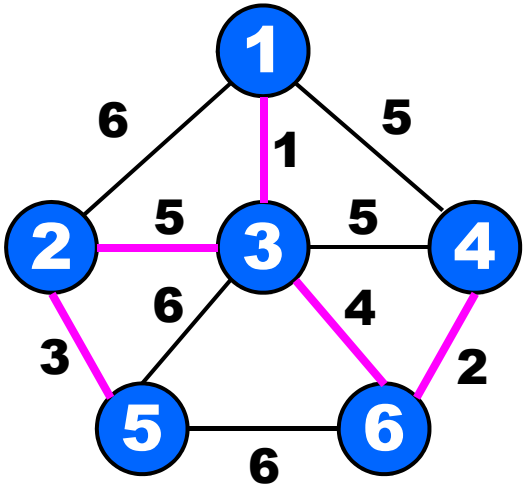
- 输入：图G　//顶点数n，边数m
- 输出：最小生成树T

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

Kruskal算法

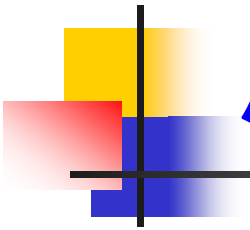
- 实例





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。



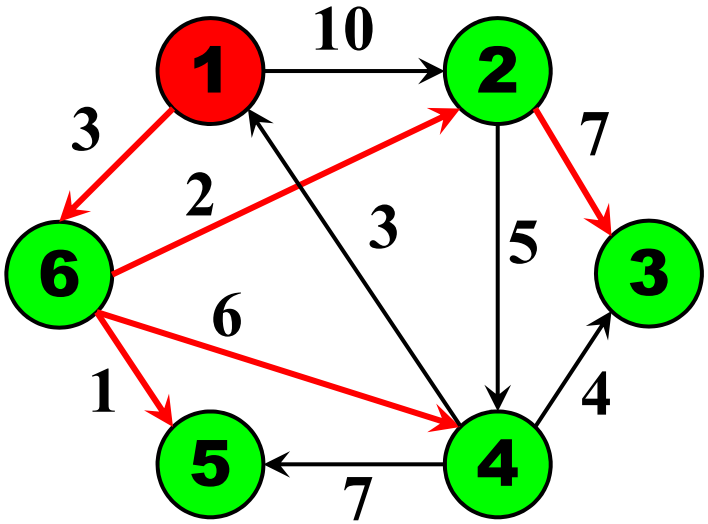
思考

- 基于上述Prim和Kruskal算法思想
- 对于边数相对较多（即比较接近于完全图）的无向连通带权图，比较适合于用哪种方法求解？
- 对边数较少的无向连通带权图有较高效率的又是哪一种算法？
- 请分析原因

单源最短路径问题

给定带权有向图G=(V,E,W)，每条边$e=<i, j>$的权$w(e)$为非负实数，表示i到j的距离。源点$s \in V$。

求：从s出发到达其他结点的最短路径。



- 源点：1
- 1→6→2: *short*[2]=5
- 1→6→2→3 : *short*[3]=12
- 1→6→4 : *short*[4]=9
- 1→6→5 : *short*[5]=4
- 1→6 : *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的最短路径的长度
- $short[u]$：从s到u的最短路径的长度
- $dist[u] \geqslant short[u]$

算法的设计思想

- **输入：** 有向图G=(V,E,W), V={1,2,...,*n*}, *s*=1
- **输出：** 从*s*到每个顶点的最短路径
- **步骤：**
  1. 初始S={1}；
  2. 对于*i*∈V-S，计算1到*i*的相对S的最短路径，长度记为*dist*[*i*]；
  3. 选择V-S中*dist*值最小的*j*，将*j*加入到S，修改V-S中的顶点的*dist*值；
  4. 继续上述过程，直到S=V为止。

算法伪码

- 算法Dijkstra

```
1.  S←{s}
2.  dist[s]←0
3.  for i∈V-{s} do
4.      dist[i]←w(s,i) //s到i没边，w(s,i) =∞
5.  while V-S ≠ ∅ do
6.      从V-S取相对S的最短路径顶点 j
7.      S←S∪{ j }
8.      for i∈V-S do
9.          if dist[j]+w(i,j)<dist[i]
10.         then dist[i]←dist[j]+w(i,j)
```

更新*dist*值

运行实例

- 输入：G=(V,E,W), V={1,2,3,4,5,6}, 源点1

S={1}

dist[1]=0

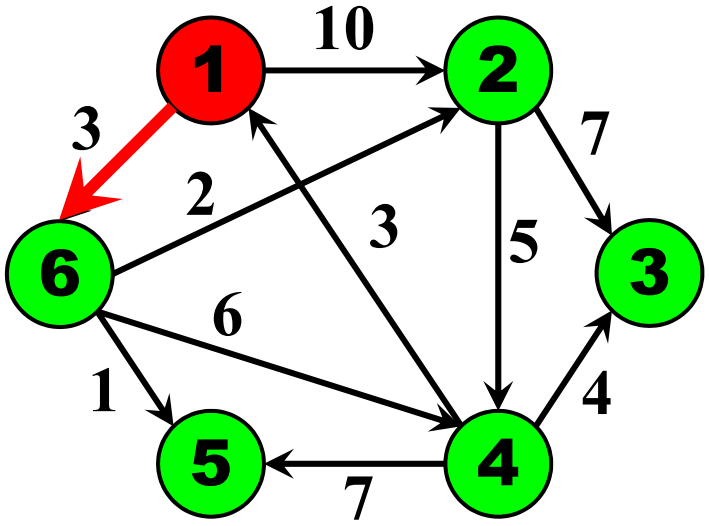
dist[2]=10

dist[6]=3

dist[3]=∞

dist[4]=∞

dist[5]=∞



运行实例

- 输入：G=(V,E,W), V={1,2,3,4,5,6}, 源点1

S={1,6}

dist[1]=0

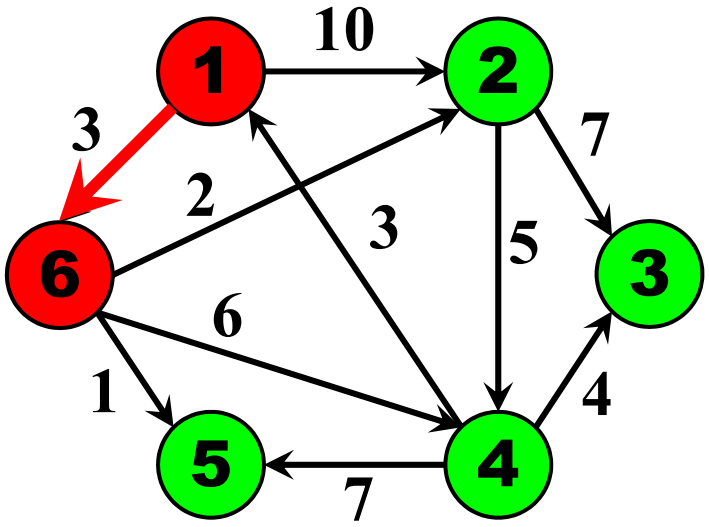
dist[6]=3

dist[2]=5

dist[4]=9

dist[5]=4

dist[3]=∞



运行实例

- 输入：G=(V,E,W), V={1,2,3,4,5,6}, 源点1

S={1,6,5}

dist[1]=0

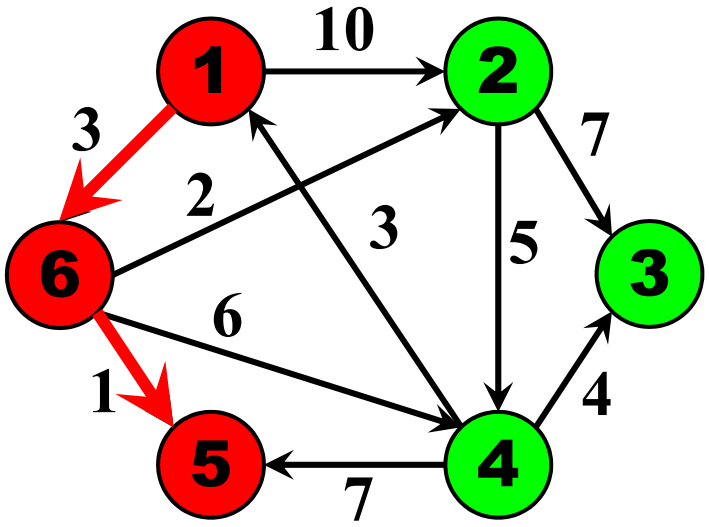
dist[6]=3

dist[5]=4

dist[2]=5

dist[4]=9

dist[3]=∞



运行实例

- 输入：G=(V,E,W), V={1,2,3,4,5,6}, 源点1

S={1,6,5,2}

dist[1]=0

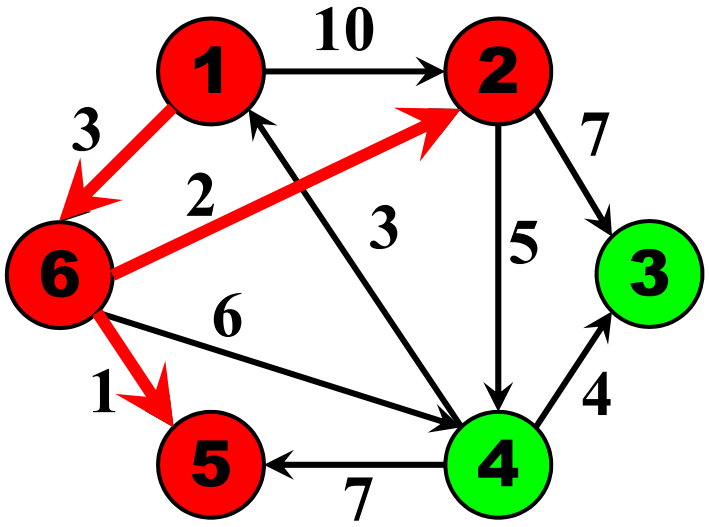
dist[6]=3

dist[5]=4

dist[2]=5

dist[4]=9

dist[3]=12



运行实例

- 输入：G=(V,E,W), V={1,2,3,4,5,6}, 源点1

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

dist[1]=0

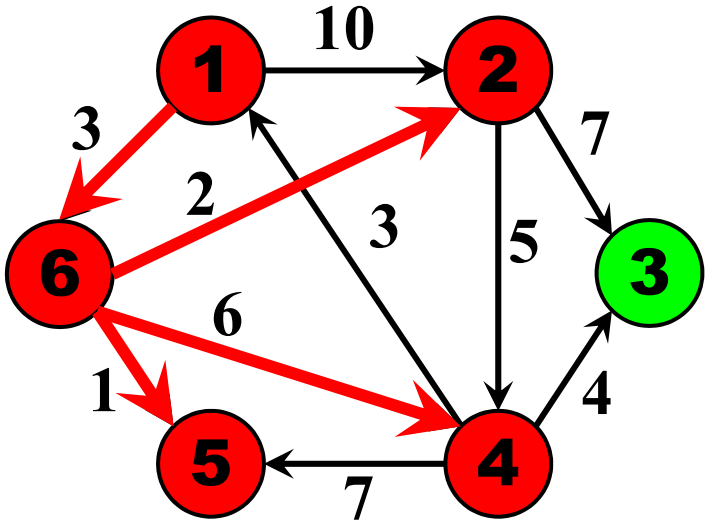
dist[6]=3

dist[5]=4

dist[2]=5

dist[4]=9

dist[3]=12



运行实例

- 输入：G=(V,E,W), V={1,2,3,4,5,6}, 源点1

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

dist[1]=0

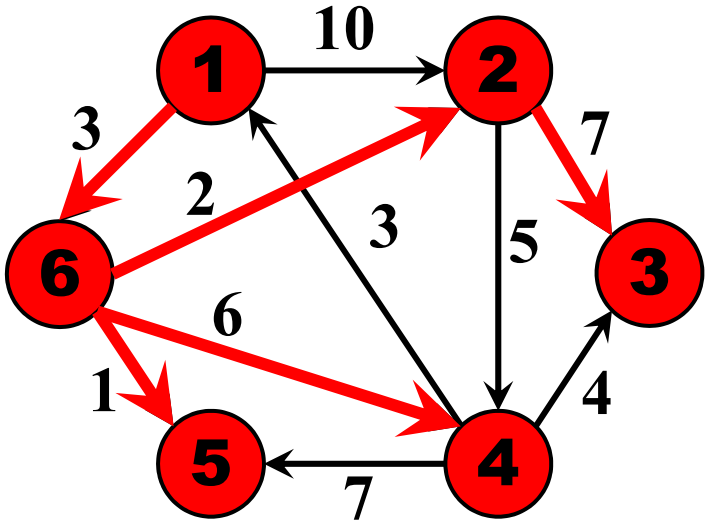
dist[6]=3

dist[5]=4

dist[2]=5

dist[4]=9

dist[3]=12



找到了问题的解！

时间复杂度分析

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

- 选用基于堆实现的优先队列的数据结构，可以将算法时间复杂度降低到O($m$log$n$)