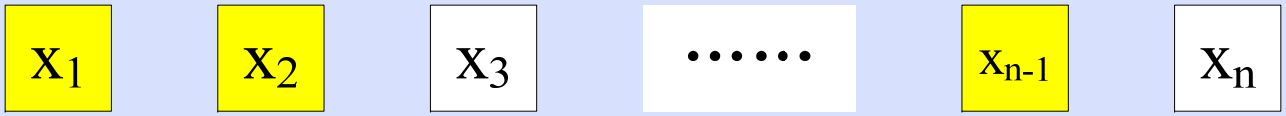
第4章 贪心算法

最优化问题的解一般可以描述为这样的形式



如：背包问题中x_i=0或1，表示不放或放

改进算法：多步决策，每步确定可行解

问：决策到第i步时，能知道前面i个物品放还是不放吗？为什么？

贪心算法：

（1）每步决策通过所谓的贪心选择（“只顾眼前”），可以确定一个xi的值

（2）不需计算子问题

好处：高效　　坏处：不一定能得到最优解

提　纲

- 应用背景和动机
- 贪心算法的基本思想
- 背包问题
- 最小生成树
- 单源最短路径
- 哈夫曼编码

应用背景和动机 —— 优化问题和贪心算法

❖ 最优化问题

做出一组选择以达到一个最优解，而不仅仅是一个解

❖ 贪心算法

贪心算法通常应用于最优化问题，目标也是获得一个最优解（不是总能如愿）

❖ 贪心算法如何工作

- 多阶段决策，做出一序列以获得一个最优解
- 每步选择都以局部最优（贪心选择）的方式来做出，希望通过多步的局部最优选择，达到最终的全局最优

❖找钱币的贪心算法

例子：有面值为5元、2元、1元、5角、2角、1角的货币，需要找给顾客4元6角现金

目标：找出的货币的数量最少

贪心算法：

多步决策：根据总额，每步确定不超过总额的最大面额货币数（贪心选择：最快地满足支付要求，其目的是使付出的货币张数最慢地增加），总额中减去已找面额获得新的总额（子问题）；重复上述过程直到剩余总额为0。

选择：

- 选出2元*2，余6角
- 选出5角，余1角
- 选出1角，余0
- 总共付出4张货币

得到的是整体最优解！！

例子：如果某一货币系统中，面值改为3元、1元、8角、5角、1角，同样需要找给顾客4元6角

贪心算法：

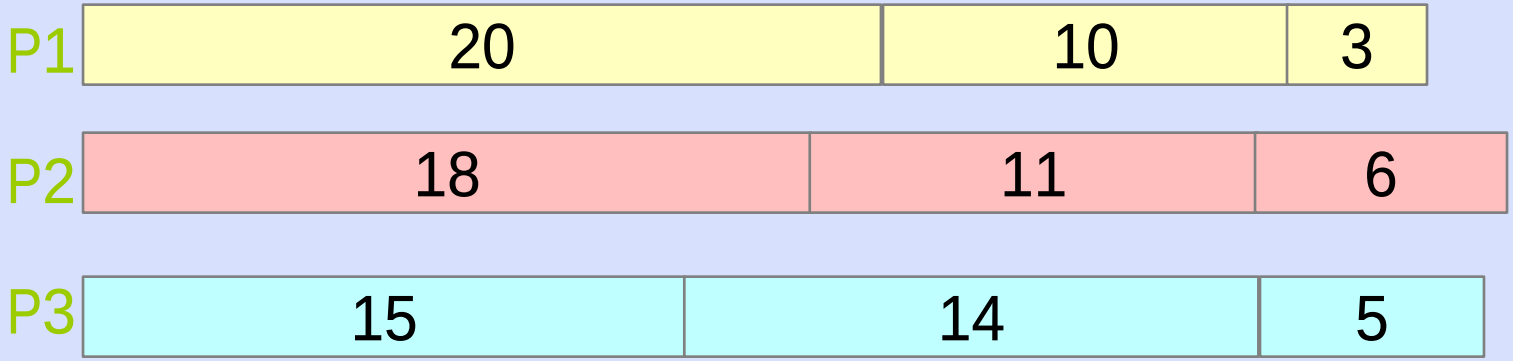
找给顾客的是1个3元、1个1元、1个5角和1个1角共4张货币

整体最优解是3张货币：1个3元和2个8角：

结论：贪心算法能获得可行解，但不一定是最优解！

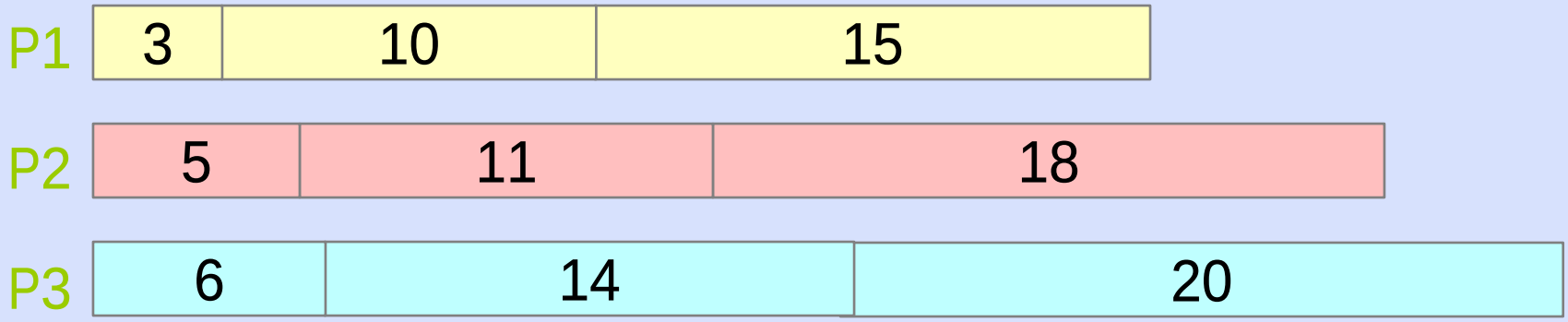
多机调度问题

- 有9个作业需要在处理机上运行，其运行时间分别是：3, 5, 6, 10, 11, 14, 15, 18, 20
- 有3台处理机可运行作业，目标：用最短时间处理完所有作业
- 多机调度问题是*NP*难问题，到目前为止还没有有效的解法。
- 贪心选择是最长处理时间作业优先，即把处理时间最长的作业分配给最先空闲的机器。



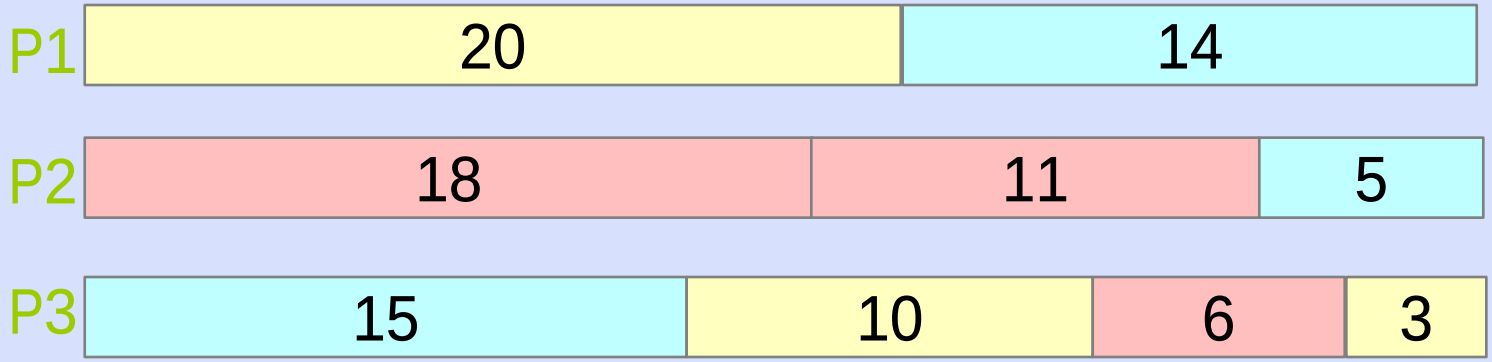
- 完成时间: 18 + 11 + 6 = 35
- 这个解不错，不过可能有更好的解

- 如果贪心策略采用最短处理时间作业优先



- 完成时间: **6 + 14 + 20 = 40**
- 这个贪心策略的结果不好
- 贪心算法足够快，每次决策仅仅需要挑出最大或最小的作业

- 最优解



- **20+14=34**
- 这个解明显是最优解
- 最优解不唯一
- 怎样获得最优解
  - 尝试所有可能安排
  - 不幸的是，这是指数时间

结论：贪心算法是有效的，算法的结果足够好！

提　纲

- 应用背景和动机
- 贪心算法的基本思想
- 背包问题
- 最小生成树
- 单源最短路径
- 哈夫曼编码

贪心算法的基本思想

❖引例

有n项活动申请使用同一个礼堂，每项活动有一个开始时间和一个截止时间，如果任何两个活动不能同时举行，问如何选择这些活动，从而使得被安排的活动数量达到最多？

建模：设S={1, 2,…., n}为活动的集合，s_i和f_i分别为活动i的开始时间和截止时间，i=1, 2,…, n，定义

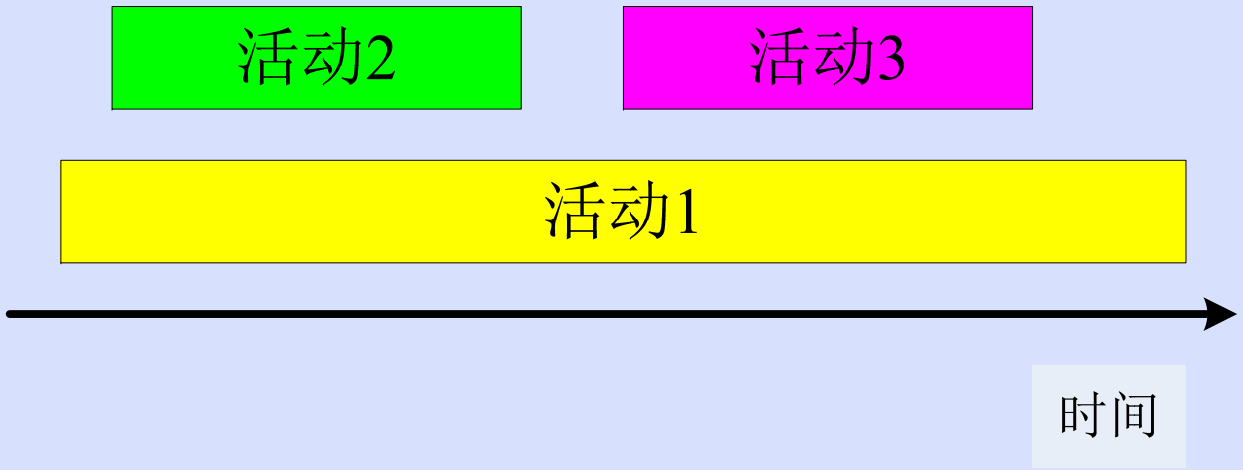
活动i与j相容 ⇔ $s_i \geq f_j$或$s_j \geq f_i$，$i \neq j$

目标：求S的最大的两两相容的活动子集A

多步决策:每步选择一项活动加入A

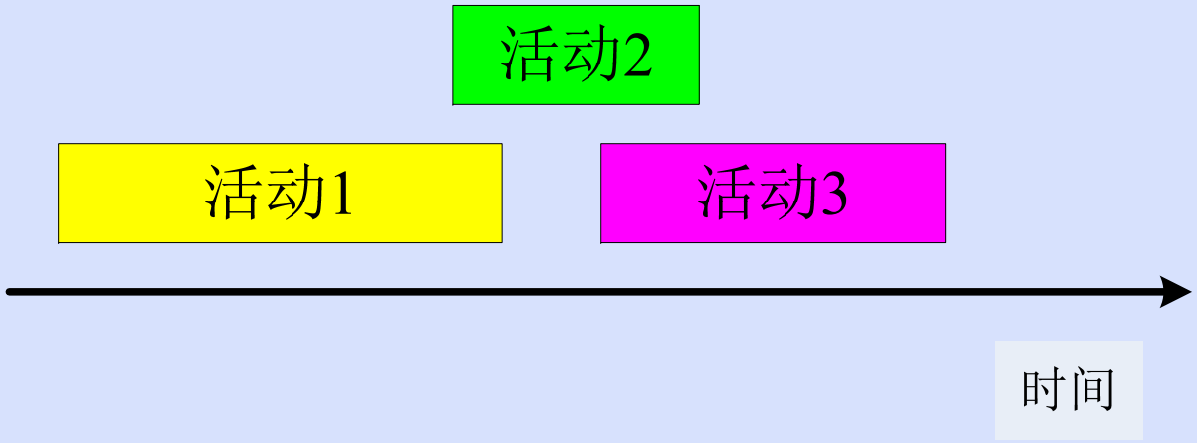
贪心选择策略:

（1）把活动按照开始时间从小到大排序，使得$s_1 \leqslant s_2 \leqslant \ldots \leqslant s_n$，然后从前向后挑选，只要与前面选的活动相容，就可以把这项活动选入A。



贪心选择策略：

（2）计算每个活动的占用时间，即f_i-s_i，然后，按照占用时间从小到大对活动排序，使得f_1-s_1≤f_2-s_2≤…≤f_n-s_n，然后从前向后挑选，只要与前面选的活动相容，就可以把这项活动选入A。



（3）把活动按照截止时间从小到大排序，使得$f_1 \leqslant f_2 \leqslant \ldots \leqslant f_n$，然后从前向后挑选，只要与前面选的活动相容，就可以把这项活动选入A（可行解）。

例：

i	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
s[i]	1	3	0	5	3	5	6	8	8	2	12
f[i]	4	5	6	7	8	9	10	11	12	13	14

最优解：A={x1=1, x2=4, x3=8, x4=11} 总共可安排的最大活动数是4个。

最优解不唯一：A={2, 4, 8, 11}

定理： 该算法执行到第k步，选择k项活动$i_1, i_2,\ldots, i_k$，那么存在最优解A包含$i_1, i_2,\ldots, i_k$

证明：将S中的活动按截止时间递增顺序排列，1, 2, … , n

归纳基础：k=1时，算法选择活动1。

证明：存在一个最优解包含了活动1，设$A=\{i_1, i_2,\ldots, i_j\}$是一个最优解，如果$i_1 \neq 1$，那么用1替换$i_1$，得到A′，即

$$A' = (A - \{i_1\}) \cup \{1\}$$

那么A′和A的活动个数相等，且活动1比i1结束的更早，因此和$i_2,\ldots, i_j$等活动相容，于是A′也是问题的一个最优解。

贪心选择性质: 贪心选择的结果包含在一个最优解中

归纳步骤：假设对于任意正整数**k**，命题正确。

令$i_1=1, i_2,\ldots, i_k$是算法前**k**步顺序选择的活动，那么存在一个最优解

$$A=\{i_1=1, i_2,\ldots, i_k\}\cup B$$

如果令**S′**是**S**中剩下的与$i_1=1, i_2,\ldots, i_k$相容的活动，即

$$S' = \{j \mid s_j \geq f_{i_k}, j \in S\}$$

那么，**B**是**S′**（子问题）的一个最优解，如若不然，假设**S′**有解**B′**，$|B'|>|B|$，那么用**B′**替换**B**以后得到的解$\{i_1=1, i_2,\ldots, i_k\}\cup B'$将比**A**的活动更多，与**A**是最优是矛盾的。

最优子结构性质: 原问题的最优解包含子问题的最优解

根据归纳基础的证明，算法第一步选择结束时间最早的活动总是导致一个最优解（贪心选择性质），故对于子问题S′存在一个最优解B″={i_{k+1},…}，（i_{k+1}是贪心算法第k+1步，也就是子问题S′第1步选择的结果）由于B″和B都是S′的最优解（最优子结构性质），因此|B″|=|B|，于是

$$A' = \{i_1 = 1, i_2, ..., i_k\} \cup B'' = \{i_1 = 1, i_2, ..., i_k, i_{k+1}\} \cup (B'' - \{i_{k+1}\})$$

与A（归纳假设）的活动数目一样多，也是一个最优解，而且恰好包含了算法前k+1步选择的活动，根据归纳法命题得证。

结论：最多n步的贪心选择得到的也是问题的最优解！！

- 贪心选择性质
- 所求问题的全局最优解可以通过一系列局部最优选择获得——贪心选择。
- 这是贪心算法可行（可求最优）的第一个基本要素，也是贪心算法与动态规划算法的主要区别。
- 贪心算法则通常以自顶向下的方式进行，以迭代的方式作出相继的贪心选择，每作一次贪心选择就将所求问题简化为规模更小的子问题。

- 最优子结构性质
- 问题的最优解包含其子问题的最优解
- 在原问题中做了一个贪心选择而得到一个子问题，

  证明：贪心选择+子问题最优解=原问题的最优解

❖贪心算法设计的步骤：

- 将优化问题转化为这样的一个问题，即先做出选择，再解决剩下的子问题

-证明原问题总是有一个最优解可以通过贪心选择得到，从而说明贪心选择是安全的

-说明在做出贪心选择后，剩余的子问题具有这样的性质，即如果将子问题最优解和贪心选择联合起来，可以得原问题的最优解

❖证明贪心算法的正确性（针对最优化问题的求解）：

- 证明每一步所作的贪心选择最终导致问题的整体最优解
- 数学归纳法

提　纲

- 应用背景和动机
- 贪心算法的基本思想
- 背包问题
- 最小生成树
- 单源最短路径
- 哈夫曼编码

背包问题

背包问题

- 背包容量为 C
- n 种物品，重量分别是 $\{w_1, \ldots, w_i, \ldots, w_n\}$
- 物品的总重量大于 C
- 求一个n元量$\{x_1, x_2, \ldots, x_n\}$ $(0 \leqslant x_i \leqslant 1)$（可以部分装入）
- 问题：使得装入背包中物品的总价值最大

- 目标函数：

$$\text{maximize} \sum_{1 \le i \le n} v_i x_i$$

- 约束：

$$\sum_{1 \le i \le n} w_i x_i \le C$$

- 例子：$n=3, C=20, (v_1,v_2,v_3)=(25,24,15), (w_1,w_2,w_3)=(18,15,10)$

至少有三种看似合理的贪心策略：

（1）选择价值最大的物品

（2）选择重量最轻的物品

（3）选择单位重量价值最大的物品

(x_1, x_2, x_3)	$\sum w_i x_i$	$\sum v_i x_i$	
(1) (1, 2/15, 0)	20	28.2	价值最大！
(2) (0, 2/3, 1)	20	31	重量最轻！
(3) (0, 1, 1/2)	20	31.5	单位重量价值最大

贪心选择：

将尽可能多地将单位重量价值(v_i/w_i)最高的物品装入背包，直到装满

算法：

- 按照v_i/w_i进行非升序排序

```
for(int i=0;i<n;i++) d[i]←v[i]/w[i];
mergeSort(d);
```

- 初始化

```
for (i=0;i<n;i++) x[i]←0;
```

- 贪心选择

```
for (i=0;i<n;i++) {
 if (w[i]>c) break;
 x[i]←1; opt←opt+v[i]; c←c-w[i];
}
if (i<n){
 x[i]←c/w[i]; opt+=x[i]*v[i];
}
```

- 背包问题用贪心算法能够得到最优解

- 主要计算时间
  单位重量价值的排序：
  $O(n\log n)$

把所有物品按照*vi/wi*进行非升序排序，

物品：**1, 2, …., n**

就这个问题而言就需要证明以下命题：

贪心选择性质：存在一个最优解包含了物品**1**

最优子结构性质：假设对于任意正整数**k**，命题正确。令**x1=1, x2,…, xk**是算法前**k**步顺序选择的物品,那么存在一个最优解设为：

X={x1=1,,x2, …..xk}∪B

如果令**C′ =C-** $\sum_{1\le i\le K} w_i x_i$,那么，**B**是**C′** 的一个最优解

背包问题 vs. 0-1背包问题

- 背包：　背包装满
  0-1背包： 背包未必装满（贪心选择失效）
- 背包：　子问题不重叠，每次选择物品使得局部最优
  0-1背包： 子问题重叠，考查的是物品选择与不选择
- 背包：　每一步的工作很少，且基于少量的信息
  在少量的计算基础上做出猜想而不急于考虑以后的情况
  0-1背包： 第k阶段的决策用到前面1~k-1阶段决策的结果
- 背包：　自顶向下每一步都扩大解的规模
  0-1背包： 自底向上计算最优值

◆ 结论：

贪心算法的关键：贪心选择标准

贪心算法的困难：对于最优化问题，算法是否求解了所要解决的问题
（证明问题的贪心算法是正确的）

提　纲

- 应用背景和动机
- 贪心算法的基本思想
- 背包问题
- 最小生成树
- 单源最短路径
- 哈夫曼编码

最小生成树——问题的提出

- **应用背景和动机：**

若图*G*的顶点表示城市，边*(v,w)*的权表示城市*v*和*w*之间通信线路所需的费用，如何建立代价费用最低的通信网络？

- **问题：**

对连通网来说，最小生成树（ **Minimum Spanning Tree ， MST**）是它的一个极小连通子图：

- 包含所有顶点
- 包含**n-1**条边
- 连通
- 边的权值相加最小

- 穷举搜索

  寻找所有可能的边的子集，直到找到最小生成树

考虑时间

- $G=(V, E)$, $|V|=n$, $|E|=m$，包含$n-1$条边的子集数 C_m^{n-1}
- 找到一个子集，是否连接了所有顶点，计算总的权值
- 指数阶

❖Prim 算法

- $G=(V, E)$, $V=\{1, 2, \ldots, n\}$, $c[i, j]$为边(i, j)的权重

基本思想：

初始$S=\{1\}$

$i \in S, j \in V-S$ 且 $c[i, j]$最小

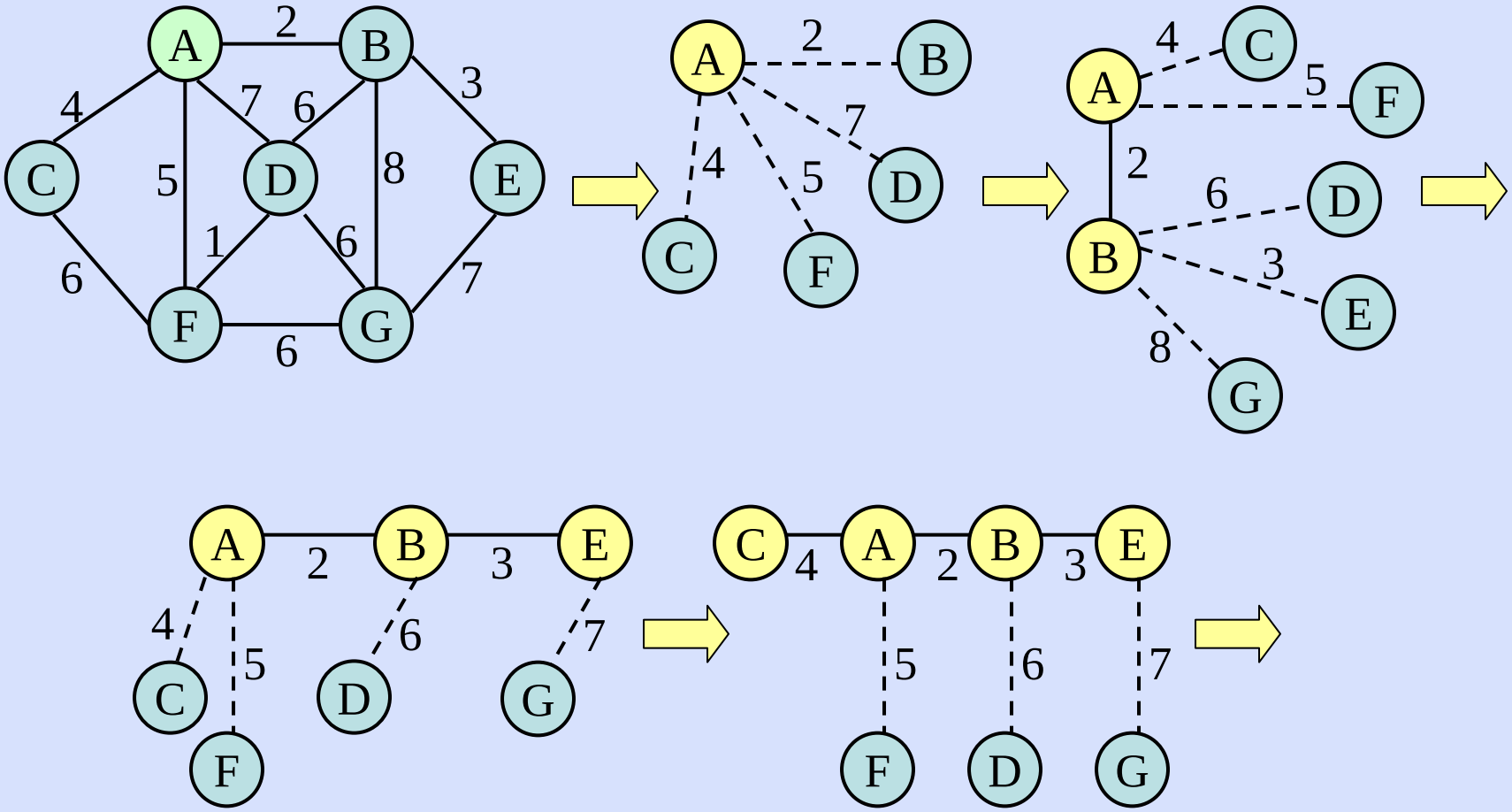
直到$S=V$

贪心选择

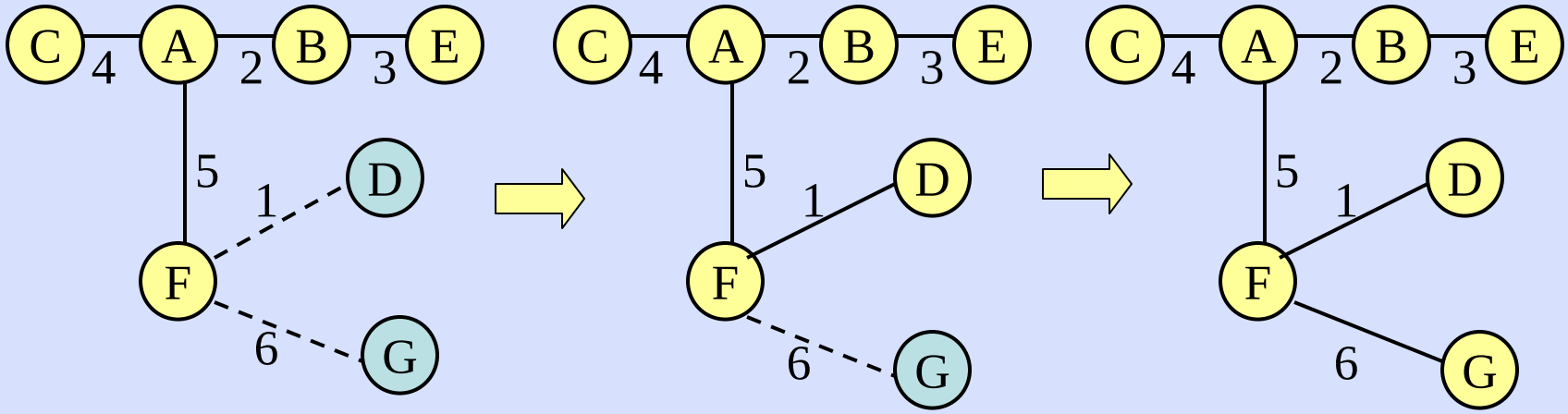
算法：

```
T=Φ
S={1}
while(S!=V){
 (i,j)=i  S且j  V-S的c[i,j]最小的边
 T=T  {(i,j)}
 S=S  {j}
}
```

例子：



- 接上页



- 时间复杂度

  $G=(V, E)$, $|V|=n$, $O(n^2)$

正确性证明

定理：对于任意正整数k<n，存在一棵最小生成树包含算法前k步选择的边。

证：k=1时，选择的边包含在一棵最小生成树中

假设k-1时成立, $E=\{e_1,e_2,\ldots,e_{k-1}\} \cup B$

证k时选择的边也成立（MST性质）

❖Kruskal 算法

- 根据边构造最小生成树

基本思想

1. 初始化：V中每个顶点自成一个连通分量。 TE={ }
2. 循环直到图中的连通分量个数为1
   - 2.1 在E中寻找最短边(u，v);
   - 2.2 如果顶点u、v位于图中的两个不同连通分量，则
     - 2.2.1 将边(u，v)并入TE；
     - 2.2.2 将这两个连通分量合为一个；
   - 2.3 E=E-{(u，v)}；

贪心选择

怎样实现算法?

算法：

输入： G=(V,E); 输出： TE: MST

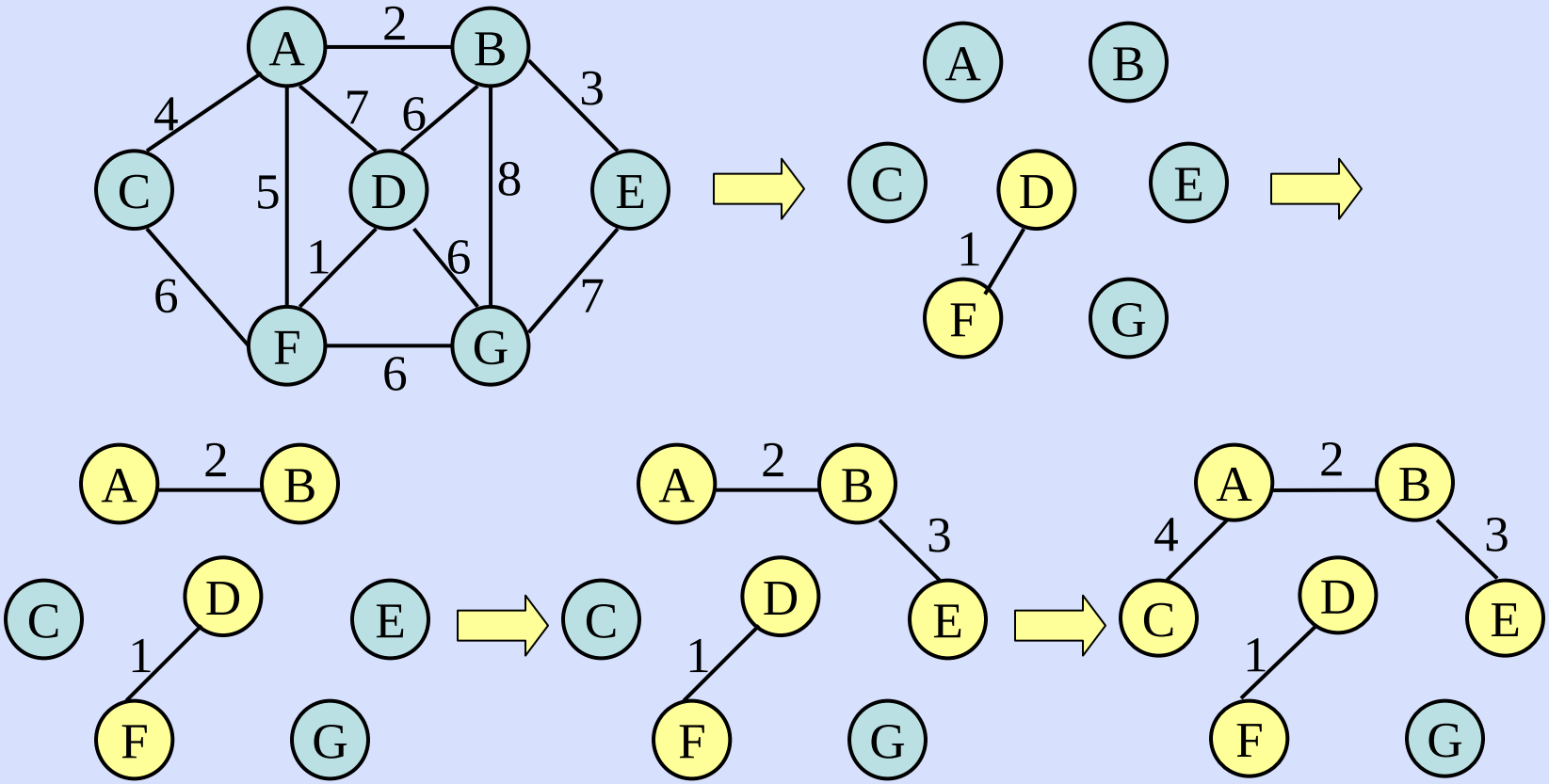
1.按照非降序权重将E中的边排序
2.for 每条边v∈V do
3. makeSet(v);
4.end for
5.TE={}
6.while |TE|<n-1 do
7. 令(x,y)为E中下一条边
8. if find(x)≠find(y) then
9. 将(x,y)加入TE
10. end if
11.end while

计算时间$O(e \log e)$

另一种实现方式：

- 优先队列
- 用min堆实现这个优先队列
- 运算：removeMin(H)，$O(\log e)$
- 计算时间 $O(e \log e)$

- 例子


A
B
C
D
E
F
G
2
4
7
6
3
5
8
1
6
6
7
6
A
B
C
D
E
1
F
G
A
2
B
C
D
1
E
F
G
A
2
B
3
C
D
1
E
F
G
A
2
B
4
3
C
D
1
E
F
G

- 接上页

- 考虑权值为 6的边： ***BD, DG, GF, FC***

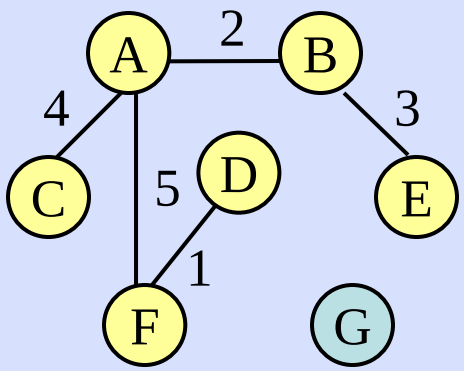
- 选那一个?

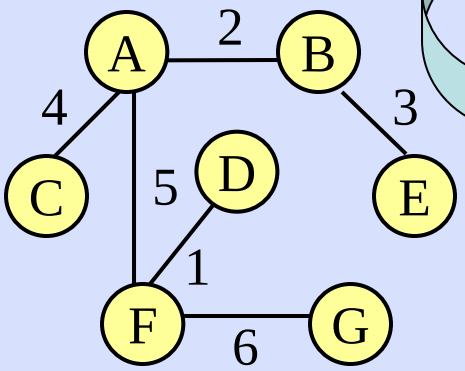
(1) 是否会产生环?

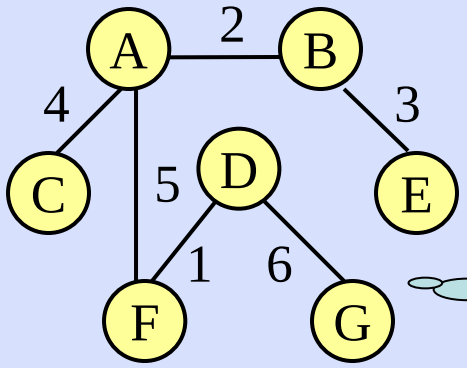
(2) 不会，可以选择

- 考虑 ***BD*** **and** ***FC*****?** 出现环！！

- 可以在 ***GF*** **and** ***DG***中进行选择







最小生成树不唯一！！

正确性证明

定理：对于任意正整数k<n，存在一棵最小生成树包含算法前k步选择的边。

证：k=1时，选择的边包含在一棵最小生成树中

假设k时成立，$E=\{e_1,e_2,\ldots,e_k\} \cup B$

证k+1时选择的边也成立（MST性质）

Kruskal 算法的应用：k聚类（层次凝聚）

设集合S={1,2,…,n},$\forall i,j \in S, i \neq j, d(i,j) = d(j,i)$表示i与j的相似度，假设需要将S划分为k个子集$C_1,C_2,\ldots,C_k$,聚类$L=\{C_1,C_2,\ldots,C_k\}$的间隔（距离）定义为：

$$D(L) = min\{d(i,j)|i \in C_s, j \in C_t, 1 \leq t < s \leq k\}$$

给定S中元素之间的相似度，求使得D(L)达到最大的k聚类。

提　纲

- 应用背景和动机
- 贪心算法的基本思想
- 背包问题
- 最小生成树
- 单源最短路径
- 哈夫曼编码

单源最短路径 —— Dijkstra 算法

- 基本思想：
  - 给定带权有向图$G=(V, T)$
  - 初始：S中仅含有源点v；
  - 从源到u，$u \in V-S$的特殊路径：从源到u且中间只经过S中顶点的路径，路径长度记为$dist[u]$
  - while $S \neq V$ do
    对于$u \notin S$的结点，选出$dist[u]$最小者加入S
    修改$dist$的值：对于$j \notin S$且与u邻接的结点，
    置$dist[j]=\min\{dist[j], dist[u]+a[u, j]\}$
    end while

贪心选择

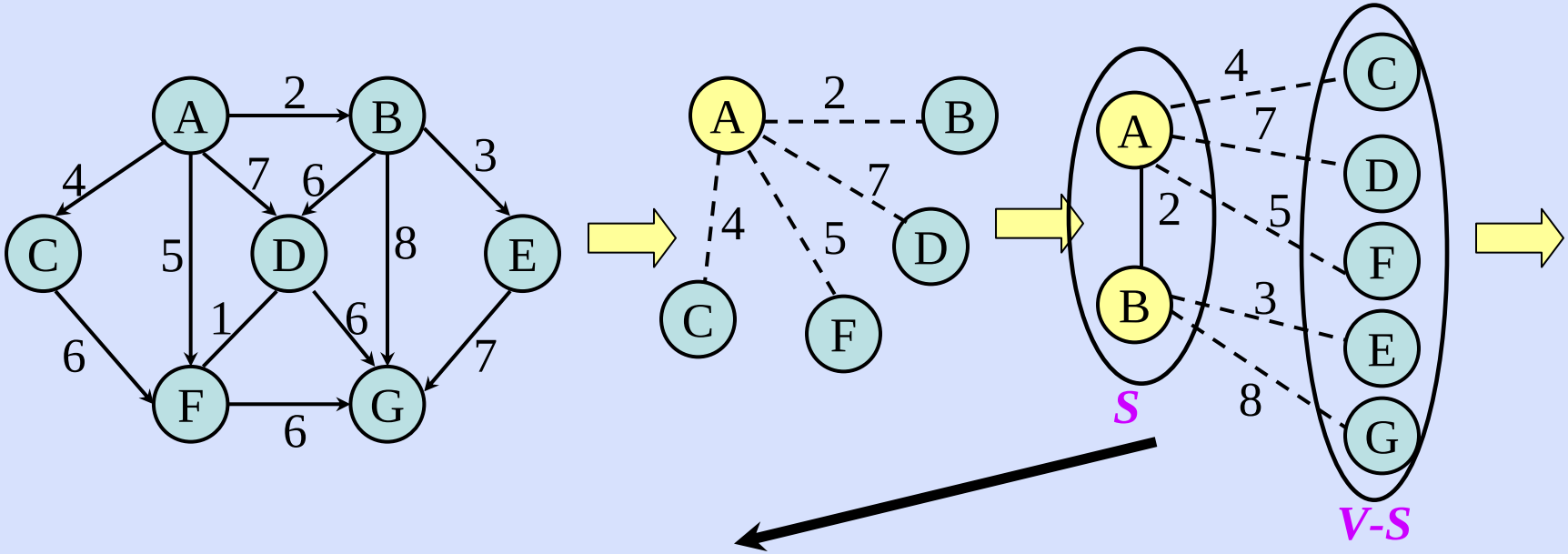
- 另一种理解：

计算时间 $O(n^2)$

每次贪心选择：

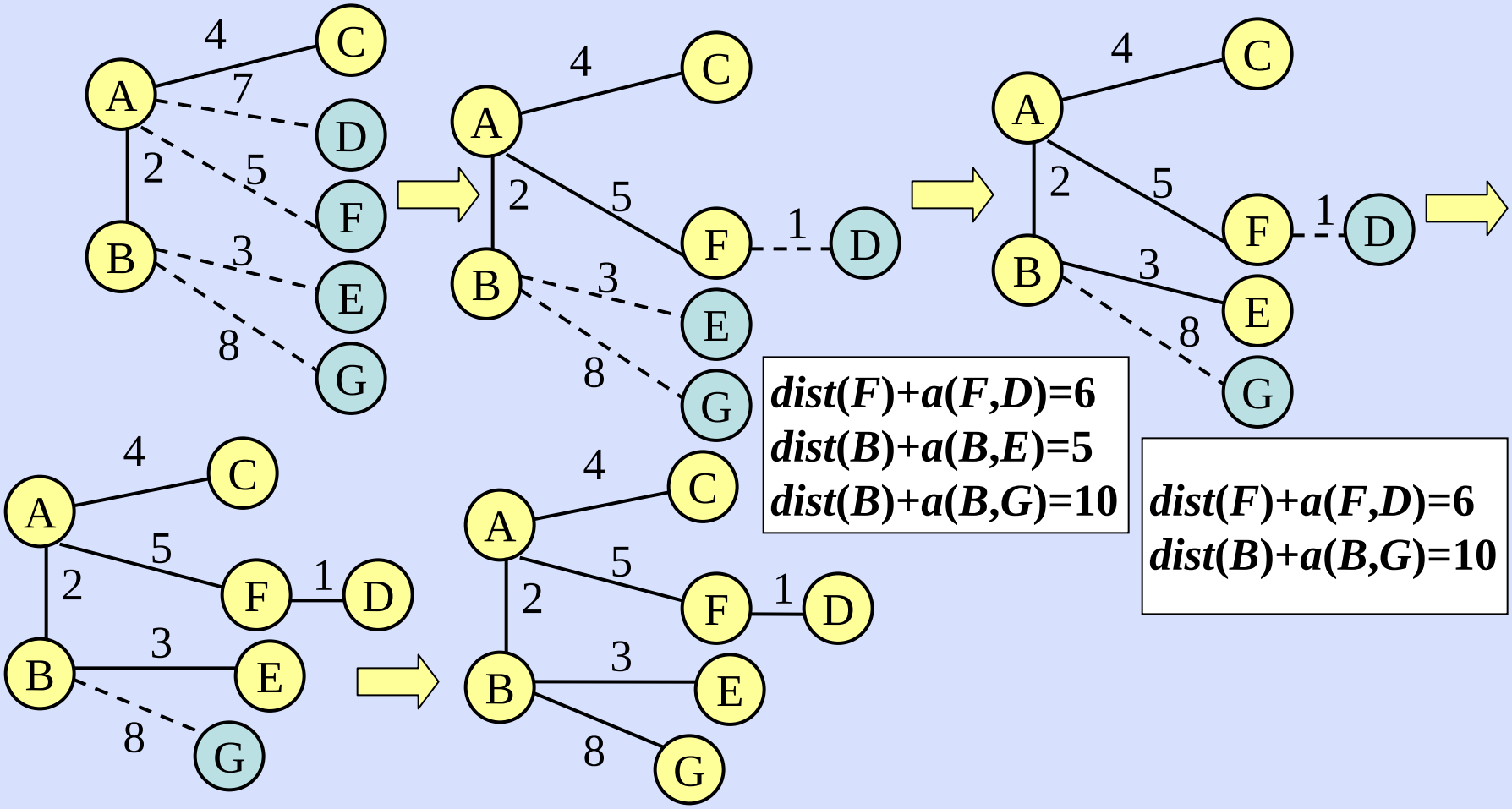
对于$x \in S, j \notin S$, 选择使得$dist[x]+a[x, j]$最小的结点j加入S, 并修改候选结点集

- 例子



***dist*(*A*)+*a*(*A*,*C*)=4　*dist*(*A*)+*a*(*A*,*D*)=7**
***dist*(*A*)+*a*(*A*,*F*)=5　*dist*(*B*)+*a*(*B*,*E*)=5**
***dist*(*B*)+*a*(*B*,*G*)=10**

- 接上页


A
B
C
D
E
F
G
4
7
2
5
3
8
1
dist(F)+a(F,D)=6
dist(B)+a(B,E)=5
dist(B)+a(B,G)=10
dist(F)+a(F,D)=6
dist(B)+a(B,G)=10

算法正确性证明

定理 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.

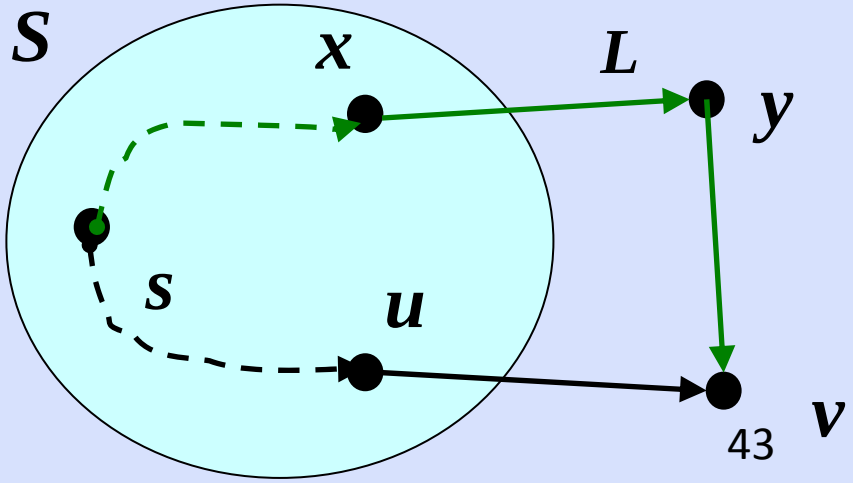
归纳步骤　假设命题对于k 为真. 考虑 k+1步, 选择顶点v (边{u,v}). 假若存在另一条 s-v 路径 L (绿色)，第一次出S 的顶点为 x, 在这次从S 中出来后经过V–S 的第一个顶点为 y.

$dist[v] \le dist[y]$　　//v先被选

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

$dist[v]=short[v]$

时间复杂度 $T(n)=O(n^2)$



提　纲

- 应用背景和动机
- 贪心算法的基本思想
- 背包问题
- 最小生成树
- 单源最短路径
- 哈夫曼编码

哈夫曼编码

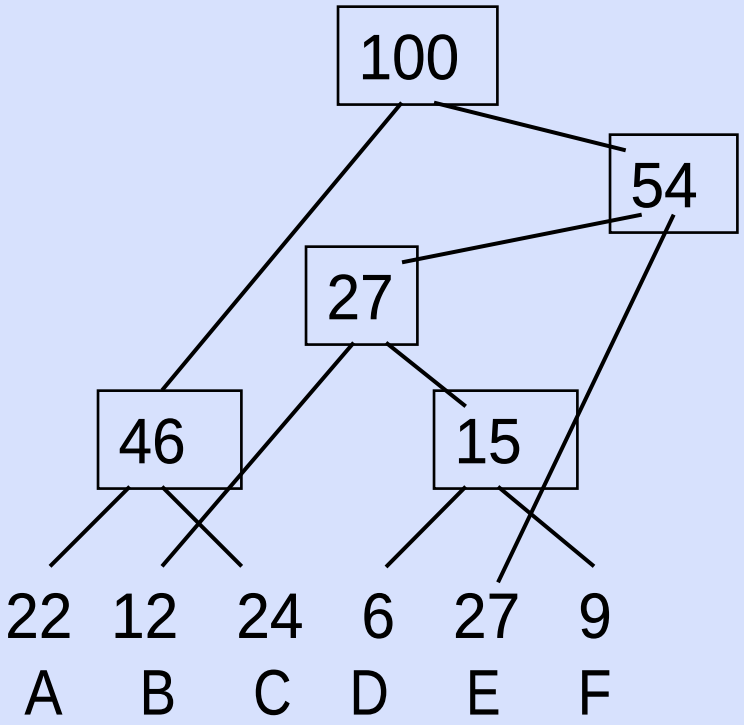
- 背景：
  - 哈夫曼编码广泛地用于数据文件压缩
  - 压缩率通常在20%～90%之间（变长码）
  - 哈夫曼编码根据字符在文件中出现的频率来建立一个用0，1串表示各字符的最优表示方式
  - 给出现频率高的字符较短的编码，出现频率较低的字符以较长的编码，可以大大缩短总码长
- 前缀码：
  - 任一字符的代码(0,1序列)都不是其他字符代码的前缀——完全二叉树T
  - 满足前缀约束，则编码无二义性
  - 尽可能多地压缩文件、源文件很容易被重建 （最优前缀码）
  - 平均码长：

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

字符c出现的频率为$f(c)$，
在T中的深度为$d_T(c)$

❖构造哈夫曼编码

- 哈夫曼算法是构造最优前缀码的贪心算法
- 自底向上，选择最小的两个权值合并（|C|-1次），最优二叉树
- 基于最优二叉树获得的前缀码是最优前缀码



A=00
B=100
C=01
D=1010
E=11
F=1011

- 平均码长:
  0.22*2 + 0.12*3 + 0.24*2 + 0.06*4 + 0.27*2 + 0.09*4 = 2.42

- 算法：

以|*C*|个叶结点开始，执行|*C*|－1次的“合并”运算后产生最终所要求的树 *T*

```
n←|C|; T←{};
MinHeap H←new minHeap( ); //优先队列——min堆
H.initialize(w, n);

for(i←1;i<n;i++){
  c←H.removeMin( );
  c'←H.removeMin( );
  f(v)←f(c)+f(c'); //合并最小频率树
  Insert(H,v);
  T=T ∪{(v,c), (v,c')}
}
```

O(*n*)

O(log*n*)

计算时间*O*(*n*log*n*)

❖哈夫曼算法的正确性证明

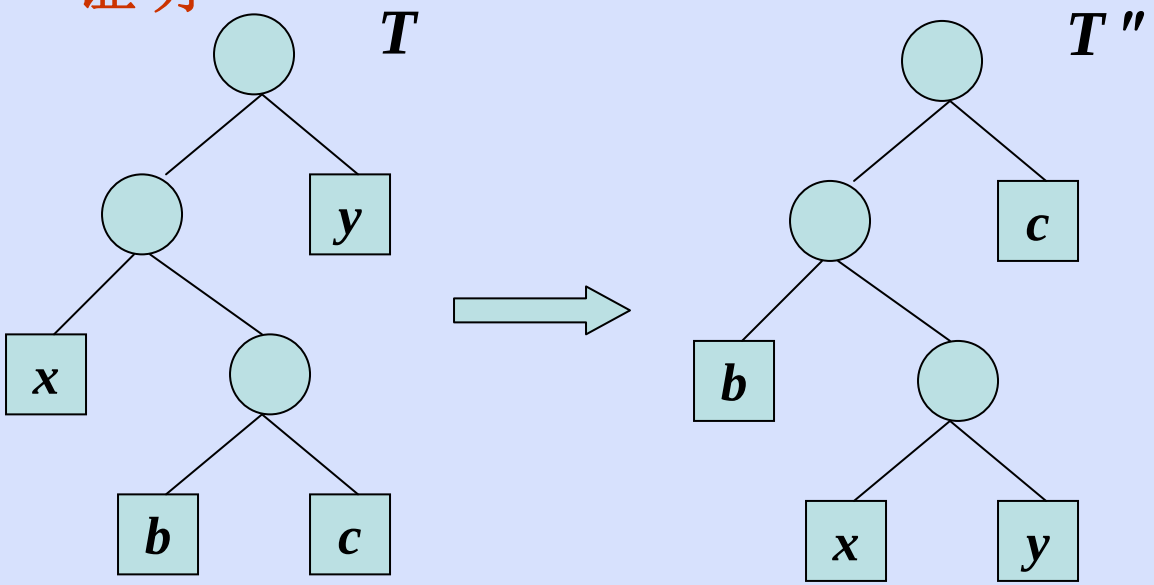
- 贪心选择性质

设*C*是编码字符集，*C*中字符*c*的频度为*f(c)*，设*x*和*y*是*C*中具有最小频率的两个字符，存在*C*的最优前缀码使*x*和*y*具有相同码长，且仅最后一位编码不同

- 思路

反证法。 对最优前缀码二叉树*T*作修改得*T"*，*T"*表示对*C*做出贪心选择得到的最优前缀 码，*x, y*是*T"*中最深叶子且为兄弟。要说明：树*T"*与*T*具有相等的平均码长。（设b,c是T中最深的两个兄弟结点）

- 证明



- *T*中：
  - $f(b) \le f(c)$
    $f(x) \le f(y)$
  - *x*和*y*是具有最小频率的两个字符
    $f(x) \le f(b)$
    $f(y) \le f(c)$

- *T "*是对*C*做出贪心选择的前缀编码树

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

$$=[f(x)d_T(x)+f(y)d_T(y)+f(b)d_T(b)+f(c)d_T(c)]-$$

$$[f(x)d_{T''}(x)+f(y)d_{T''}(y)+f(b)d_{T''}(b)+f(c)d_{T''}(c)]$$

$$=[f(x)d_T(x)+f(y)d_T(y)+f(b)d_T(b)+f(c)d_T(c)]-$$

$$[f(x)d_T(b)+f(y)d_T(c)+f(b)d_T(x)+f(c)d_T(y)]$$

$$=f(x)(d_T(x)-d_T(b))+f(y)(d_T(y)-d_T(c))+$$

$$f(c)(d_T(c)-d_T(y))+f(b)(d_T(b)-d_T(x))$$

$$=(f(b)-f(x))(d_T(b)-d_T(x))+(f(c)-f(y))(d_T(c)-d_T(y))$$

由 $f(x)\le f(b), d_T(x)\le d_T(b)$ 以及 $f(y)\le f(c), d_T(y)\le d_T(c)$

则 $B(T)-B(T'')\ge 0$ ，由于T是C的最优前缀编码树，所以$B(T)=B(T'')$

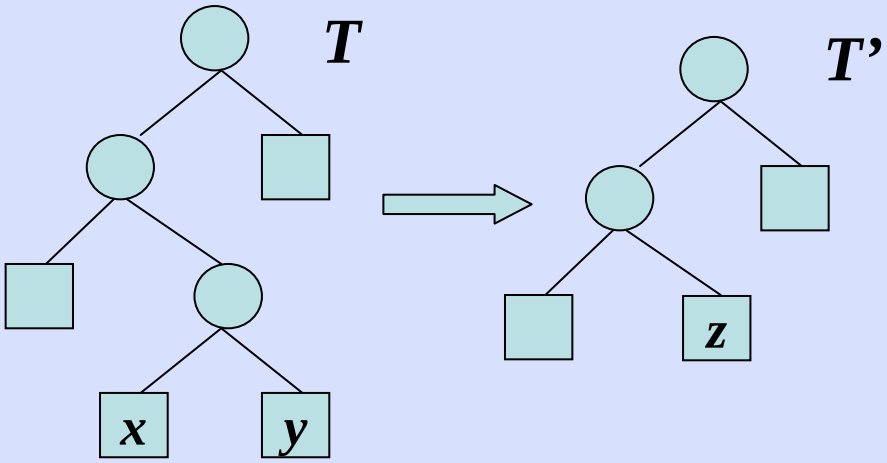
- 最优子结构

设T是表示字符集C的一个最优前缀码的完全二叉树，C中字符c的频度为$f(c)$，设x和y是树T中的两个叶子且为兄弟，z是它们的双亲。若将z看作是具有频率$f(z)=f(x)+f(y)$的字符，则树$T'=T-\{x,y\}$表示字符集$C'=C-\{x,y\}\cup\{z\}$的一个最优前缀码

- 思路

反证法，若T'不是表示字符集C'的一个最优前缀码。有T''是表示字符集C'的一个最优前缀码，把T''中的z展开，得表示C的前缀码完全二叉树T'''，推得T'''的平均码长小于T的平均码长，和T是表示字符集C的一个最优前缀码的完全二叉树矛盾。

- 证明



总结

(1)适用于优化问题， 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. 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) 如果对原始数据预处理之后，贪心法往往是一轮处理，时间复杂度和空间复杂度低.


谢 谢 大 家！