贪心算法

参考《算法导论》

Contents

- 1 活动安排问题
- 2 贪心算法的原理
- 3 哈夫曼编码
- 4 最小生成树
- 5 单源最短路

1 活动安排问题（《算法导论》）

- n个活动申请一个活动室, 各活动起始终止区间(s_i, f_i)
- 输入: n, (s_i, f_i) , i ∈[1, n]
- 输出: 最大相容活动子集(活动之间无冲突, 活动个数最多)
- 例:
  - 相容活动子集: $\{a_3, a_9, a_{11}\}$
  - 最大相容活动子集: $\{a_1, a_4, a_8, a_{11}\}$

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	9	9	10	11	12	14	16

按终止时间排序 $f_1 \le f_2 \le \ldots \le f_n$.

1 活动安排问题-贪心算法

- 贪心选择：选择一个可以加入到最优解中的活动
- 贪心选择：选择最早结束的活动
- 算法过程：
  1. 按终止时间排序
  2. 选择最早结束的活动，即第一个活动
  3. 在其余活动集合中选择与已选活动不冲突且结束时间最早的活动。

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	9	9	10	11	12	14	16

按终止时间排序 $f_1 \le f_2 \le \ldots \le f_n$.

1 活动安排问题-贪心算法

```
Greedy-Activity-Selector (int * s, int * f , int n)
{//假设活动已经按照结束时间排序
        A = {a1} ; / /选择最早结束的活动a1。
        k = 1; //设k为下一选择活动的前驱活动
        for( m = 2;  m<=n; m++){ //按结束时间依次考察各个活动
                if( s[m] >= f[k] ){ // m开始时间大于k结束时间
                        A = A∪ {am};  //选择下一个活动m
                        k = m;  //更新前驱
                }//if
        }//for
        return A;
}// Greedy-Activity-Selector
```

算法正确性证明

- 定义：任务集合S_k：a_k结束后开始的任务集合，即

$$S_k=\{a_i\in S: s_i \geq f_k\}$$

- 例如：
  - $S_1 = \{a_4, a_6, a_7, a_8, a_9, a_{11}\}$
  - $S_2 = \{a_4, a_6, a_7, a_8, a_9, a_{11}\}$
  - $S_3 = \{a_7, a_8, a_9, a_{11}\}$

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	9	9	10	11	12	14	16

算法正确性证明

- 证明1：活动选择问题具有最优子结构性质
- （最优子结构性质：问题的最优解包含其子问题的最优解）
- 证明：
  - 假设A是包含a_1的最大相容活动集(最优策略)
  - 因为a_1是最早结束的活动，
  - 所以与a_1兼容的活动都必须在a_1结束之后开始。
  - 即A-{a_1}是剩下的S_1(子问题)上的最大相容活动集

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	9	9	10	11	12	14	16

算法正确性证明

- 证明2：贪心选择是最优解的一部分。
- 定理16.1 对于任意非空子问题S_k，令a_m是S_k中结束最早的活动，则a_m在S_k的某个最大相容子集中。
- 证明：
  - 令A_k是S_k的一个最大相容子集，且a_j是A_k中结束最早活动
  - 若$a_j=a_m$，则得证。
  - 若a_j<>a_m，则 $f_j >= f_m$。将A_k中的a_j替换为a_m，得到子集A'_k
  - $|A'_k| = |A_k|$
  - A'_k是S_k的一个最大相容子集.

$\{a_2, a_4, a_8, a_{11}\}$

$\{a_1, a_4, a_8, a_{11}\}$

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	9	9	10	11	12	14	16

1 活动安排问题-贪心算法总结

- 算法设计
- 第一步 做出贪心选择：
  - 选择一个活动，加入到最优解中。
  - 即选择最早结束的活动a_1 。
- 第二步 求解剩余子问题：
  - 即寻找a_1结束后开始的活动。
  - 若A是包含a_1的最大相容活动集(最优策略),
  - 则A-{a_1}是剩下的S_1(子问题)上的最大相容活动集
- 证明算法的正确性：即证明贪心选择总是最优解的一部分
  - 数学归纳法

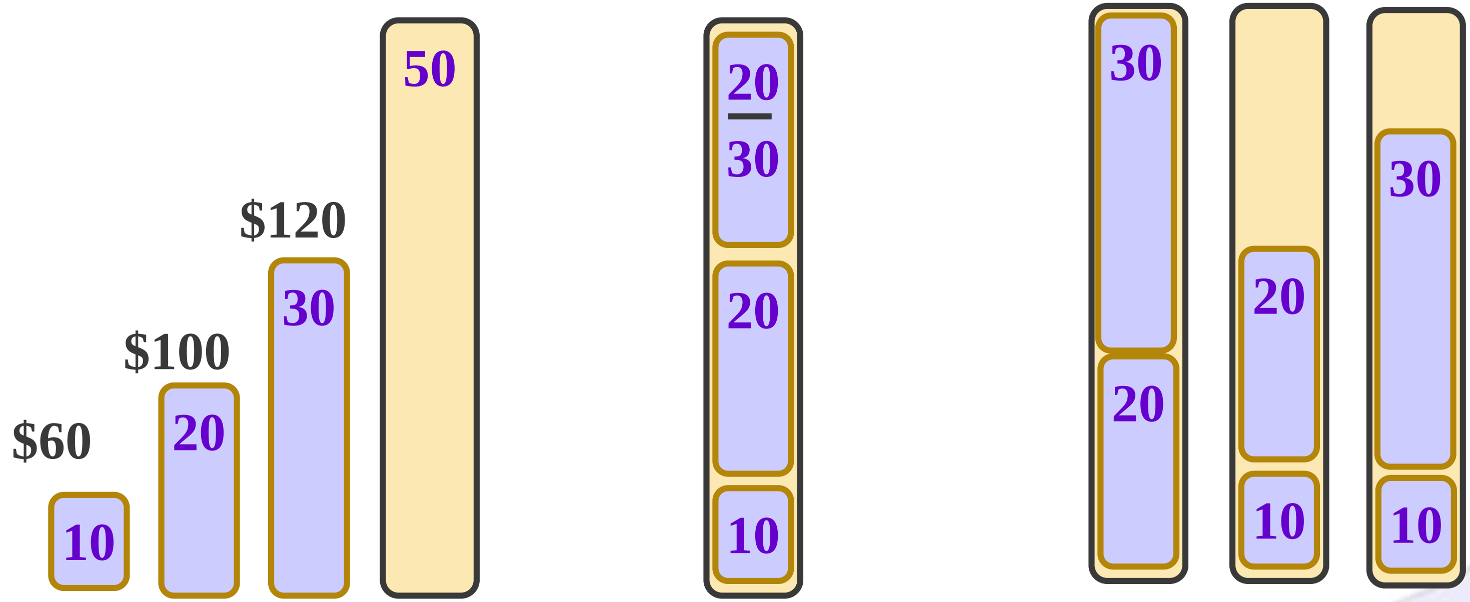
2 贪心算法原理

- 贪心算法的设计步骤
  I. 将最优化问题转化为下面的形式：做出一次选择后，只剩下一个子问题需要求解
  II. 证明算法正确性：
     a) 贪心选择性：即可以通过做出局部最优选择来构造全局最优解。
     b) 证明最优子结构性：一个问题的最优解包含其子问题的最优解。

  即做出贪心选择后，剩余子问题的最优解与贪心选择组合即可得到原问题的最优解。

2 贪心算法原理

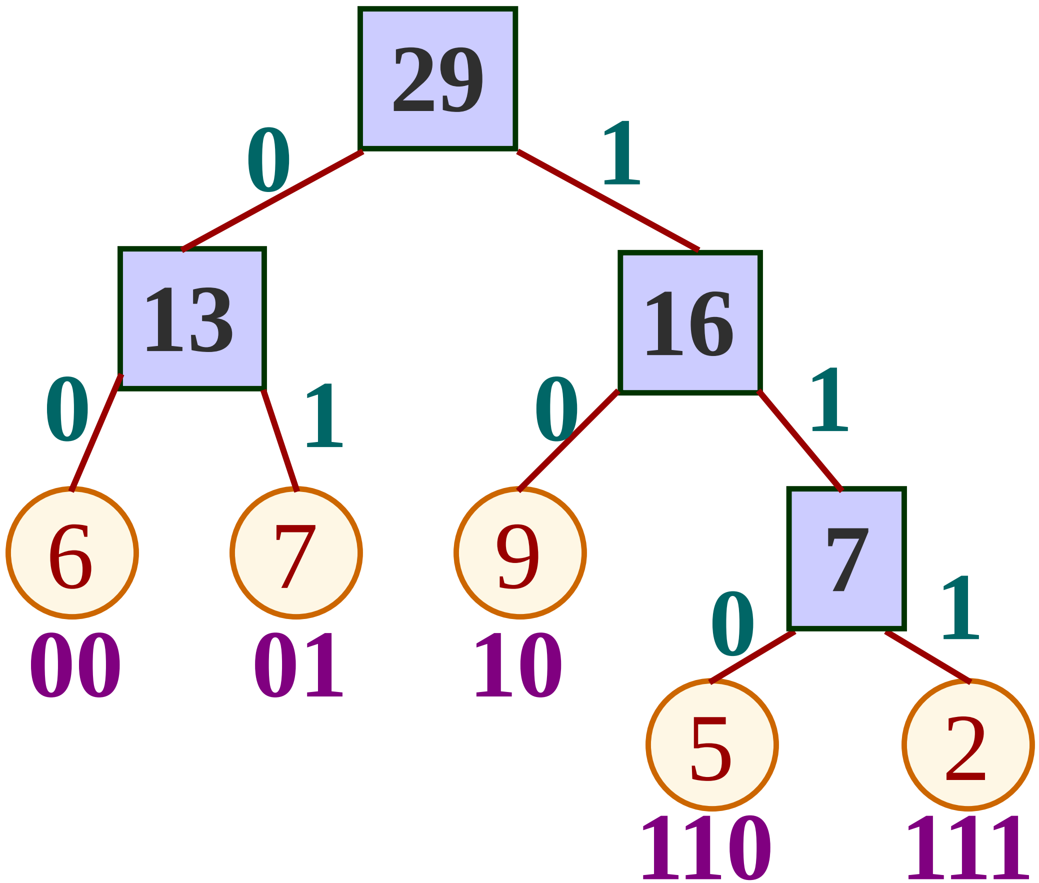
- 贪心对动态规划
- 是否满足贪心选择性
  - 0-1背包问题与分数背包问题
  - 分数背包问题满足：按照每单位价值降序装入商品即可



分数背包问题满足　0-1背包问题不满足

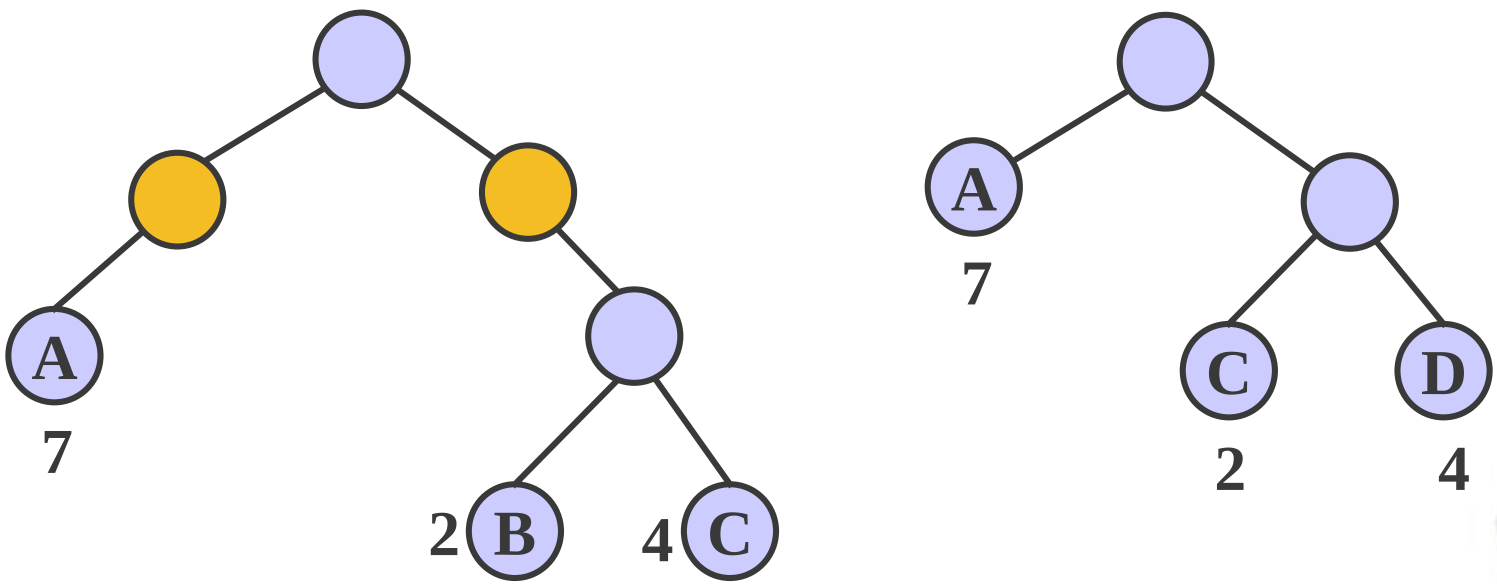
3 哈夫曼树

- 例如：已知权值 W={ 2, 5, 6, 7, 9}



哈夫曼算法的正确性(《算法导论》)

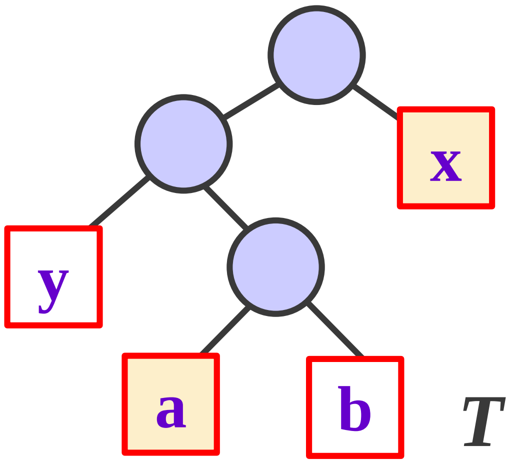
- 0）证明哈夫曼树一定是正则的二叉树
- 1）证明哈夫曼树问题具有贪心选择性质
- 2）证明哈夫曼树问题具有最优子结构性质



哈夫曼树一定是正则的二叉树

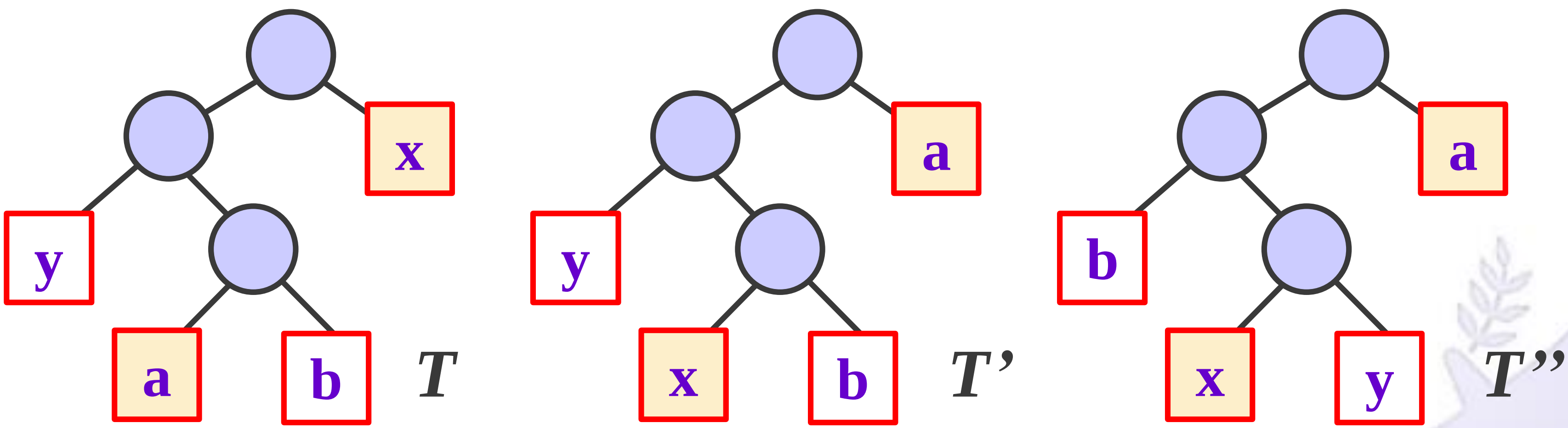
哈夫曼算法的正确性

- 1）证明哈夫曼编码问题具有贪心选择性质
- 引理16.2：令C为一个字母表，其中每个字符c的频率为f_c。令x和y是C中频率最低的两个字符，设f_x<=f_y。那么存在C的一棵最优树，其中x和y互为兄弟节点。
- 证明：令T是任意一个哈夫曼树，令a和b是T中深度最大的兄弟叶节点，设f_a<=f_b。则有f_x<=f_a且f_y<=f_b。



哈夫曼算法的正确性

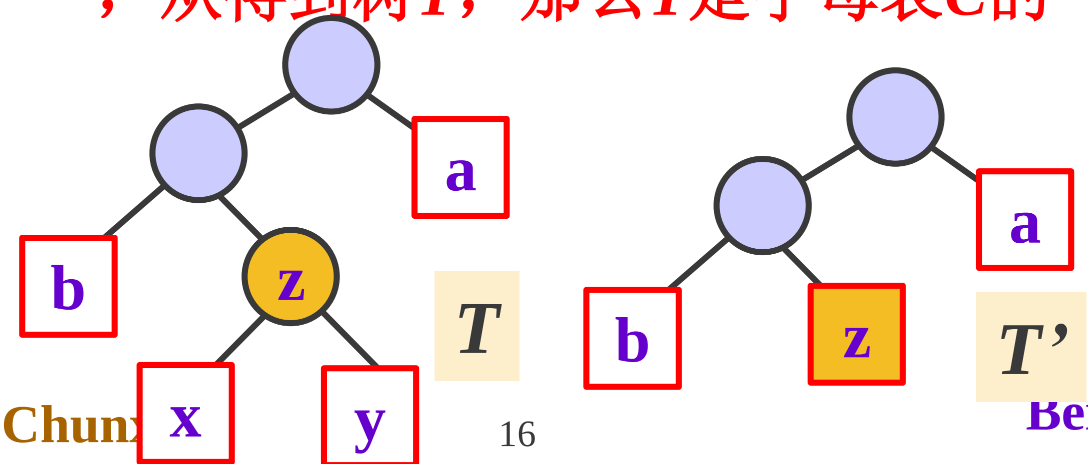
- *T*中交换节点*a*和*x*得到*T'*, *T'*中交换节点*b*和*y*得到*T''*.
- 则树*T*和*T'*的代价满足：*WPL*(*T*) >= *WPL*(*T'*) ;
- 树*T*和*T''*的代价满足：*WPL*(*T*) >= *WPL*(*T''*) ;
- 又由于*T*是最优树，所以*WPL*(*T''*) = *WPL*(*T*)，即*T''*也是最优树。



$f_x<=f_a$且$f_y<=f_b$

哈夫曼算法的正确性

- 2）证明哈夫曼树问题具有最优子结构性质
- 引理16.3：
  - 令C为一个字母表，其中每个字符c的频率为f_c。
  - 令x和y是C中频率最低的两个字符，设$f_x<=f_y$。
  - 令C'为去掉字符x和y，加入新字符z得到的（$f_z= f_x+ f_y$）。
  - 令T'为C'的任意一个哈夫曼树。
  - 则：将T'中叶节点z替换为一个以x和y为子节点的内部节点，从得到树T，那么T是字母表C的一棵哈夫曼树。



$WPL(T) =$
$WPL(T') + f_x + f_y;$

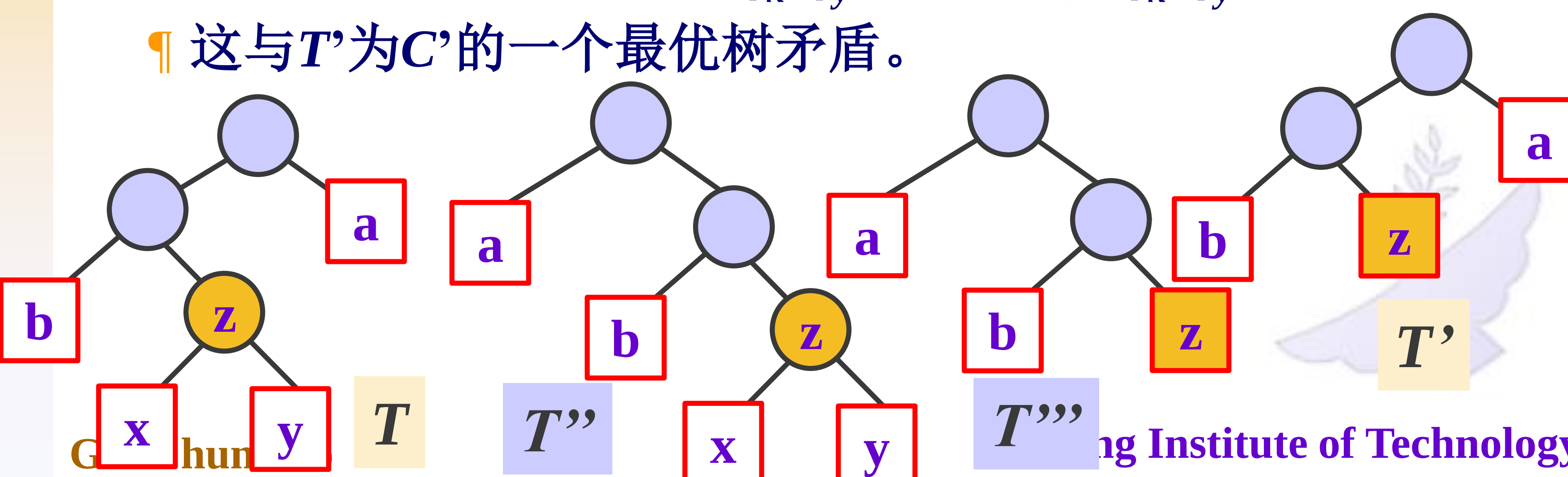
哈夫曼算法的正确性

$WPL(T) = WPL(T') + f_x + f_y$;

- 证明：反证法
  - 1）假设T不是C的哈夫曼树，必存在一棵最优树T''，根据贪心选择性，T''包括频率最小的节点x和y，且为兄弟节点则：　　$WPL(T'') < WPL(T)$
  - 2）令T'''为将T''中x和y及其父节点替换为叶节点z得到的树，则：

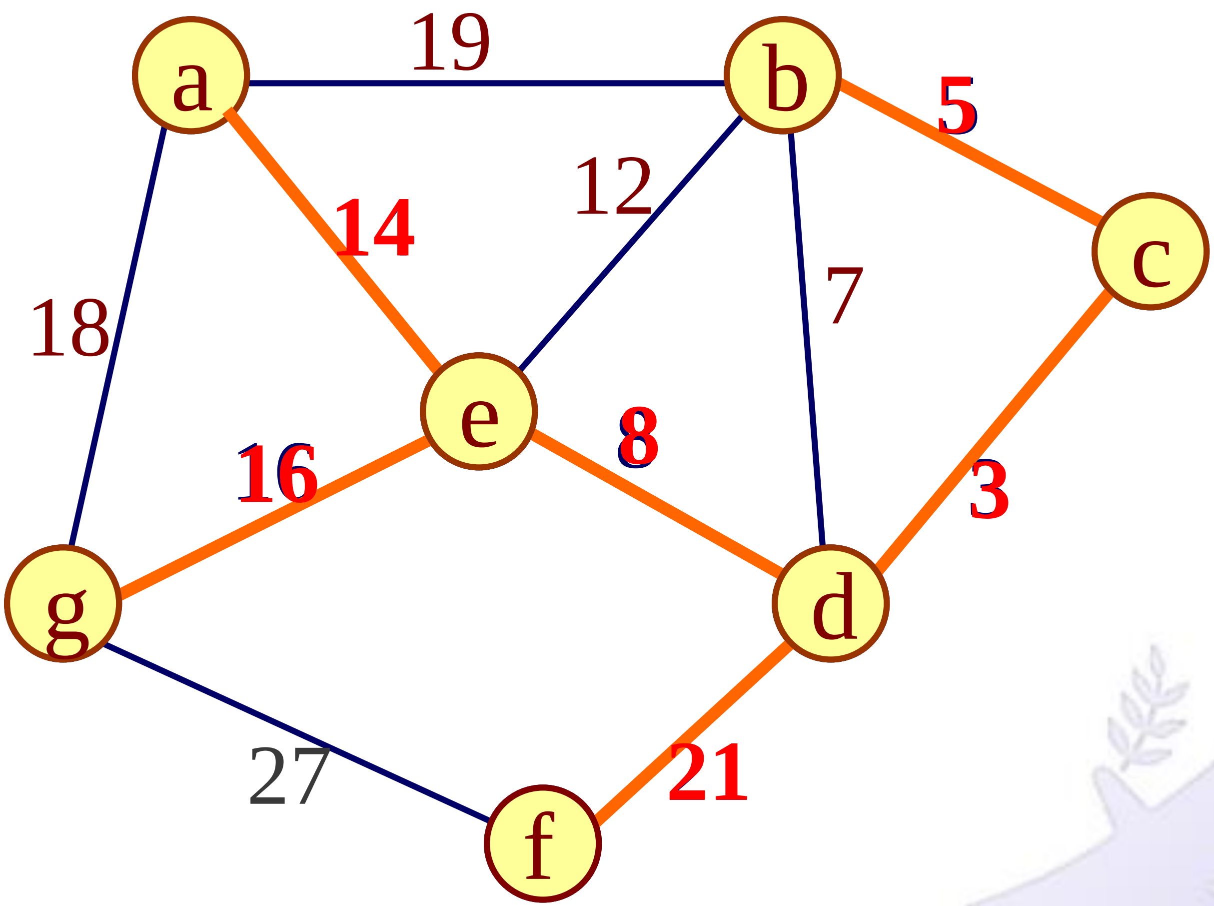
$$WPL(T''') = WPL(T'') - f_x - f_y < WPL(T) - f_x - f_y = WPL(T')$$

  - 这与T'为C'的一个最优树矛盾。



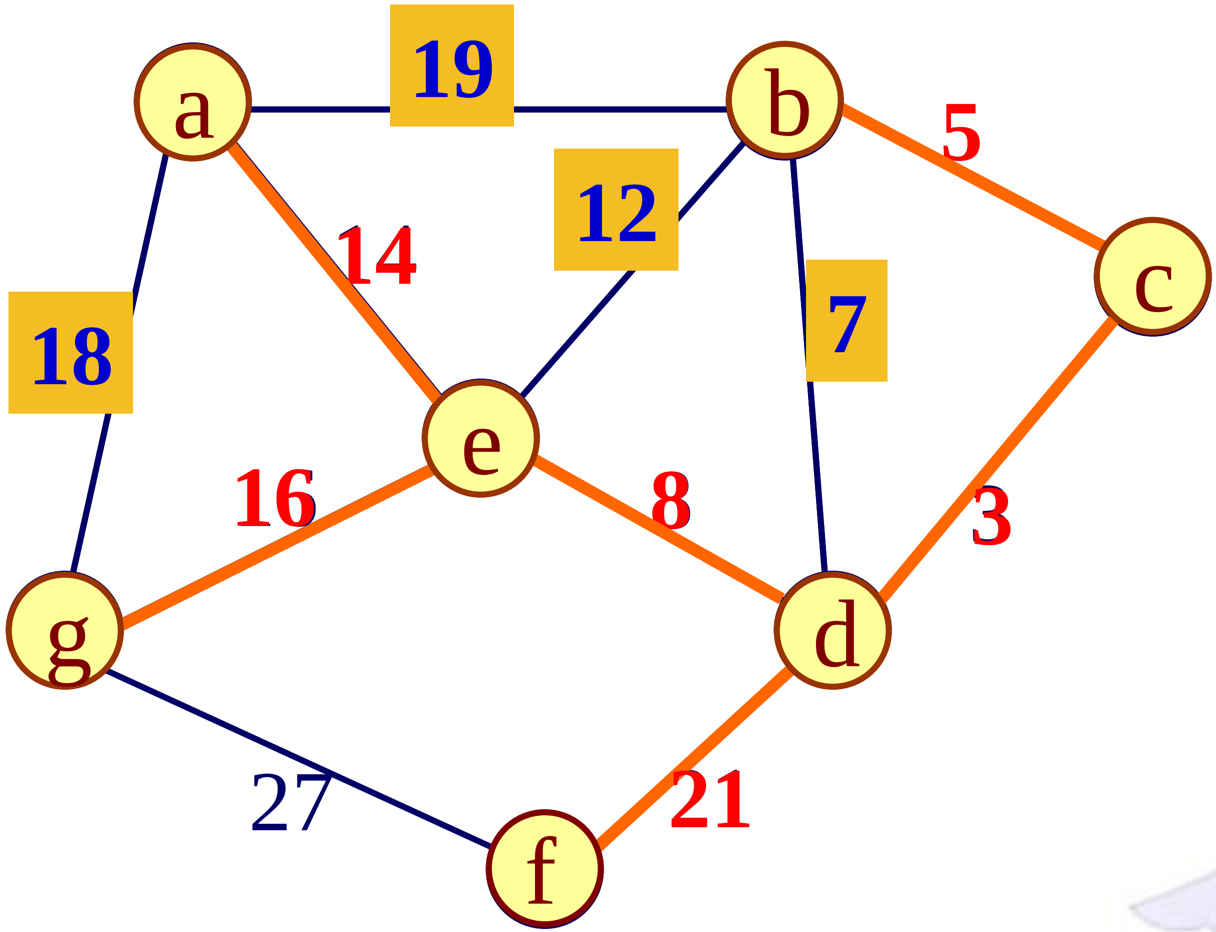
4 最小生成树

- **Prim**



4 最小生成树

- **Kruskal**



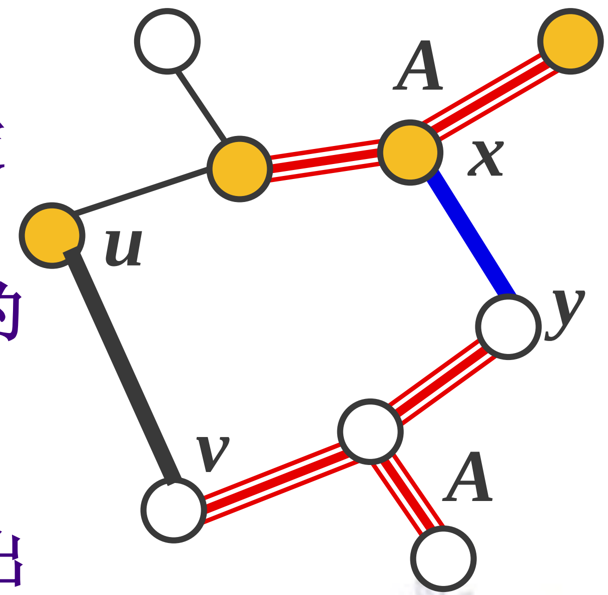
4 最小生成树

- 1）证明最小生成树问题 具有贪心选择性质
  - 即证明贪心选择的边属于某棵最小生成树

- 2）证明最小生成树问题 具有最优子结构性质
  - 即证明加入新选择的边形成的部分是最小生成树的一部分。

4 最小生成树

- 1）证明具有贪心选择性质
- 即证明贪心选择的边属于某棵最小生成树
- 定理23.1
  - 设$G=(V, E)$是一个连通无向图，边上定义了实数权重函数
  - 设集合A是E的一个子集，且包括在G的某棵最小生成树T中。
  - 设$(S, V-S)$是G的一个切割，且A的边要么在S的导出子图中，要么在$V-S$的导出子图中。
  - 设(u, v) 是横跨切割$(S, V-S)$的一条权重最小的边。
- 那么(u, v) 包含在G的某棵最小生成树中



4 最小生成树

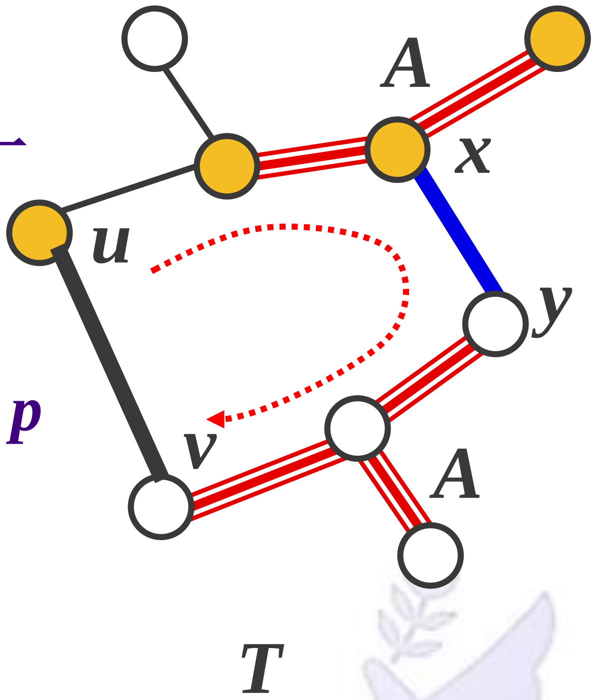
- 证明：
  - 设T是包含A的最小生成树
  - 若T包含边(u, v)，则得证
  - 若T不包含边(u, v)，则T中必定包含一条边连通S和$V\text{-}S$，设为(x, y). 显然

$$w(u, v) <= w(x, y)$$

  - 则T中必定存在一条从u到v的路径p，p包含边(x, y)。
  - 令树$T' = T - \{(x, y)\} \cup \{(u, v)\}$，则

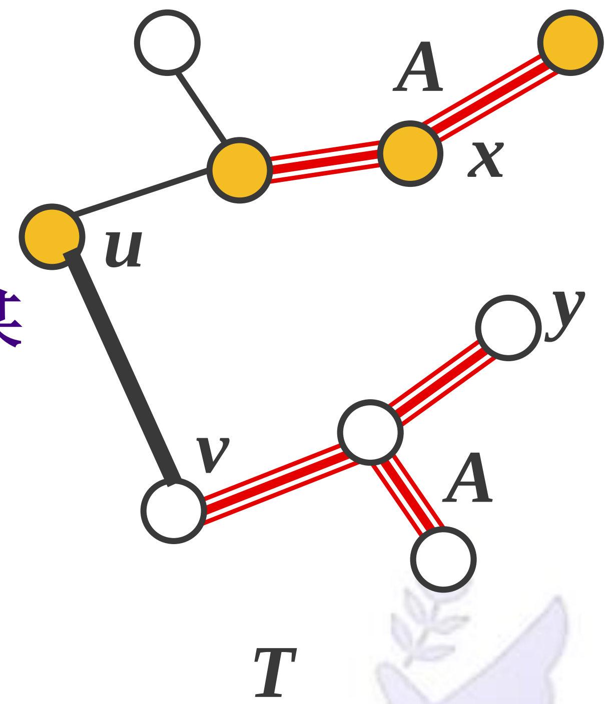
$$w(T') <= w(T)$$

- 所以T'也是G的最小生成树



4 最小生成树

- 2）证明具有最优子结构性质
- 即证明加入新选择的边形成的部分是最小生成树的一部分。
- 证明：
  - 已知A属于某棵最小生成树T，$A \subseteq T$
  - 由上述证明1）知：$A \cup \{(u, v)\}$属于某一棵最小生成树，$A \cup \{(u, v)\} \subseteq T'$



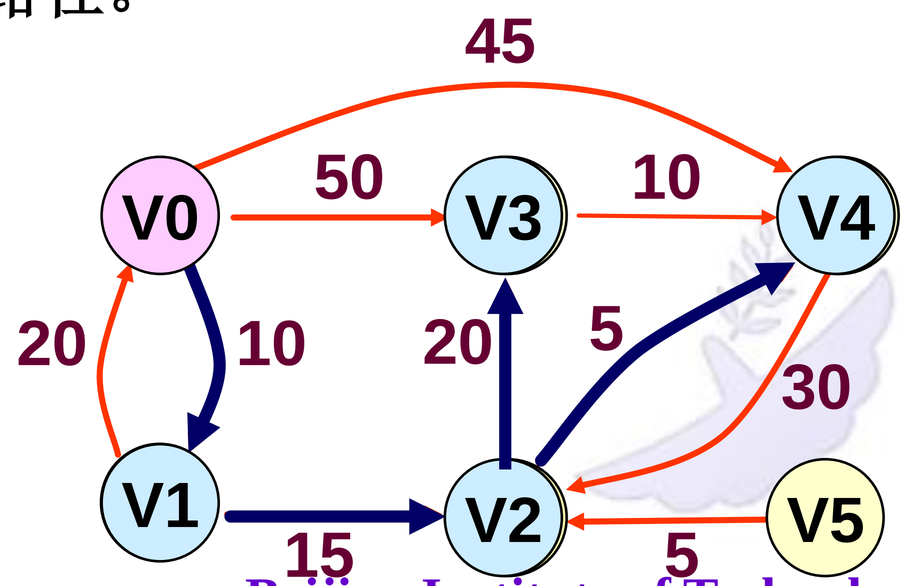
单源最短路径问题（Dijkstra算法）

- 1）证明单源最短路径问题 具有贪心选择性质
  - 证明：每次选择的点即得到起点到该点的最短路径

- 2）证明单源最短路径问题 具有最优子结构性质
  - 证明：最短路径的子路径也是最短路径

单源最短路径问题（Dijkstra算法）

- 2）证明最优子结构性质
- 引理24.1（最短路径的子路径也是最短路径）
  - 设$G=(V, E)$是带权重的有向图.
  - 设$p = (v_0, v_1, \ldots, v_k)$是从$v_0$到的$v_k$一条最短路径。
  - 设$p_{ij} =(v_i, v_{i+1}, \ldots, v_j)$ $(0<=i<=j<=k)$是p中从v_i到v_j的子路径。
- 那么p_{ij} 是G中从v_i到v_j的最短路径。
- 证明：反证法
  - 若p_{ij} 不是最短路径，
  - 则存在一条最短路径p'_{ij}
  - 将p中的p_{ij} 替换为p'_{ij}后
  - 将得到更短的路径p'。
  - 与是p最短路径矛盾。



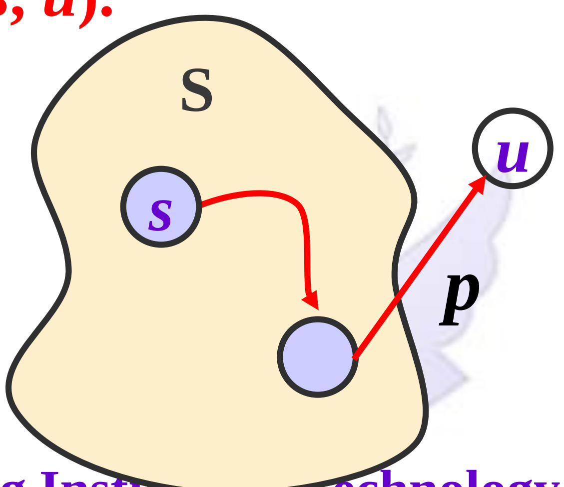
回顾：Dijkstra算法

- 辅助集合S：
  - 当前已经得到最短路径的顶点集合
  - 初始时，S={V0}
- 辅助数组Dist
  - 求解过程中，Dist[k] 表示 “当前”所求得的从源点到顶点 k 的最短路径
  - Dist[k] = <源点到顶点 k 的弧上的权值>
  - 或者Dist[k] = <源点到顶点j的路径长度> +

    + <顶点j到顶点 k 的弧上的权值>

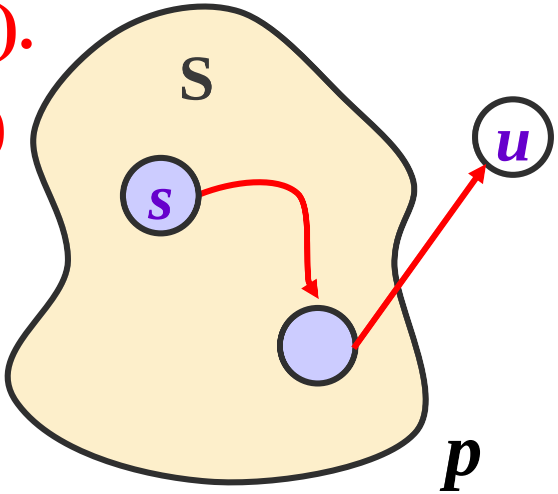
单源最短路径问题（Dijkstra算法）

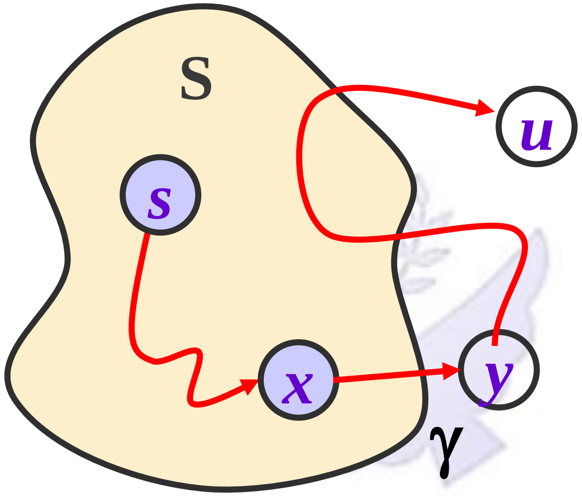
- 1）证明具有贪心选择性质
- 即证明：从Dist[ ] 中选择具有最小值的节点u时就得到从s到的u一条最短路径，即$d[u]=\delta(s, u)$.
- 证明：（数学归纳法）
  - 归纳基础: 初始$S=\{s\}$, $d[s]=\delta(s, s)=0$, 性质成立.
  - 归纳假设: 在某一步性质成立, 即 $\forall v\in S$, $d[v]=\delta(s, v)$.
  - 归纳证明: 从Dist[ ] 中取出u 时$d[u]=\delta(s, u)$.
- 反证法:
  - 取出u时得到一条路径p，长度为$d[u]$
  - 假设p不是最短路径，那么$d[u]>\delta(s, u)$.



单源最短路径问题（Dijkstra算法）

- 归纳证明: 从Dist[ ] 中取出u 时$d[u]=\delta(s, u)$.
- 反证法: 假设p不是最短路径，$d[u]>\delta(s, u)$
  - 设γ是s到u的最短路径：s-x-y-u
  - 设y是γ中第一个出现在V-S中的节点
  - x是与S中与y邻接节点，$d[x]=\delta(s, x)$.
  - 在最短路径γ中y位于u之前，所以
  - $d[y] =\delta(s, y) \le \delta(s, u) \le d[u]$
  - 因为选择的是节点u，$d[u] \le d[y]$
  - 因此$d[y] =\delta(s, y) = \delta(s, u) = d[u]$.
  - 与假设$d[u]>\delta(s, u)$矛盾





贪心算法原理

- 贪心算法的设计步骤
  I. 将最优化问题转化为下面的形式：做出一次选择后，只剩下一个子问题需要求解
  II. 证明算法正确性：
     a) 贪心选择性：即可以通过做出局部最优选择来构造全局最优解。
     b) 证明最优子结构性：一个问题的最优解包含其子问题的最优解。

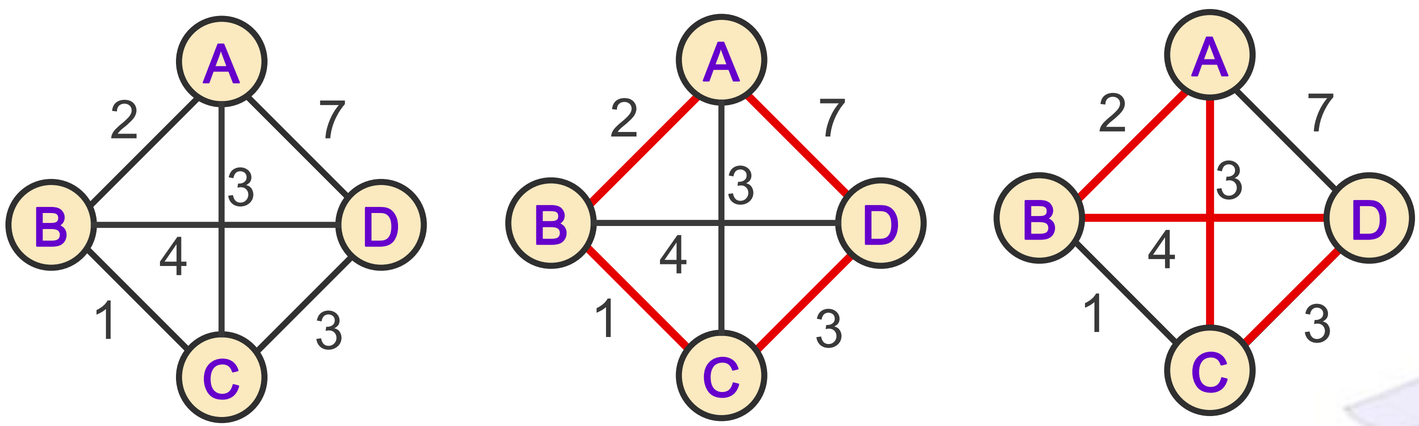
  即做出贪心选择后，剩余子问题的最优解与贪心选择组合即可得到原问题的最优解。

构造“贪心”反例

- **找零问题：**一出纳员支付一定数量的现金。假设他手中有各种面值的纸币和硬币，要求他用最少的货币数支付规定的现金
- 例如，现有4种硬币：它们的面值分别为1分、2分、5分和1角，要支付2角5分
- 首先支付2个1角硬币，然后支付一个5分硬币，这就是贪心策略
- 反例：三种：1、4、6，支付8

构造“贪心”反例

- **货郎担（TSP）问题：**设售货员要到五个城市去售货，最后再回到出发的城市，已知从一个城市到其他城市的费用，求总费用最少的路线.
- **贪心策略：**采用类似MST的策略
  - **BC，BA，CD，AD：13**
- **最优解：12**



作业题

- 1. 字符a~h出现的频率恰好是前8个Fibonacci数,
  - 它们的Huffman编码是什么?
  - 将结果推广到n个字符的频率恰好是前n个Fibonacci数的情形.
- 2. 若在0-1背包问题中, 各物品依重量递增排列时, 其价值恰好降序排列, 对这个特殊的0-1背包题, 设计一个有效算法找出最优解, 并说明算法的正确性.

作业题

- **3. 最优分解问题.**
- **问题描述:**设**n**是一个正整数,将**n**分解为若干互不相同的自然数之和,且使这些自然数的乘积最大.
- **算法设计:**对于给定的正整数**n**,计算最优分解方案.
- **数据输入:**正整数**n.**
- **结果输出:**将计算的最大乘积输出到文件**output.txt**
- 例如若**n=10**, 则最优分解为**2+3+5**, 最大乘积为**30.**

分发饼干

- **问题描述：假设要给你的孩子们一些小饼干。但是，每个孩子最多只能给一块饼干。对每个孩子 i，都有一个胃口值 gi，这是能让孩子们满足胃口的饼干的最小尺寸；并且每块饼干 j，都有一个尺寸 sj 。如果 sj >= gi，我们可以将这个饼干 j 分配给孩子 i，这个孩子会得到满足。你的目标是尽可能满足越多数量的孩子，并输出这个最大数值。**
- 解题思路，首先需要将胃口值和饼干尺寸由小至大排序。设定一个计数器child，用来记得到满足的孩子个数，再维护一个饼干指针cookies。如果饼干尺寸可以满足孩子胃口值，即g[child]<=s[cookies]，就将child、cookies分别加一(向后移动一位)，否则只将cookies向后移动一位。因为孩子的胃口值是由小到大的，若不满足当前的胃口值更不会满足之后的。

55. 跳跃游戏

- 题目描述：给定一个非负整数数组，你最初位于数组的第一个位置。数组中的每个元素代表你在该位置可以跳跃的最大长度。判断你是否能够到达最后一个位置。

435.无重叠区间

- 题目描述：给定一个区间的集合，找到需要移除区间的最小数量，使剩余区间互不重叠。
- 注意：
- 可以认为区间的终点总是大于它的起点。
- 区间 [1,2] 和 [2,3] 的边界相互“接触”，但没有相互重叠。

376. 摆动序列

- 如果相邻数字之间的差严格地在正数和负数之间交替，则数字序列称为摆动序列。第一个差（如果存在的话）可能是正数或负数。少于两个元素的序列也是摆动序列。
- 例如， [1,7,4,9,2,5] 是一个摆动序列，因为差值 (6,-3,5,-7,3) 是正负交替出现的。相反, [1,4,7,2,5] 和 [1,7,8,6,4,2,3] 不是摆动序。
- 给定一个整数序列，返回作为摆动序列的最长子序列的长度。 通过从原始序列中删除一些（也可以不删除）元素来获得子序列，剩下的元素保持其原始顺序。
- 输入：[1,7,4,9,2,5] ， 输出 6
- 输入：[1,4,7,2,5]，输出4
- 输入：[1,7,8,6,4,2,3] ，输出4

餐馆问题

- 题目：某餐馆有n张桌子，每张桌子能容纳的最大人数为a;有m批客人，每批客人有两个参数：b人数，c预计消费金额数。在不允许拼桌的情况下，请选择其中一部分客人，使得总预计的消费金额最大。
- 输入：总共输入m+2行
- 第一行输入n m,即桌子的个数和客人的批数（1<=n<=50000;1<=m<=50000）
- 第二行是n个a,即每张桌子能容纳的客人数
- 接下来m行输入：b c,即每批客人的人数和预计消费金额
- 输出：一个整数，即总预计消费金额
- 测试实例：3 5
- 2 4 2

- 思路：先用贪心算法对每批客人的消费金额进行降序排序，对金额相同的客人的人数进行升序排序；然后用二分法枚举每批客人去最合适的桌子。