贪心算法

12分8分给证明，是否可以利用贪心算法求解，证明其具有贪心选择性质和最优子结构性质，也会要求写伪码

1. 贪心算法概述

1.1. 算法概念

1 核心思路：不从整体最优考虑，所作的选择只是某种意义上的局部最优

2 基本思路：

1. 从问题的一个初始解出发，逐步逼近给定的目标，以尽可能快地求得更好的解
2. 当达到某算法中的某一步不能再继续前进时，算法停止

3 算法存在的问题：

1. 最终得到的可能是整体最优，也可能是整体最优的局部近似
2. 不能用来解最大/最小解问题，只能用来求满足某些约束条件的可行解范围

4 如何判断一个问题是否要用贪心算法：证明每一步所作的贪心选择最终能够导致问题的最优解

1. 用最优子结构证明：作了贪心选择后，原问题简化为一个规模更小的类似子问题
2. 用数学归纳法证明：通过每一步作贪心选择，最终可得到问题的一个整体最优解

1.2. 算法要素

1 贪心选择性质：

1. 贪心选择：局部最优的选择
2. 第一要素：整体最优可通过一系列贪心选择得到(根本要素)

2 最优子结构的性质

1. 最优子结构：一个问题的最优解包含其子问题的最优解
2. 问题具有最优子结构性质 $\Longleftrightarrow$ 问题可用动态规划/贪心求解

2. 算法实例

2.1. 活动安排问题

1 问题描述

1. 有n个活动$E=\{1,2,...,n\}$需要使用同一临界资源
2. 每个活动都有起始s_i和结束f_i时间
3. 活动i,j相容 $\Longleftrightarrow$ 若区间与$[s_i,f_i)$区间$[s_j,f_j)$不相交
4. 要选择一个互相兼容的活动组成的最大集合

2 算法分析：

1. 核心思想：每次都**贪心地**选择下一个与已选活动集合相容且最早结束的活动

2. 数据结构以及代码实现
    - 每个活动的开始/结束时间，存储在`s`和`f`中
    - 活动按照结束时间排序，`f[i]<=f[i+1]`
    - 用bool型数组`A`表示，`A[i]=1`表示活动`i`被选中

```
template<class Type>
void GreedySelector(int n, Type s[], Type f[], bool A[])
{
    A[1]=1;//它选择第一个活动，因为它最早结束
    int j=1;
    for(int i=2; i<=n; i++)//遍历剩下的活动
    {
        //如果活动i的起始时间s[i]要晚于上一个被选中的活动j的结束时间f[i]
        //那么活动ij就相容，可以选择活动i，即A[i]=1，更行j=i
        if(s[i]>=f[j]) {A[i]=1; j=i; }
        //否则的话，就补选中活动i
        else A[i]=0;
    }
}
```

3. 复杂性分析：当给出活动排序好后，$O(n)$时间就可安排n个活动，不排序的话就要$O(n\log n)$

3 **贪心选择性质证明**：存在一个最优解以贪心选择开始，这里是要证明存在一个最优解从选择1开始

1. 假设存在一个最优解：即包含最大数量相容活动的集合，但是不包含1
2. 由于活动1结束最早：将活动1加入这个假设的最优解，不会和其他活动冲突，所以假设不成立
3. 所以最优解必须从1开始

4 **最优子结构性质**：证明A是最优解的话，$A'=A-\{1\}$是$E'=i\in E:s_i\geq f_1$的最优解

1. 假设存在B'为E'的更优解，那么B'比A'包含更多的活动
2. 那么$B=B'+\{1\}$包含比A更多的活动，所以A就不是最优解了
3. 所以假设不存在，A'就是E'的最优解

2.2. 分数背包问题

1 问题描述

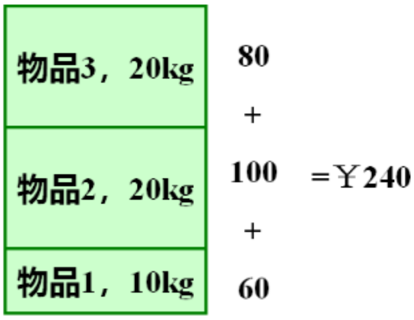
1. 与0-1背包问题的区别：0-1背包问题中只能选择物品全放进去/全不放进去，分数背包问题可以将物体一定比利放入
2. 分数背包问题满足贪心选择性质+最优子结构性质，0-1背包就不满足

2 题目描述：背包容量为50千，要求所装物品价值最大

物品	重量kg	价值 (元)	元/kg
1	10	60	6

物品	重量kg	价值 (元)	元/kg
2	20	100	5
3	30	120	4

3 贪心策略：先把值钱的往里塞，塞完最值钱的再塞第二值钱的，如图



4 算法描述

```
void Knapsack(int n, float M, float v[], float w[], float x[]) {
    sort(n, v, w); // 按照单位价值从高到低排序

    int i;
    for (i = 1; i <= n; i++)  x[i] = 0; //初始化数组x，其中x[i]表示第i个物品被装入的比例。

    float c = M; // c代表当前背包的剩余容量，初始值为M。

    for (i = 1; i <= n; i++)  { // 遍历所有物品。
        if (w[i] > c)  break; // 如果当前物品重量大于背包剩余容量，跳出循环。
        x[i] = 1; // 如果当前物品可以完全装入背包，则标记为完全选中（1表示100%选中）。
        c -= w[i]; // 从背包容量中扣除已经装入的物品重量。
    }

    if (i <= n)  x[i] = c / w[i]; // 如果循环中断了，说明背包还有空间，但无法装下整个物品i。这时，按比例装入物品i的一部分。
}
```

2.3. Haffman编码

2.3.1. 前缀码

1 用一串0101来编码一个字符，0101吗可以变长也可以定长，变长时字符出现频率越高变产码越短

2 非前缀特性：

1. 含义：没有任何一个字符的编码，是另一个字符编码的前缀，例如a的编码为01，那么其他字符的编码肯定不是01XXX
2. 应用：前缀码可以从左到右依次识别出一个字符，无需符号间隔，例如`001011101`为`aabe`

	a	b	c	d	e	f
变长码	0	101	100	111	1101	1100

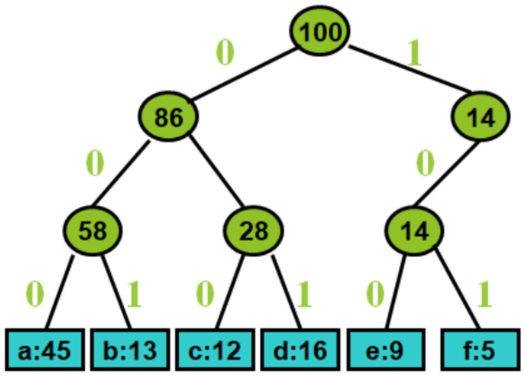
3 Huffman编码：根据字符出现的频率动态生成最优的前缀码

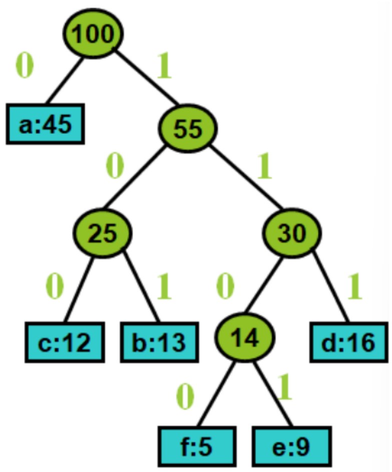
2.3.2. 前缀码的二叉树表示

1 叶节点表给定的字符，非叶节点代表具有相同前缀的字符的出现频率

2 从根到叶的一条道路，算作对应叶的前缀码，道路上遇0左拐遇1右拐

3 左边为定长码，右边为变长码





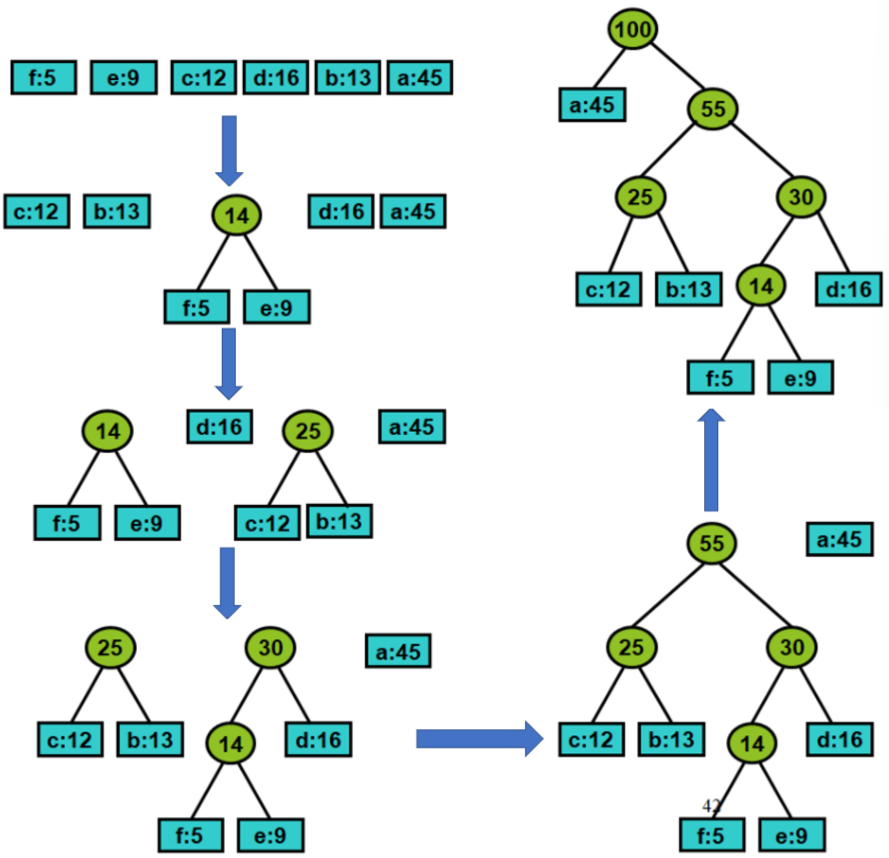
2.3.3. 构造哈夫曼编码

1 核心：自底向上，构造表示最优前缀码的二叉树T

2 构造过程：

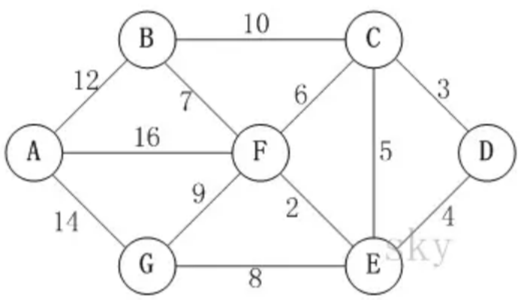
1. 用某字符ch的出现频率$f(ch)$
2. 根据所有字符的$f(ch)$值，排列出一个优先队列Q，频率最低的排最前面
3. 根据贪心选择，选取合并后频率最低的两个树合并，然后更新优先队列
4. 重复上述过程

3 示例



2.4. 单源最短路径

1 问题概述：带权有向图中，求原点到某个点的最短路径，用Dijkstra算法(就是一种贪心算法)



2 算法的数据结构&初始化

1. 图G，源节点v_0
2. 引进集合S：S记录已求出最短路径的顶点，初始化$S = \{v_0\}$

3 算法执行步骤示例

步骤	操作	集合S的更新	A	B	C	D	E	F	G
1	初始化距离	S = {D}	*	*	3	0	4	*	*
2	选择最短距离的点C	S = {D, C}	*	13	3	0	4	9	*
3	选择最短距离的点E	S = {D, C, E}	*	13	3	0	4	6	12

步骤	操作	集合S的更新	A	B	C	D	E	F	G
4	选择最短距离的点F	S = {D, C, E, F}	22	13	3	0	4	6	12
5	选择最短距离的点G	S = {D, C, E, F, G}	22	13	3	0	4	6	12
6	选择最短距离的点B	S = {D, C, E, F, G, B}	22	13	3	0	4	6	12
7	加入剩余的点A	S = {D, C, E, F, G, B, A}	22	13	3	0	4	6	12

2.5. 最小生成树

2.5.1. 问题描述

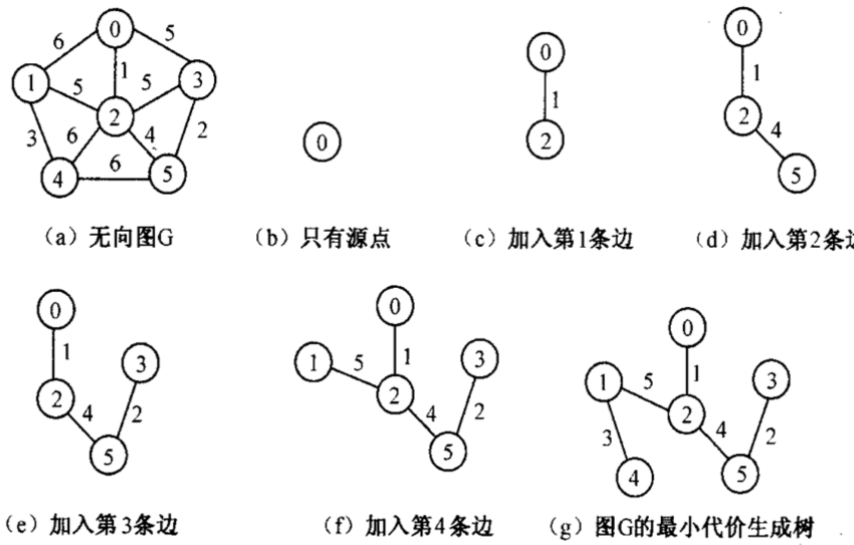
1 数据结构：$G=(V,E)$是无向带权连通图，给每条边一个权值，如边(v,w)的权为$c[v][w]$

2 G'是G的生成树：满足G'是G的子图，G'是数，G'包含G中所有顶点

3 最小生成树：

1. 生成树的耗费：生成树G'的所有边的权值之和
2. 最小生成树：耗费最小的生成树

2.5.2. prim算法：从点出发，$O(n^2)$



1 设 $N=(V,E)$ 是连通网，$TE \subseteq E$ 是 N 上最小生成树中边的集合

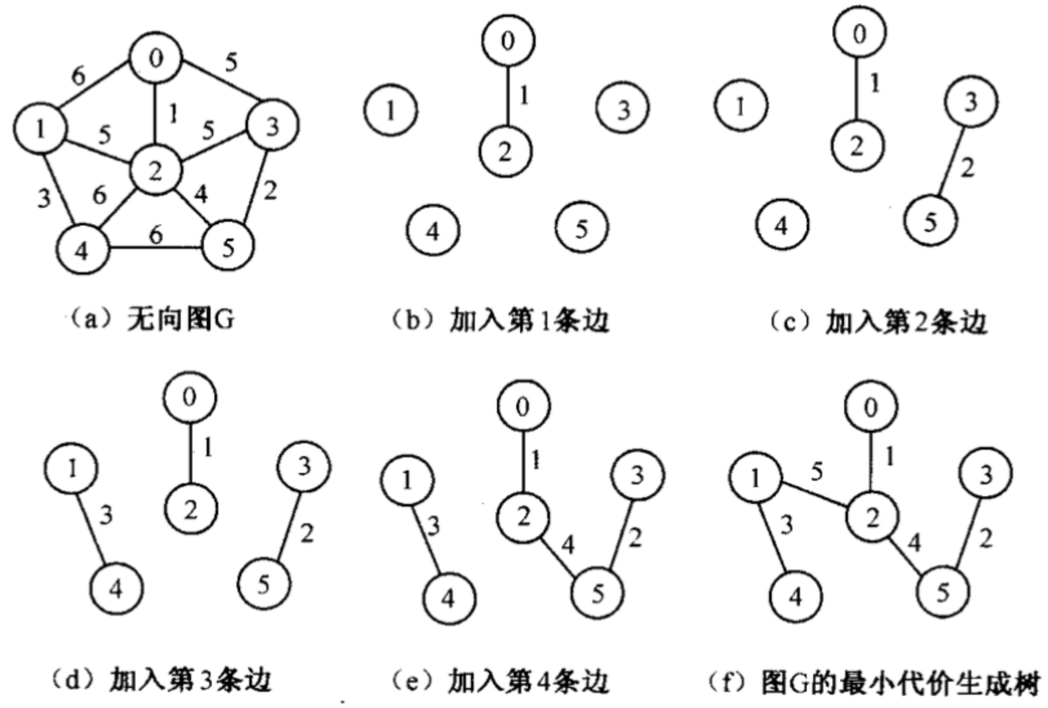
2 初始化 $U=\{u_0\},(u_0 \in V), TE=\emptyset$

3 在所有 $u \in U, v \in V-U$ 的边 $(u,v) \in E$ 中，挑选权值最小的边 (u_0,v_0)

4 将 (u_0,v_0)并入集合 TE，同时 v_0 并入 U

5 重复上述操作直至 $U=V$ 为止，则得到 N 上的最小生成树

2.5.3. Kruskal算法：从边出发，$O(n\log n)$



1. 将图中所有的边按权值从小到大排序
2. 初始化只包含所有顶点的森林
3. 从权值最小的边开始，检查当前边是否会与已经选择的边一起形成环路：
   - 如果不会形成环路，则将这条边加入最小生成树中
   - 如果会形成环路，则忽略这条边，继续检查下一条边
4. 重复步骤3，直到森林变成了树

2.6. 多机调度问题

1. 问题描述
   1. n个作业在尽可能短的时间内，由m台机器
   2. 单个作业不能被拆分为子作业，完成处理前不可以中断
   3. 该问题NP完全，无有效解决方案
2. 算法近似：采用贪心策略——最长处理时间作业优先处理
3. 示例：7个作业三台机器，按如下处理