研究生算法课课堂笔记

上课日期：**2016.10.31**　　　　　　　**第(2)节课**

组长学号及姓名：位冰镇 **1501214415**
组员学号及姓名：黄兴 **1601214434** 刘洲 **1601214513**

一、 内容概要

a) 树形动规：最大独立集的递推公式；

b) Anniversary party 作业题讲解；

c) Bagging 、Random Forest；

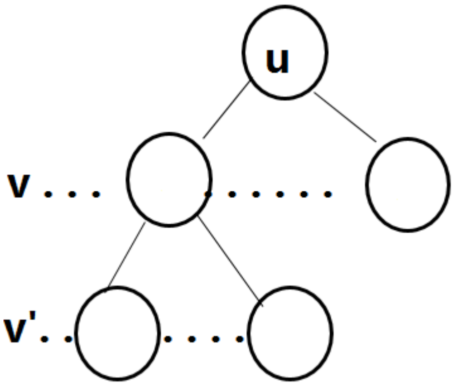
二、 详细内容

a) 树形动规

i. 指导思想： 沿着树的拓扑结构进行动态规划。

ii. 怎样写递推公式？

1. **尝试一：** 如图



求以 u 为根节点的树的权重最大的独立集

这里设 u,v 表示节点，wu 表示节点的权重，f(u)表示以节点 u 为根节点的子树的最优解，设根节点为'root'。。

$$f(u) = \max\begin{cases} w_u + \sum_{v\in children(u)} \sum_{v'\in children(v)} f(v') & \text{包含节点u时} \\ \sum_{v\in children(u)} f(v) & \text{不包含节点u时} \end{cases}$$

Return $f(root)$.

这样写有什么问题吗？

* 我们小组经过讨论认为，这样的递推公式也是可以的。和背包问题有所不同的是，这里节点之间的影响具有局部性，在每一步递推的过程中，每个节点只需要看它的直接前驱与后继是否被选择；而背包问题中的容量这一变量的影响具有全局性，第一个物品的选择与否可能会影响到最后一个物品能不能选（因为可能容量空间不够了），因此我们必须传入容量 W 这一维信息。这里的 f(u)代表以节点 u 为根节点的子树的最优解，包含节点 u 时，其直接子节点不能选，但其孙子节点和 u 是相容的，因此我们把所有孙子节点的最优解和 u 的权重加起来就可以了。

2. 尝试二：

f_{in} 表示包含节点 u 时以 u 为根节点的子树的最优解，f_{out} 表示不包含节点 u 时以 u 为根节点的子树的最优解，设根节点为'root'。

$$\begin{cases} f_{in}(u) = w_u + \sum_{v\in children(u)} f_{out}(v) & \text{包含节点u时} \\ f_{out}(u) = \sum_{v\in children(u)} f_{in}(v) & \text{不包含节点u时} \end{cases}$$

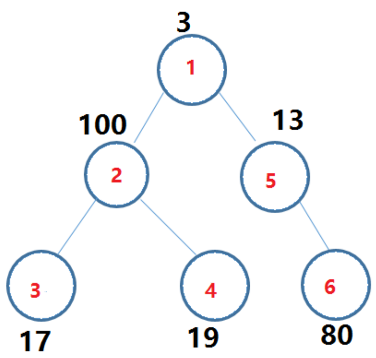
Return $\max\{f_{in}(root), f_{out}(root)\}$ ；

我们通过下面的例题验证这个式子正确与否。

注意： 这里假设所有的权重都大于 0， 和作业题不同。 最大独立集肯定不包含权值为负值的节点， 因为它们的加入非但没有做贡献， 还会影响到其他节点的加入。

b) 例：

i. 公司聚会，为每一个人都分配一个权值，下属和其直接领导避免同时在场，求权值最大的组合。如下：



利用上面的公式,我们可以计算:

	1	2	3	4	5	6
f_{in}	119	100	17	19	13	80
f_{out}	113	36	0	0	80	0

应修改为:**180= 100+80**

初始化

表格中最大值为 119， 但这显然不是最优解， 因为节点 2 和节点 6 组成的独集才是最优解，其权值之和为 180。 那么问题出在哪里？

注意到， 当不包含父节点 1 的时候， 其直接子节点并不一定包含在最优解内，而其孙子节点可能在最优解中。 因此这里我们的子结构应该是以各个子节点为根节点的子树， 然后对这些子树的最优解(也就是最大值)进行求和。

因此， 递推公式应该修改成这样：

$$\begin{cases} f_{in}(u) = w_u + \sum_{v \in children(u)} f_{out}(v) & \text{包含节点u时} \\ f_{out}(u) = \sum_{v \in children(u)} \max\{f_{in}(v), f_{out}(v)\} & \text{不包含节点u时} \end{cases}$$

初始条件： 后序遍历到叶节点时， 如果被选择则初始化为其权重， 否则初始化为 0， 如表格中所示；

返回值： 设根节点为 root， 则 return max{f_{in}(root)， f_{out}(root)} ；

时间复杂度与空间复杂度： 均与树的规模有关， 假设树的节点数为 n， 那么复杂度为 O(n)；

P.S.： 作业题中并不是二叉树， 所以需要自己定义数据结构来维护每个节点的子节点，从而进行遍历。 比如每个节点设置一个 vector 保存其子节点。后续遍历子节点求出 f_{in} 与 f_{out} ，然后再计算父节点的 f_{in} 与 f_{out} , 直到根节点计算完毕返回最大值。

ii. 小问题：

Q1： 如果我们以上讨论的不是树而是有权重的图， 其中存在度为 1 的节点(看成叶子节点)，那这些叶节点还在最大独立集里面吗？

Answer： 和树的情况一样，同样不一定，需要用动态规划求解。

Q2： 在返回值时返回的是根节点的 f_{in}(root)与 f_{out}(root)的最大值，那么如何找到根节点？ 如果找错了根节点，会有什么后果？

Answer： 在这个问题中哪个节点作为根节点并不重要。 因为树的边是无向的，而最大独立集只考虑两个节点是否直接相连，并

不考虑方向，所以并不需要找到根节点。随便找一个节点，"抖一抖"，　就是一棵树。

c)随机森林

1>例：老师录取研究生的时候，怎样录取最好的学生？那在现实生活中怎么更好的提高录取质量呢？

策略：让评审委员会的 n 个老师一起投票。

此策略的优点：

①可以避免老师的偏见（比如有的老师喜欢北大本科的，有的老师有地域偏见）

②可以弥补"每个老师接触到的面比较窄的情况,比如老师不可能了解全国的所有学校。

2>按照个体学习器的生成方式划分为两类：

①个体学习器之间存在强的依赖关系（不独立），必须串行生成的序列化方法（代表：**Boosting**)

② 个体学习器之间不存在强的依赖关系，可同时生成的并行化方法（代表：**Bagging, random forest)**

3> Bagging

基本思想：对训练集进行扰动。

实现方式：有 m 个样本的数据集，有放回的抽取 m 个样本，就得到一个有 m 个样本的采样集，这样进行 T 次，那么就可以用这 T 个采样集分别训练出 T 个基学习器，然后再把这些基学习器结合。

基于以上思想的一些问题：

Q1: m 个样本，有放回的选取 m 个，选中的样本恰好就是原来的 m 个样本的概率是多少？

Answer： $\frac{m!}{m^m}$ （原因：因为有放回的抽 m 次的所有可能情况有 m^m 种，其中把所有 m 个都选中的情况第一次有 m 种选法，第二次有 m-1 种选法......，所以一共有 m*(m-1)*....*1=m! 种选法，所以答案如上。）

Q2: 有的样本没有选中（叫做 out of bag sample 即 OOB），某样本没有被选中概率是多少？

Answer: $\left(1-\frac{1}{\mathrm{m}}\right)^{m}$ （原因：每次都选不中它的概率是$1-\frac{1}{\mathrm{m}}$,那么 m 次选不中它的概率就是$\left(1-\frac{1}{\mathrm{m}}\right)^{m}$）。（当 m 很大时，公式趋近于$\frac{1}{e}\approx 0.37$）。

Q3: Q2 中的概率说明了什么？

Answer: 说明选的某个样本集中有 37%的数据集中的样本并未选中。

Q4: Q3 中剩下的 37%的样本怎么利用起来？有什么作用？

Answer:
- 被抽取的 63%：训练集
- 未被抽取的 37%：验证集

Q5:在老师招收研究生的例子中，上面 Bagging 的方法有何好处？

Answer:对训练集的采样进行了扰动，增加了多样性。

Q6:那么还有什么好的办法可以增加多样性呢？

Answer: 对属性进行扰动。（这样想：不同的老师看重的方面不同）即就是在 Bagging 的方法上做小小的改动，增加了属性扰动，就扩展成了下面的随机森林。

4>随机森林

基本思想：在 Bagging 的基础上加入了属性扰动。

实现方式：对于基决策树的每个节点，从该节点的 n 个属性集合当中随机选择一个包含 k 个属性的子集，再从这个子集中找出最优属性进行划分。（这里的 k 值可以取$\sqrt{\mathrm{n}}$，$\log_2 n$，$\frac{\mathrm{n}}{2}$）