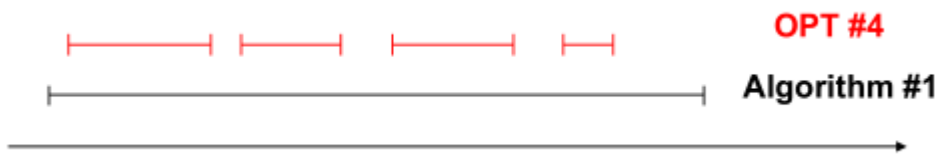
> 贪心算法即不从整体最优考虑，只是做出在当前看来最好的选择，它所做出的选择只是在某种意义上的局部最优选择。贪心算法对于大多数优化问题都能得到整体最优解（如单源最短路径问题、最小生成树问题等），虽然说并不总是这样的。在一些情况下，即使贪心算法不能得到整体最优解，但是其最终的结果是最优解的近似。
>
> 动态规划算法通常以**自底向上**的方式解各子问题，而贪心算法则通常以**自顶向下**的方式进行，每作一次贪心选择就将所求问题简化为规模更小的子问题。
>
> 下面来看一些具体的例子。

区间调度（Interval Scheduling）

问题描述：假如我们有多个任务，每个任务都具有各自的开始时间和结束时间，求在任务不重叠的情况下任务的最大组合数。

根据贪心算法，我们可以从不同的角度分析这个问题。

Rule1：选择开始最早的任务



每次选择当前最先开始的任务，并依次选到最后一个。

从这个例子看出，失败！

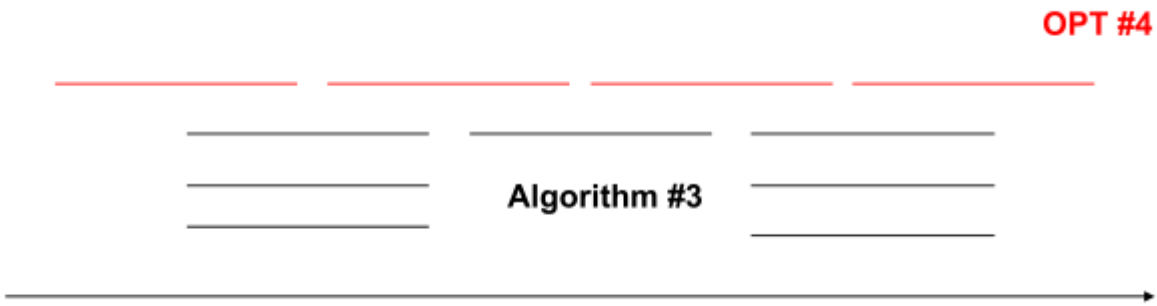
Rule2：选择区间最短的任务

从最短的任务开始选择，并依次选择不重复的当前最短的任务，按照所用时间长短排序。

从这个例子看出，失败！

Rule3：选择冲突最少的任务

对于每一个任务，计算与它冲突的任务个数，每次选择当前与其冲突最少的任务。



从这个例子看出，还是失败了。

Rule4：选择结束最早的任务

每次选择当前最早结束的任务，并依次选到最后一个。

可以看出，在上面的三个例子中，本算法都是可行的。事实上，Rule4的算法是解决区间调度问题的一个可行算法。直观上来说，按这种方法选取的任务为未安排的任务留下了尽可能多的时间。也就是说，该算法的贪心选择使剩余的可安排时间段极大化，以安排尽可能多的不重叠活动。这个算法的时间复杂度是$O(nlogn)$。

集合覆盖（Set Cover）

问题描述：在一个集合B以及B内元素构成的若干集合$S_1, S_2, \ldots, S_m$中，找到数目最少的S_i使得S_i中的所有元素都包含了B中所有元素。为便于理解，给一个具体的例子。例如，B={1,2,3,4,5}，S_1={1,2,3}，S_2={2,4}，S_3={3,4}，S_4={4,5}，可以找到一个集合覆盖S_1, S_4。

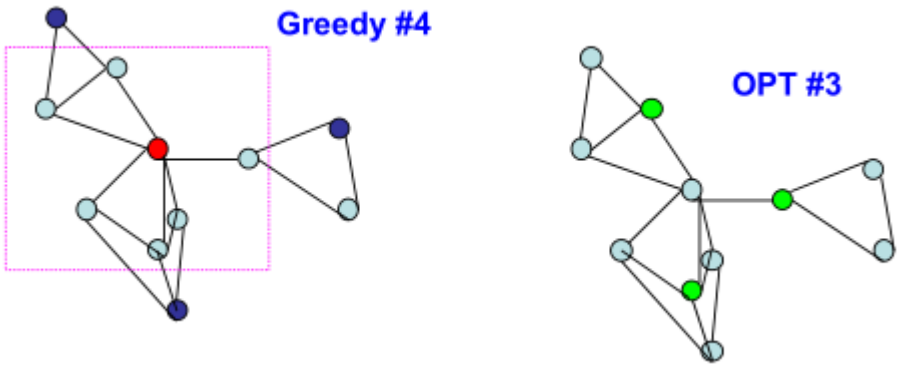
其实集合覆盖问题是一个NP难的问题，但是我们仍然可以用贪心算法得到这个问题的近似解。

假设我们要在城镇里建几所学校，需要满足两点：1、每个学校都要建在一个城镇里。2、城镇离学校的距离不能超过30英里。

根据贪心算法，我们可以得到非常接近的解：

1. 选出这样一个学校，即它覆盖了数量最多的城镇。
2. 重复第一步，直到覆盖所有的城镇。

这是一种近似算法，贪心算法的运行时间为$O(n^2)$，可得到如下图所示的结果，图中点的位置代表城镇的位置。



最小生成树（Minimum Spanning Tree）

用贪心算法设计策略可以设计出构造最小生成树的有效算法，如Prim算法和Kruskal算法，尽管它们做贪心选择的方式不同。

关于MST的这两个算法相信大家应该都十分熟悉了，这里便不再展开。