算法设计与分析

第4章 贪心算法
(1)

谢晓芹
哈尔滨工程大学计算机科学与技术学院

学习要点

- 理解贪心算法的概念
- 掌握贪心算法的基本要素
  - 最优子结构性质
  - 贪心选择性质
- 理解贪心算法与动态规划算法的差异
- 理解贪心算法的一般理论

学习要点

- 通过应用范例学习贪心设计策略
  - 活动安排问题
  - 最优装载问题
  - 哈夫曼编码
  - 单源最短路径
  - 最小生成树

引言

- 和动态规划算法类似，都用于优化问题
- 优化问题一般都有多个步骤，每个步骤会有多种选择
- 对于很多优化问题，用动态规划算法来找出最优解往往具有过度的杀伤威力
- 贪心算法: 更简单，更有效

引言

- 应用
  - 找钱
    - 四种硬币：1分，5分，1角，2角5分
    - 找六角3分，怎么给？
  - 排队时出现拥挤情况，如何做才能节省用户时间？

引言

- 研究
  - 白梅,王习特,李冠宇,宁博,周新. 基于最大覆盖的代表Skyline问题的优化算法研究,计算机学报, 2020,12, 2276-2297 (提出优化贪心算法OGA和ε-OGA)
  - 杨挺等.云计算数据中心HDFS差异性存储节能优化算法,2019,42(4):721-735
  - 李智慧等.位置敏感的社交网中最小种集选取算法研究.计算机学报,2017,10:2305-2319
  - 马茜,马军. 在影响力最大化问题中寻找种子节点的替补节点. 计算机学报,2017, 3, 674-686

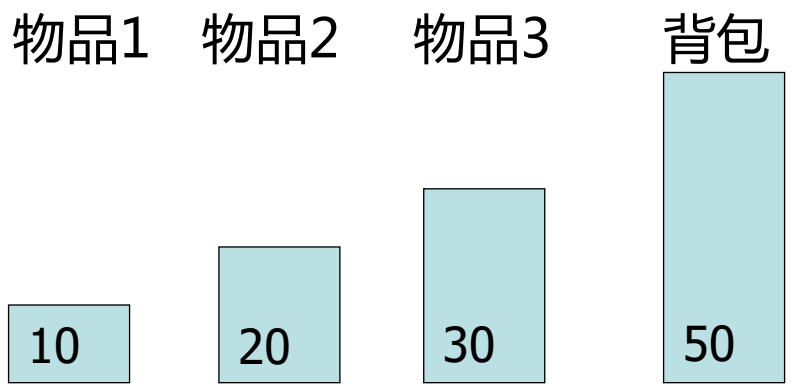
引言-实例

- 0-1背包问题
  - n 个物品, 物品i 价值v_i ，重w_i
  - 背包的最大负载量为W，如何选取物品，使得背包装的物品价值最大
  - 每个物品是一个整体，不能分割，因此，要么选取该物品，要么不选取
- (分数)背包问题：
  - 和0-1背包问题类似, 但是可以取走物品的一部分.
  - 输入：C>0, w_i >0, v_i >0, 1≤ i≤n
  - 输出：(x_1, x_2, …, x_n), 0≤ x_i ≤1, 满足

$$\max \sum_{i=1}^{n} v_i x_i \ , \ \sum_{i=1}^{n} w_i x_i \le C$$

引言-实例

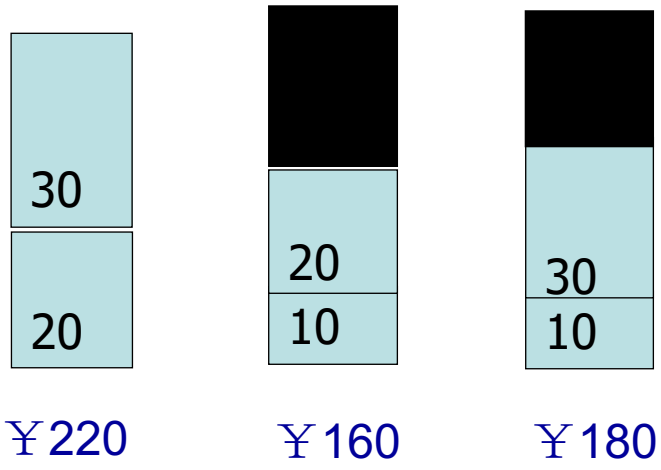
- 实例
  - 三个物品,C=50
    - 重量(10,20,30)
    - 价值(60,100,120)



i	1	2	3
v_i	60	100	120
w_i	10	20	30
v_i/w_i	6	5	4

0/1背包问题

动态规划法

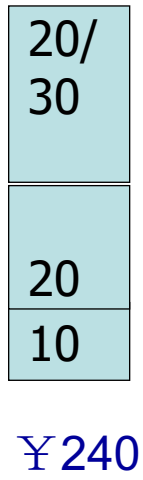


贪单位重量价值？

背包问题

贪心法

（贪单位重量价值）



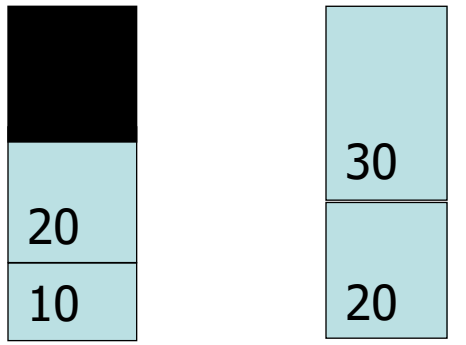
引言-实例

- 贪心策略：贪单位重量价值最大
  - 取物品1 and 2
  - value = 160, weight = 30
  - 为什么不是最优解？
    - 20 pounds 剩余空间.

贪什么？

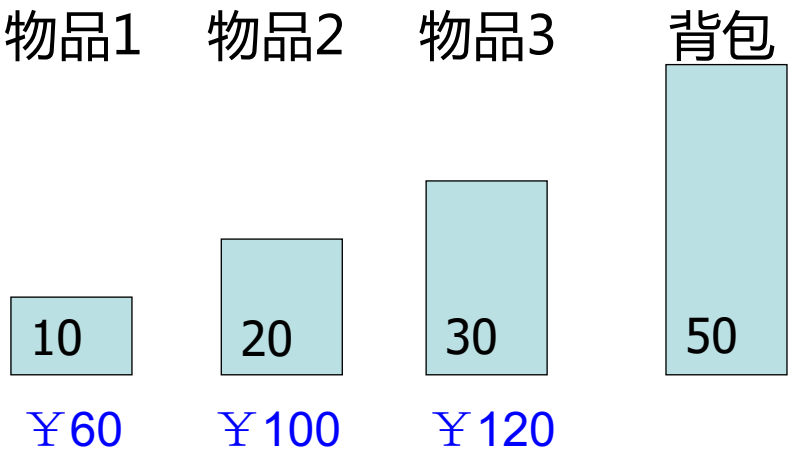
i	1	2	3
v_i	60	100	120
w_i	10	20	30
v_i/w_i	6	5	4

- 0-1 背包问题
  - 没有贪心选择性质
  - 最优策略:
    - 取 物品 2 and 3
    - value=220, weight=50
    - 没有剩余空间

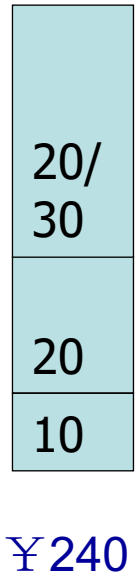


引言-实例

- 分数背包问题的贪心算法
  - 贪什么：贪单位重量价值最大的物品
  - 怎么贪：
    - 按照**单位重量价值**(v_i /w_i)将物品降序排列
    - v_i /w_i ≥ v_{i+1} /w_{i+1} for all i



i	1	2	3
v_i	60	100	120
w_i	10	20	30
v_i/w_i	6	5	4




```
FRACTIONAL-KNAPSACK(v, w, W)
1  load ← 0          //load:目前已装重量
2  i ← 1             //初始选择物品
3  while load < W and i ≤ n{
4        do if w_i ≤ W – load     //判断全装还是装部分
5              then take all of item i
6              else take W-load of w_i from item i
7        add what was taken to load
8        i ←i + 1}
```

贪心算法的基本要素 *what*

- 贪心(Greedy)算法总是作出在当前看来最好的选择。
  - 贪心算法并不从整体最优考虑，它所作出的选择只是在某种意义上的局部最优选择。
  - 希望贪心算法得到的最终结果也是整体最优的。虽然贪心算法不能对所有问题都得到整体最优解，但其最终结果却是最优解的很好近似。

贪心算法的基本要素 *when*

- 贪心算法求解的问题一般具有2个重要的性质
  - 最优子结构性质
    - 每个大问题的最优解里面包括下一级子问题的最优解
  - 贪心选择性质
    - 问题的整体最优解可以通过一系列局部最优的贪心选择来达到
    - 每个小问题可由贪心选择获得

贪心算法的基本要素

- 贪心选择性质
  - 贪心选择性质是指所求问题的**整体最优解**可以通过一系列**局部最优**的选择，即**贪心选择**来达到。这是贪心算法可行的第一个基本要素，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。动态规划算法通常以**自底向上**的方式解各子问题

缩小问题规模的方式是：每次的贪心选择

- 最优子结构性质
  - 当一个问题的最优解包含其子问题的最优解时，称此问题具有最优子结构性质
  - 问题的最优子结构性质是该问题可用动态规划算法或贪心算法求解的关键特征

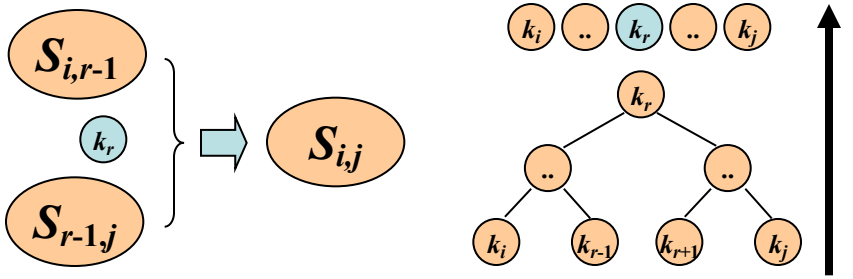
贪心算法的基本要素

- 背包问题实例分析
  - 都有最优子结构性质
    - 0-1：首先选择最有价值的物品 j ，其重量 $w_j \le C$(背包容量)，然后选择剩下物品中最有价值的物品 i，其重量$w_i \le C-w_j$
    - 分数背包: 若先选 item j 的一部分，其重 w ，则接下来的最优挑选方案为从余下的 n-1 个物品〔除 j 外〕 **和 j 的另外重 w_j-w** 的部分中挑选，其重量不超过 C-w
  - 但是分数背包问题具有贪心选择性质，而0-1背包问题没有这个性质
    - 可用贪心方法时，动态规划方法可能不适用 (背包问题)
    - 可用动态规划方法时，贪心方法可能不适用(0/1背包问题)

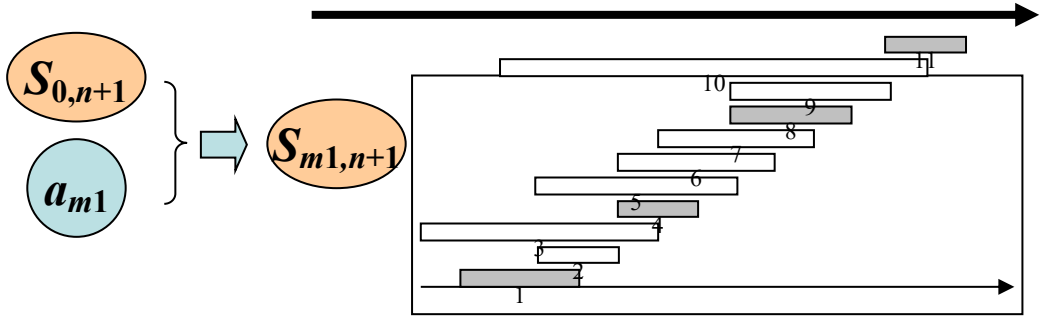
贪心算法的基本要素

- 最优子结构：通过局部最优解可导出全局最优解

- 动态规划
  - 在每一步做出选择
  - **依赖于已知子问题的最优解再作出选择**
  - 自底往上解子问题



- 贪心算法
  - 在每一步做出选择
  - **先作选择，再解子问题**
  - 自上往下解子问题



贪心算法的基本要素 *How*

- 问题的定义
- 最优解的结构分析
- 算法设计
- 算法复杂性分析
- 算法正确性证明

贪心算法的基本要素 *How*

- 贪心算法需要正确性证明
  - ① 证明算法所求解的问题具有最优子结构
  - ② 证明算法所求解的问题具有贪心选择性
  - ③ 算法按照②中的贪心选择性进行局部优化选择能得到全局最优解

- 简化的步骤
  - 最优子结构问题：做一个贪心选择，留下一个待解的子问题
  - 证明存在一个基于贪心选择的最优解，因此贪心选择是安全的
  - 说明由贪心选择和子问题的最优解 ⇒ 原问题的最优解

活动安排问题

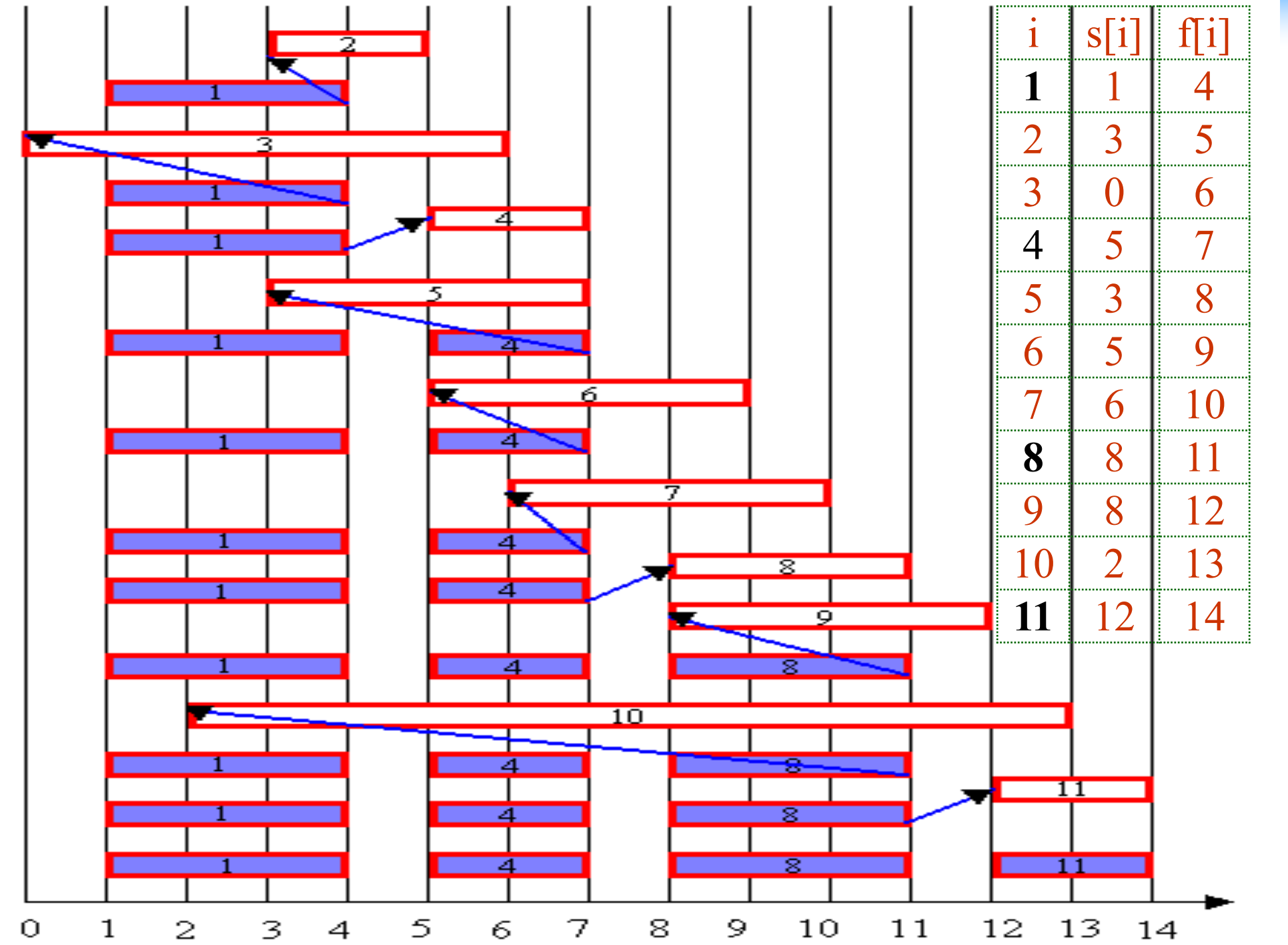
- 定义（活动） 设有n个活动的集合E={1,2,…,n}，其中每个活动都要求使用同一资源，而在同一时间内只有一个活动能使用这一资源
  - 每个活动i都有一个要求使用该资源的起始时间s_i和一个结束时间f_i，且$s_i < f_i$。如果选择了活动i，则它在半开时间区间$[s_i, f_i)$内占用资源
- 定义（相容活动） 若区间$[s_i, f_i)$与区间$[s_j, f_j)$不相交，则称活动i与活动j是相容的。也就是说，当$s_i \geq f_j$或$s_j \geq f_i$时，活动i与活动j相容

活动安排问题

- 问题定义
  - 输入：n个活动的集合S={1，2，…，n}，每个活动占用资源的时间为F={ $[s_i, f_i)$ }，$1 \le i \le n$
  - 输出：S 的最大相容活动集合
- 贪心思想
  - 使尽可能多的活动能兼容地使用公共资源
  - 为了选择最多的相容活动，每次选f_i 最小的活动，使我们能够选更多的活动

活动安排问题

- 算法基本思想
  - 存储各个活动的开始时间和结束时间: 数组按照结束时间非递减排序:
    - $f_1 \le f_2 \le \ldots \le f_n$
    - O(nlogn)排序时间
  - 活动1肯定是占用时间最少的, 直接选用
  - 从活动2开始找与活动1兼容且占用时间最少. 扫描过程中符合条件的第1个活动就是所求.
  - 找与活动2兼容的符合条件的活动直到所有的活动都扫描一遍.



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

活动安排问题

各活动的起始时间和结束时间存储于数组s和f中且按结束时间的非减序排列

```
void GreedySelector(int n, Type s[], Type f[], bool A[])
{
      A[1]=true;
      int j=1;
      for (int i=2;i<=n;i++) {
         if (s[i]>=f[j])  { A[i]=true;    j=i; }
         else             A[i]=false;
         }
}
```

A存储所选择的活动

记录最近一次加入到A中的活动

活动i与集合 A中所有活动相容的充要条件:其开始时间s_i不早于最近加入集合A中的活动j的结束时间

活动安排问题

- 由于输入的活动已经按照完成时间非减序排列，所以算法greedySelector每次总是选择具有最早完成时间的相容活动加入集合A中。
- 贪什么？
  - 按这种方法选择相容活动为未安排活动留下尽可能多的时间。
  - 也即，该算法的贪心选择的意义是使剩余的可安排时间段极大化，以便安排尽可能多的相容活动。

活动安排问题

- 分析
  - 如果输入的活动已按结束时间非减序排列，算法只需O(n)的时间安排n个活动
  - 如果输入的活动未按结束时间非减序排列，算法需要O(nlogn)的时间重排。

```
void GreedySelector(n, s[], f[], A[]) {
        A[1]=true;
        int j=1;
        for (int i=2;i<=n;i++) {
          if (s[i]>=f[j])  { A[i]=true;    j=i; }
          else              A[i]=false;
          }
}
```

活动安排问题

- 适用范围
  - 贪心算法并不总能求得问题的整体最优解
  - 但对于活动安排问题，贪心算法却总能求得的整体最优解，即它最终所确定的相容活动集合A的规模最大
  - 这个结论可以用数学方法严格证明

活动安排问题

- 贪心算法证明的三部分
  - 贪心选择性质
    - 证明活动安排问题有一个最优解以贪心选择开始
  - 最优子结构性质
    - 证明原问题的最优解包含贪心选择后得到的子问题的最优解
  - 每一次贪心选择最终可获得全局最优

活动安排问题 - 贪心性质证明

- 引理1
  - 设S={1,2,…,n}是n 个活动集合，$[s_i,f_i]$是活动的起始终止时间，且$f_1 \leq f_2 \leq \ldots \leq f_n$ ，S的活动安排问题的某个最优解包括活动1.
- 分析
  - 贪心选择 → 贪结束时间最短的活动→活动1
  - 构造这样一个最优解
  - 目的：证明活动安排问题有一个最优解以贪心选择开始

活动安排问题 - 贪心性质证明

- 引理1
  - 设S={1,2,…,n}是n 个活动集合， $[s_i,f_i]$是活动的起始终止时间，且$f_1 \leq f_2 \leq \ldots \leq f_n$ ，S的活动安排问题的某个最优解包括活动1
- 证：设A 是一个最优解，按结束时间排序A中活动，设其第一个活动为k，第二个活动为j
  - 如果k=1，引理成立
  - 如果k≠1，令B=A-{k}∪{1}
  - 由于A 中活动相容且$f_1 \leq f_k \leq s_j$,推出B 中活动相容.
    因为|B|=|A|, 所以B 是一个最优解，且包括活动1

结论：总存在一个以贪心选择开始的最优活动安排方案

活动安排问题 - 最优子结构性质证明

- 引理2
  - 设S={1, 2, …, n}是n 个活动集合，$[s_i, f_i]$是活动i 的起始终止时间，且$f_1 \leq f_2 \leq \ldots \leq f_n$，设A是S的活动安排问题的一个最优解且包括活动1，则A′ =A-{1}是S′ ={i∈S| $s_i \geq f_1$}的活动安排问题的最优解。
- 分析
  - 原问题：S ={1, 2, …, n}　　最优解： A且包含活动1
  - 子问题： S′={$i \in S | s_i \geq f_1$}　　子问题最优解：A-{1}

➢要证明 活动安排问题具有最优子结构性质

活动安排问题 - 最优子结构性质证明

- 引理2
  - 设S={1, 2, …, n}是n 个活动集合，$[s_i, f_i]$是活动i 的起始终止时间，且$f_1 \leq f_2 \leq \ldots \leq f_n$，设A是S的活动安排问题的一个最优解且包括活动1，则A′ =A-{1}是S′ ={i∈S| $s_i \geq f_1$}的活动安排问题的最优解。
  - 证: 显然，A′ 中的活动是相容的，我们仅需要证明A′ 是最大的。设不然，存在一个S′ 的活动选择问题的最优解B′ ，|B′ |>|A′ |
  - 令B={1}∪B′ ，对于i∈S′ ，$s_i \geq f_1$, B 中活动相容， 得到B 是S 的一个解. 由于|A|=|A′ |+1, |B|=|B′ |+1>|A′ |+1=|A|，与A 最优性相矛盾

结论：每一步作的贪心选择将问题简化为一个更小的具有相同形式的子问题

➢说明活动安排问题具有最优子结构性质

活动安排问题

- 引理3
  - 设 S={1, 2, …., n} 是 n 个活动集合， $f_{l_0}=0$, l_i 是$S_i=\{j\in S \mid s_j \geq f_{l_{i-1}}\}$ 中具有最小结束时间 f_{l_i} 的活动. 设A是S 的包含活动1 的最优解, 其中 $f_1 \leq f_2 \leq \ldots \leq f_n$ ，则 $A=\bigcup_{i=1}^{k}\{l_i\}$

f_{l_i}: 表示子问题i的最优解中第1个活动的结束时间(最小结束时间)

- 分析

		待选活动	贪心选择的活动	第1个活动结束时间
原问题 (子问题1)	S={1,2,…, n} $=\{j\in S \mid s_j \geq f_{l_0}\}$	$l_1^1, l_1^2, \ldots, l_1^j$	$l_1^1 \rightarrow l_1$	f_{l_1}
子问题2	$S=\{j\in S \mid s_j \geq f_{l_1}\}$	$l_2^1, l_2^2, \ldots, l_2^j$	$l_2^1 \rightarrow l_2$	f_{l_2}
子问题3	$S=\{j\in S \mid s_j \geq f_{l_2}\}$	$l_3^1, l_3^2, \ldots, l_3^j$	$l_3^1 \rightarrow l_3$	f_{l_3}
子问题i	$S=\{j\in S \mid s_j \geq f_{l_{i-1}}\}$	$l_i^1, l_i^2, \ldots, l_i^j$	$l_i^1 \rightarrow l_i$	f_{l_i}

子问题i对应解的第1个活动

活动安排问题

- 引理3

f_{l_i}: **表示子问题i的最优解第1个活动的结束时间(最小结束时间)**

  - 设 S={1, 2, …., n} 是 n 个活动集合， f_{l_0} =0, l_i 是S_i ={ j∈S | $s_j \geq f_{l_{i-1}}$} 中具有最小结束时间 f_{l_i} 的活动. 设A是S 的包含活动1 的最优解, 其中 $f_1 \leq f_2 \leq \ldots \leq f_n$ ，则

$$A = \bigcup_{i=1}^{k} \{l_i\}$$

➢ **证**

对|A|作**归纳法**.

当|A|=1 时, 由引理1，命题成立.

设|A|<k 时，命题成立.

当|A|=k 时，由引理2，A={1}∪A_1, A_1 是 S_2 = { j∈S | $s_j \geq f_1$ } 的最优解.

由归纳假设，$A_1 = \cup_{i=2}^{k}\{l_i\}$　　于是，$A = \cup_{i=1}^{k}\{l_i\}$

结论：贪心算法最终产生原问题的一个最优解

活动安排问题

- 定理：Greedy-Activity-Selector 算法能够产生最优解
- 证明：
  - Greedy-Activity-Selector 算法按照引理3 的Greedy 选择性进行局部优化选择能够产生最优解.

活动安排问题

- 算法正确性证明 需要证明以下三点：
  - 活动选择问题具有贪心选择性
  - 活动选择问题具有最优子结构
  - 算法按照贪心选择性能计算出解

活动安排问题 - 贪心性质证明

- 证明贪心性质的通用方法
  - 首先考虑问题第一个整体最优解
    - 证明可修改这个最优解,使其从贪心选择开始
  - 利用最优子结构性质证明，做贪心选择后,原问题简化为一个规模更小的子问题
  - 用数学归纳法证明,通过每一步作贪心选择,最终可以得到问题的一个整体最优解

分治、动态规划和贪心的比较

	标准分治	动态规划	贪心
适用类型	通用问题	优化问题	优化问题
子问题结构	每个子问题不同	很多子问题重复	只有一个子问题
最优子结构	不要求	必须满足	必须满足
解决子问题数	全部子问题都要解决	全部子问题都要解决	只要解决一个子问题
子问题依赖性	先选择后解决子问题	先解决子问题后选择	先选择后解决子问题
求解顺序	递归与递推都可	从底往上	从顶往下
基本要素		最优子结构和重叠子问题	最优子结构和贪心选择性

小结

- 理解贪心算法的概念
- 掌握贪心算法的基本要素
  - 最优子结构性质
  - 贪心选择性质
- 理解贪心算法与动态规划算法的差异
- 活动安排问题

- 难点
  - 算法证明

创新实践能力

- 找零问题
  - 给定硬币的面值情况cointype=[1,5,10,25,100]，如果需要给用户给的amount钱款找零，问：需要如何组合这几种面值的货币，使得所找零的硬币数最少？
- 分析
  - 把原问题分解为若干个子问题，每个子问题对应着选择一个硬币。这个子问题是：每次选择后，完成剩余数额的找零。
  - 利用贪心策略求解子问题：从所有满足条件的零钱中，选择面额最大的硬币作为找零。求解所有子问题，知道找零的累加值等于指定数目
  - 把所有子问题的解合并即得到原问题的解

子问题？： 找零后剩下的数额

思考题

- 背包问题，n=3， C=20， v=(25,24,15), w=(18,15,10), 请写出不同贪心策略下的总价值。
- 提示： 贪价值，贪重量，贪单位重量价值。