贪心法(Greedy Approach)

- 基本思想
- 算法设计
  - 设计要素
  - 与动态规划法的比较
  - 正确性证明
- 得不到最优解的处理办法
- 应用实例

活动选择问题

输入：$S=\{1, 2, \ldots, n\}$为n 项活动的集合

s_i, f_i 分别为活动 i 的开始和结束时间

活动 i 与 j 相容 当且仅当 $s_i \geq f_j$ 或 $s_j \geq f_i$

求最大的两两相容的活动集

策略1：排序使得 $s_1 \leq s_2 \leq \ldots \leq s_n$，从前向后挑选

策略2：排序使得 $f_1 - s_1 \leq f_2 - s_2 \leq \ldots \leq f_n - s_n$，从前向后挑选

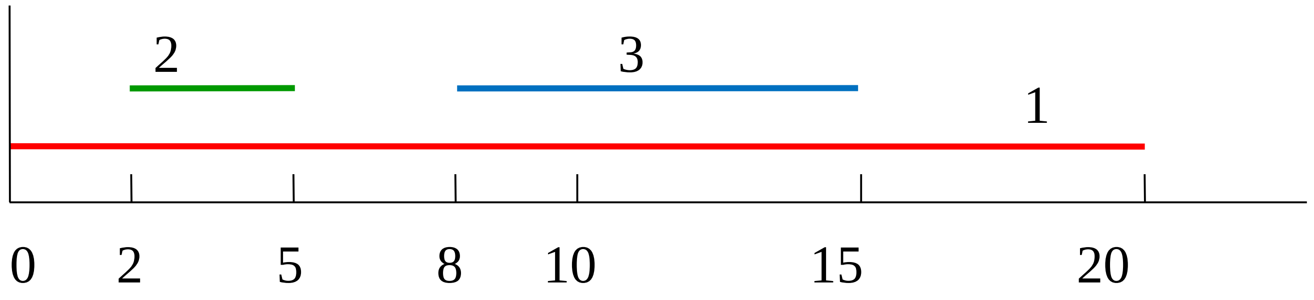
策略3：排序使得 $f_1 \leq f_2 \leq \ldots \leq f_n$，从前向后挑选

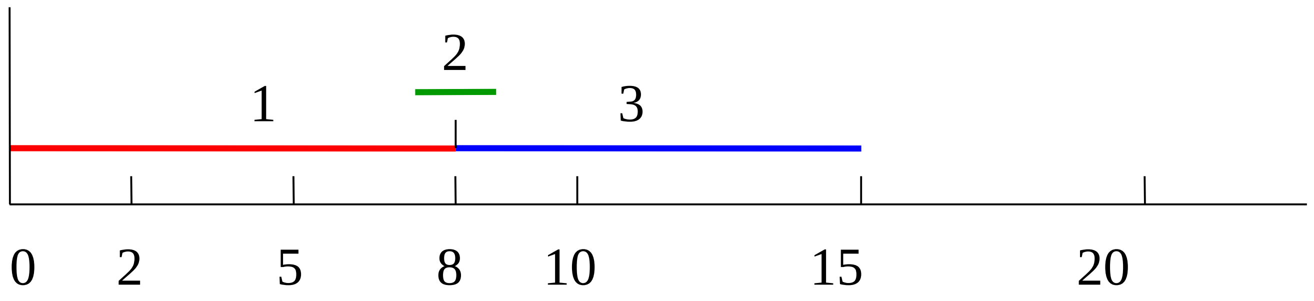
以上策略中的挑选都要注意满足相容性条件

两个反例

策略1：S={1,2,3}，s_1=0，f_1=20，s_2=2，f_2=5，s_3=8，f_3=15

策略2：S={1,2,3}，s_1=0，f_1=8，s_2=7，f_2=9，s_3=8，f_3=15





贪心算法：按截止时间排序

算法 Greedy Select

1. $n \leftarrow length[S]$;
2. $A \leftarrow \{1\}$;
3. $j \leftarrow 1$;
4. for $i \leftarrow 2$ to n
5. do if $s_i \geq f_j$
6. then $A \leftarrow A \cup \{i\}$;
7. $j \leftarrow i$;
8. return A.

最后完成时间 $t = \max\{ f_k : k \in A \}$.

实例

输入

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	8	9	10	11	12	13	14

解为 $A = \{1, 4, 8, 11\}$ $t = 14$

正确性证明

证明方法：

(1) 归纳法：证明贪心法得到最优解

叙述一个描述正确性的命题

对算法步数归纳或者对问题规模归纳

(2) 交换论证：在保证最优性不变的前提下，从一个最优解进行逐步替换，最终得到贪心法的解

定理1 算法Select 执行到第 k 步, 选择 k 项活动 $i_1 = 1$, i_2, …, i_k, 那么存在最优解 A 包含 $i_1=1$, i_2, … , i_k

根据定理：算法至多到第 n 步得到最优解

归纳证明

归纳基础：证明存在最优解包含活动 1
归纳步骤：假设按照算法前 k 步选择都导致最优解，证明第k+1步选择也导致最优解

归纳步骤的证明思路

1. 算法第 k 步选择活动 $i_1=1, i_2, \ldots, i_k$，根据归纳假设，存在最优解

$$A=\{ i_1=1, i_2, \ldots, i_k \} \cup B$$

B是剩下的待选活动集 S'的一个最优解

2. 由归纳基础，存在S'的最优解 B' 包含i_{k+1}
3. 由 $|B'|=|B|$ 知 $A'=\{i_1=1, i_2, \ldots, i_k\} \cup B'$ 最优
4. $A'=\{i_1=1, i_2, \ldots, i_k, i_{k+1}\} \cup (B' - \{i_{k+1}\})$最优.

证明：归纳基础

设$S=\{1,2,\ldots,n\}$是活动集，活动按截止时间递增顺序排序.

$k=1$, 证明存在最优解包含活动1.

任取最优解A, A中的活动按照截止时间递增的

顺序排列. 如果A的第一个活动为j，$j\neq 1$, 令

$$A' = (A-\{j\})\cup\{1\},$$

由于 $f_1 \le f_j$, A'也是最优解，且含有1.

证明：归纳步骤

假设命题对 k 为真,证明对 k+1 也为真.

算法执行到第 k 步, 选择了活动 i_1=1, i_2, …, i_k, 根据归纳假设存在最优解 A 包含 i_1= 1, i_2, … , i_k,
A中剩下的活动选自集合$S'=\{ i \mid i\in S, s_i \ge f_k\}$,且

$$A = \{ i_1, i_2, \ldots , i_k \}\cup B$$

B是S'的最优解.（若不然, S'的最优解为B^*, B^*的活动比 B 多，那么$B^*\cup\{1, i_2, \ldots , i_k\}$是 S 的最优解，且比 A的活动多, 与 A 的最优性矛盾.）
根据归纳基础，存在 S'的最优解B'含有S'中的第一个活动，设为i_{k+1}, 且$|B'|=|B|$, 于是

$$\{i_1,i_2,\ldots,i_k\}\cup B' = \{i_1,i_2,\ldots i_k,i_{k+1}\}\cup(B'-\{i_{k+1}\})$$

也是原问题的最优解.

贪心算法设计要素

- 适用：

  问题求解表示成多步判断

  整个判断序列对应问题的解，子序列对应子问题的解
- 判断的依据——贪心选择：短视的优化策略
- 正确性证明：归纳法(算法步数、问题规模)，交换论证
- 自顶向下计算：通过贪心选择，将原问题规约为子问题
- 线性表记录选择的结果
- 时间复杂度：取决于判断步数与每步的工作量

最优装载 Loading

n 个集装箱1,2,…,n 装上轮船，集装箱 i 的重量 w_i, 轮船装载重量限制为c， 无体积限制. 问如何装使得上船的集装箱最多？不妨设 $w_i \le c$.

$$\max \sum_{i=1}^{n} x_i$$

$$\sum_{i=1}^{n} w_i x_i \le c$$

$$x_i = 0,1 \quad i = 1,2,\ldots,n$$

贪心法：将集装箱按照从轻到重排序，轻者先装

贪心选择性质证明

命题： 对任何规模为n (n是正整数) 的输入，上述贪心法都得到最优解.

证明思路　对规模的归纳

- 设集装箱标号按照从轻到重记为1, 2, … , n
- n=1,贪心选择得到最优解（只有1个箱子，不证）
- 假设对于规模为 n–1 的输入得到最优解，证明对规模为 n 的输入也得到最优解

归纳步骤

假设对于 $n-1$ 个集装箱的输入，贪心法都可以得到最优解，考虑 n 个集装箱的输入 $N = \{1, 2, \ldots, n\}$, 其中

$$w_1 \le w_2 \le \ldots \le w_n.$$

由归纳假设，对于$N' = \{2,3,\ldots,n\}$，$c' = c-w_1$, 贪心法 得到最优解 I'. 令 $I = \{1\} \cup I'$，则 I 是关于 N 的最优解.

若不然，存在包含 1 的关于 N 的最优解 I^*（如果 I^* 中没有 1，用 1 替换 I^* 中的第一个元素得到的解也是最优解），且 且$|I^*| > |I|$； 那么 $I^* - \{1\}$是 N' 的解且

$$|I^* - \{1\}| > |I - \{1\}| = |I'|$$

与 I' 的最优性矛盾.

说明

- Loading 算法

  复杂性　$T(n)=O(n\log n)$

- Loading 问题 是 **0-1背包问题** 的特例：

  即$v_i=1$, $i=1,2,\ldots,n$.

  该问题是 $O(n\log n)$ 时间可解的

  0-1背包问题是NP难的

最小延迟调度

问题：

任务集合S，$\forall i \in S$，d_i为截止时间，t_i为加工时间，

d_i, t_i为正整数.

一个调度 $f: S \to N$，$f(i)$为任务 i 的开始时间. 求最大延迟达到最小的调度，即求 f 使得

$$\min_{f}\{\max_{i \in S}\{f(i)+t_i-d_i\}\}$$

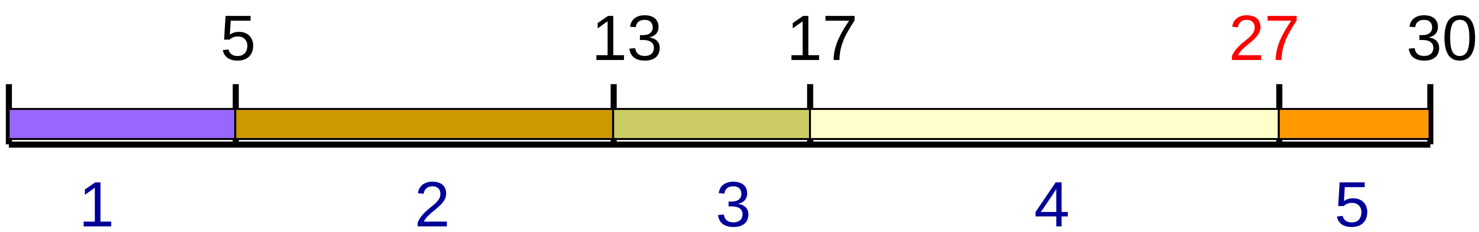
$$\forall i,j \in S, i \neq j, f(i)+t_i \leq f(j) \text{ or } f(j)+t_j \leq f(i)$$

实例

$S=\{1, 2, 3, 4, 5\}$, $d=<10, 12, 15, 11, 20>$, $t=<5, 8, 4, 10, 3>$

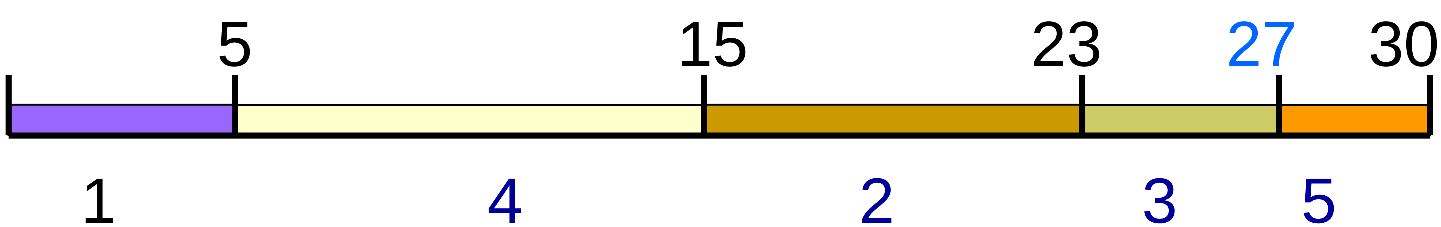
调度1： $f_1(1)=0, f_1(2)=5, f_1(3)=13, f_1(4)=17, f_1(5)=27$

各任务延迟：0, 1, 2, 16, 10； 最大延迟：16



调度2： $f_2(1)=0$, $f_2(2)=15$, $f_2(3)=23$, $f_2(4)=5$, $f_2(5)=27$

各任务延迟：0, 11, 12, 4, 10；最大延迟：12



贪心策略选择

贪心策略1：按照 t_i 从小到大安排任务

贪心策略2：按照 $d_i - t_i$ 从小到大安排任务

贪心策略3：按照 d_i 从小到大安排任务

策略1 对某些实例得不到最优解.

反例：$t_1=1, d_1=100, t_2=10, d_2=10$

策略2 对某些实例得不到最优解.

反例：$t_1=1, d_1=2, t_2=10, d_2=10$

算法设计

算法思想：

按照截止时间 d_i 从小到大选择任务

安排时不留空闲时间

算法

1. Sort(S) 使得 $d_1 \le d_2 \le \ldots \le d_n$
2. $f(1) \leftarrow 0$，$i \leftarrow 2$
3. while $i \le n$ do
4. $\quad f(i) \leftarrow f(i-1) + t_{i-1}$
5. $\quad i \leftarrow i+1$

交换论证：正确性证明

算法的解的性质：

没有空闲时间，没有逆序.

逆序 (i, j): $f(i) < f(j)$ and $d_i > d_j$

命题1 所有没有逆序、没有空闲时间的调度具有相同的最大延迟.

证：因为 f_1 与 f_2 都没有逆序，具有相同截止时间的任务必须被连续安排. 在这些连续安排的任务中最大延迟是最后一个任务，被延迟的时间只与已安排任务加工时间之和有关，与任务标号无关.

交换论证

证明思想：从一个没有空闲时间的最优解出发，在不改变最优性的条件下，转变成没有逆序的解.

(1) 如果一个最优调度存在逆序，那么存在 $i < n$ 使得 $(i, i+1)$ 构成一个逆序.

(2) 存在逆序 (i, j)，$j=i+1$，那么交换 i 和 j 得到的解的逆序数减1，后面证明这个新的调度仍旧最优.

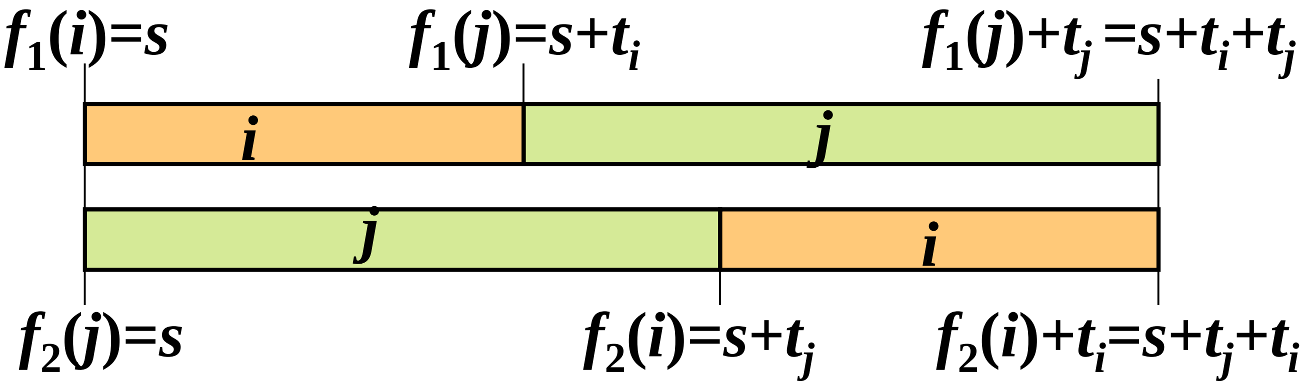
(3) 至多经过 $n(n-1)/2$ 次交换得到一个没有逆序的最优调度.

交换相邻逆序任务(i, j)不影响最优性

(1) 交换 i, j 对其他任务的延迟时间没影响

(2) 交换后不增加 j 的延迟

(3) 任务 i 在 f_2 的延迟 L_{2i} 小于任务 j 在 f_1的延迟 L_{1j}，因此小于f_1的最大延迟



i 在 f_2 结束时间 $= j$ 在 f_1 的结束时间 $= s+t_i+t_j$

i 在 f_2 的延迟：$L_{2i} = (s+t_i+t_j)-d_i$

j 在 f_1 的延迟：$L_{1j} = (s+t_i+t_j)-d_j$

$$\boxed{d_j < d_i \ \Rightarrow L_{2i} < L_{1j}}$$

得不到最优解的处理方法

- 讨论对于哪些输入贪心选择能够得到最优解
  - 输入应该满足的条件
- 讨论贪心法的解最坏情况下与最优解的误差
  - 绝对误差与相对误差估计

找零钱问题

问题描述：

设有 n 种零钱，

重量分别为：$w_1, w_2, \ldots, w_n$,

价值分别为：$v_1=1, v_2, \ldots, v_n$.

付的总钱数是：y

问：如何付钱使得所付钱的总重最轻?

$$\min\{w_1x_1 + w_2x_2 + \ldots + w_nx_n\}$$

$$v_1x_1 + v_2x_2 + \ldots + v_nx_n = y$$

$$x_i \in \mathrm{N}, \quad i = 1,2,\ldots,n$$

动态规划算法

属于整数规划问题，动态规划算法可以得到最优解

设 $F_k(y)$ 表示用前 k 种零钱，总钱数为 y 的最小重量

递推方程

$$F_{k+1}(y) = \min_{0 \le x_{k+1} \le \left\lfloor \frac{y}{v_{k+1}} \right\rfloor} \{F_k(y - v_{k+1}x_{k+1}) + w_{k+1}x_{k+1}\}$$

$$F_1(y) = w_1 \left\lfloor \frac{y}{v_1} \right\rfloor = w_1 y$$

Greedy算法

假设

$$\frac{w_1}{v_1} \geq \frac{w_2}{v_2} \geq \ldots \geq \frac{w_n}{v_n}$$

使用前 k 种零钱，总钱数为 y

贪心法的总重为 $G_k(y)$，则有如下递推方程

$$G_{k+1}(y) = w_{k+1}\left\lfloor \frac{y}{v_{k+1}} \right\rfloor + G_k(y \bmod v_{k+1})$$

$$G_1(y) = w_1\left\lfloor \frac{y}{v_1} \right\rfloor = w_1 y$$

n=1, 2 时得到最优解

$n=1$ 只有一种零钱，$F_1(y)=G_1(y)$， $F_2(y)=G_2(y)$

$n=2$, 使用价值大的钱越多，得到的解越好

$$
\begin{aligned}
&[F_1(y-v_2(x_2+\delta))+w_2(x_2+\delta)] \\
&-[F_1(y-v_2x_2)+w_2x_2] \\
=&[w_1(y-v_2x_2-v_2\delta)+w_2x_2+w_2\delta] \\
&-[w_1(y-v_2x_2)+w_2x_2] \\
=&-w_1v_2\delta+w_2\delta=\delta(-w_1v_2+w_2)\le 0
\end{aligned}
$$

$n>2$时得到最优解的判定条件

定理2 假定 $G_k(y)=F_k(y)$,

$v_{k+1}>v_k$ 且 $v_{k+1}=pv_k-\delta$, $0\le\delta<v_k$, $p\in Z^+$,

则以下命题等价.

$$(1)\ G_{k+1}(y)\le G_k(y)$$

$$(2)\ G_{k+1}(y)=F_{k+1}(y)$$

$$(3)\ G_{k+1}(pv_k)=F_{k+1}(pv_k)$$

$$(4)\ w_{k+1}+G_k(\delta)\le pw_k$$

用条件(4)需 $O(k)$ 时间验证 $G_{k+1}(y)=F_{k+1}(y)$?

对 n 种零钱作出验证, 可在 $O(n^2)$ 时间内完成

实例

$$v_{k+1} = pv_k - \delta, \quad 0 \le \delta < v_k, \quad p \in \mathbf{Z}^+$$
$$w_{k+1} + G_k(\delta) \le pw_k$$

例　v_1=1, v_2=5, v_3=14, v_4=18, w_i=1, i=1, 2, 3, 4.

对一切 y 有 $G_1(y)=F_1(y)$, $G_2(y)=F_2(y)$.

验证　$G_3(y) = F_3(y)$

$v_3=pv_2-\delta \Rightarrow 14=5p-\delta \Rightarrow p=3,\ \delta=1.$

$w_3+G_2(\delta)=1+G_2(1)=1+1=2$

$pw_2=3\times 1=3$

$w_3+G_2(\delta)\le p\,w_2$

实例

$$\boxed{\begin{array}{l} v_{k+1} = pv_k - \delta, \quad 0 \le \delta < v_k, \quad p \in \mathbf{Z}^+ \\ w_{k+1} + G_k(\delta) \le pw_k \end{array}}$$

$v_1=1$, $v_2=5$, $v_3=14$, $v_4=18$, $w_i=1$, $i=1, 2, 3, 4$.

$v_4= pv_3-\delta \Rightarrow 18=14p-\delta \Rightarrow p=2, \delta=10$

$w_4+G_3(\delta)=1+G_3(10)=1+2=3$

$pw_3=2\times 1=2$

$w_4+G_3(\delta) > pw_3$, $G_4(y)$不是最优解.

$G_4(pv_3) > F_4(pv_3)$. 即

$G_4(28)=\lfloor 28/18\rfloor+\lfloor 10/5\rfloor = 1+2 = 3$

$F_4(28)=28/14 = 2.$