两个例子：调度问题与投资问题

例1：调度问题

问题　有 n 项任务，每项任务加工时间已知. 从 0时刻开始陆续安排到一台机器上加工. 每个任务的完成时间是从 0 时刻到任务加工截止的时间.
求：总完成时间（所有任务完成时间之和）最短的安排方案.

实例

任务集 $S = \{1, 2, 3, 4, 5\}$,

加工时间：$t_1=3$, $t_2=8$, $t_3=5$, $t_4=10$, $t_5=15$

贪心法的解

算法： 按加工时间 (3,8,5,10,15) 从小到大安排

解： 1, 3, 2, 4, 5

总完成时间：

$$t = 3+(3+5)+(3+5+8)+(3+5+8+10)+(3+5+8+10+15)$$
$$= 3\times5 + 5\times4 + 8\times3 + 10\times2 + 15$$
$$= 94$$

问题建模

输入：任务集： $S=\{1, 2, \ldots, n\}$，

第 j 项任务加工时间： $t_j \in Z^+$， $j=1,2,\ldots,n$.

输出： 调度 I， S 的排列 $i_1, i_2, \ldots, i_n$，

目标函数： I 的完成时间， $\boxed{t(I)=\sum_{k=1}^{n}(n-k+1)t_{i_k}}$

解 I^*： 使得 $t(I^*)$ 达到最小， 即

$$t(I^*)=\min\{\, t(I) \mid I \text{ 为} S \text{的排列} \}$$

贪心算法

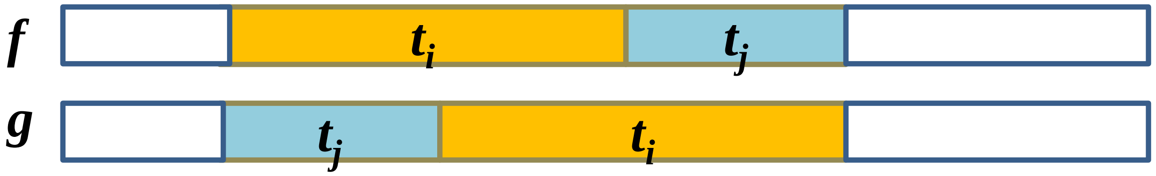
设计策略：加工时间短的先做

算法：根据加工时间从小到大排序,依次加工

算法正确性：对所有输入实例都得到最优解

证：假如调度 f 第 i, j 项任务相邻且有逆序，

即 $t_i > t_j$. 交换任务 i 和 j 得到调度 g，



总完成时间 $t(g) - t(f) = t_j - t_i < 0$

直觉不一定是正确的

反例

有4 件物品要装入背包, 物品重量和价值如下:

标号	1	2	3	4
重量 w_i	3	4	5	2
价值 v_i	7	9	9	2

背包重量限制是 6， 问如何选择物品， 使得不超重的情况下装入背包的物品价值达到最大?

实例的解

贪心法：单位重量价值大的优先，总重不超 6

按照 $\dfrac{v_i}{w_i}$ 从大到小排序：1, 2, 3, 4

$$\frac{7}{3} > \boxed{\frac{9}{4}} > \frac{9}{5} > \boxed{\frac{2}{2}}$$

贪心法的解：{ 1, 4 }，重量 5，价值为 9.

更好的解：{ 2, 4 }，重量 6，价值 11.

算法设计

1. 问题建模
2. 选择什么算法？如何描述这个方法？
3. 这个方法是否对所有实例都得到最优解？如何证明？
4. 如果不是，能否找到反例？

例2：投资问题

问题：m元钱，投资 n 个项目. 效益函数 $f_i(x)$，表示第 i 个项目投 x 元的效益， i =1, 2, …, n. 求如何分配每个项目的钱数使得总效益最大？

实例： 5 万元，投资给 4 个项目，效益函数：

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
0	0	0	0	0
1	11	0	2	20
2	12	5	10	21
3	13	10	30	22
4	14	15	32	23
5	15	20	40	24

建模

输入： $n, m, f_i(x),\ i = 1,2,\ldots,n,\ x = 1,2,\ldots,m$

解： n 维向量 $<x_1, x_2, \ldots, x_n>$, x_i 是第 i 个项目的钱数，使得下述条件满足：

目标函数 $$\max \sum_{i=1}^{n} f_i(x_i),$$

约束条件 $$\sum_{i=1}^{n} x_i = m, \quad x_i \in \mathrm{N}$$

蛮力算法

对所有满足下述条件的向量 $<x_1,x_2,\ldots,x_n>$

$$x_1 + x_2 + \ldots + x_n = m$$

x_i 为非负整数, $i = 1, 2, \ldots, n$

计算相应的效益

$$f_1(x_1) + f_2(x_2) + \ldots + f_n(x_n)$$

从中确认效益最大的向量.

实例计算

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
0	0	0	0	0
1	11	0	2	20
2	12	5	10	21
3	13	10	30	22
4	14	15	32	23
5	15	20	40	24

$x_1 + x_2 + x_3 + x_4 = 5$

s_1=<0,0,0,5>, $v(s_1)$=24

s_2=<0,0,1,4>, $v(s_2)$=25

s_3=<0,0,2,3>, $v(s_3)$=32

…

s_{56} =<5,0,0,0>,$v(s_{56})$=15

解：s=<1,0,3,1>,

最大效益：11+30+20 = 61

蛮力算法的效率

方程 $x_1 + x_2 + \ldots + x_n = m$ 的非负整数解

$< x_1, x_2, \ldots, x_n >$ 的个数估计：

可行解表示成 0-1 序列：m 个1,n-1个 0

$$\underbrace{1\ .\ .\ .\ 1}_{x_1\text{个}}\ 0\ \underbrace{1\ .\ .\ .\ 1}_{x_2\text{个}}\ 0\ .\ .\ .\ 0\ \underbrace{1\ .\ .\ .\ 1}_{x_n\text{个}}$$

n=4, m=7

可行解 <1, 2, 3, 1>

⇔ 序列 1 0 1 1 0 1 1 1 0 1

蛮力算法的效率

序列个数是输入规模的指数函数

$$C(m+n-1,m)$$

$$=\frac{(m+n-1)!}{m!(n-1)!}$$

$$=\Omega((1+\varepsilon)^{m+n-1})$$

有没有更好的算法?

小结

问题求解的关键

- 建模：对输入参数和解给出形式化或半形式化的描述
- 设计算法：
  采用什么算法设计技术
  正确性——是否对所有的实例都得到正确的解
- 分析算法——效率