《算法设计与分析》

第四章 贪心方法

马丙鹏

2024年10月21日

中国科学院大学
University of Chinese Academy of Sciences

第四章 贪心方法

- 4.1 一般方法
- 4.2 背包问题
- 4.3 带有限期的作业排序
- 4.4 最优归并模式
- 4.5 最小生成树
- 4.6 单源点最短路径

4.3 带有限期的作业排序

- ■ 1. 问题描述
  - □假定在一台机器上处理n个作业，
    - ➢每个作业均可在单位时间内完成；
    - ➢每个作业i都有一个截至期限d_i>0,
    - ➢每个作业i在其截至期限以前被完成时，则获得p_i>0的效益。
  - □问题：
    - ➢求这n个作业的一个子集J，其中的所有作业都可在其截至期限内完成。
    - ➢——J是问题的一个可行解。
  - □可行解J中的所有作业的效益之和是 $\sum_{i\in J} p_i$ ，具有最大效益值的可行解是该问题的最优解。

4.3 带有限期的作业排序

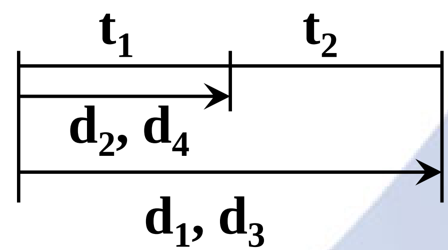
- **1. 问题描述**
  - 目标函数：
    $$\sum_{i \in J} p_i$$
  - 约束条件：
    - 所有的作业都应在其期限之前完成。
  - 分析
    - 如果所有的作业期限“足够宽松”，而使得多有作业都能在其期限之内完成，则显然可以获得当前最大效益值；
    - 否则，将有作业无法完成；
    - ——决策应该执行哪些作业，以获得最大可能的效益值。

4.3 带有限期的作业排序

■ 1. 问题描述

□ [例4.2] n=4，$(p_1, p_2, p_3, p_4)=(100, 10, 15, 20)$和$(d_1, d_2, d_3, d_4)=(2, 1, 2, 1)$。可行解如下表所示：

	可行解J	处理顺序	效益值$\sum p_j$
①	(1)	1	100
②	(2)	2	10
③	(3)	3	15
④	(4)	4	20
⑤	(1, 2)	2, 1	110
⑥	(1, 3)	1, 3 或 3, 1	115
⑦	(1, 4)	4, 1	120
⑧	(2, 3)	2, 3	25
⑨	(3, 4)	4, 3	35



问题的最优解是⑦。所允许的处理次序是：先处理作业4再处理作业1。

4.3 带有限期的作业排序

- **2. 贪心策略求解**
  - 度量标准的选择
    - 以目标函数 $\sum_{i \in J} p_i$ 作为度量。
    - 度量标准：
      - ✓下一个要计入的作业将是，使得在满足所产生的J是一个可行解的限制条件下让 $\sum_{i \in J} p_i$ 得到最大增加的作业。
    - 处理规则：
      - ✓按 p_i 的非增次序来考虑这些作业。

4.3 带有限期的作业排序

■ 2. 贪心策略求解

例4.2求解： $(p_1, p_2, p_3, p_4)=(100, 10, 15, 20)$

$(d_1, d_2, d_3, d_4)=(2, 1, 2, 1)$

① 首先令J=Φ，$p_1 > p_4 > p_3 > p_2$，$\sum_{i\in J} p_i = 0$

② 作业1具有当前的最大效益值，且{1}是可行解，所以作业1计入J；

③ 在剩下的作业中，作业4具有最大效益值，且{1, 4}也是可行解，故作业4计入J，即J={1, 4}；

④ 考虑{1, 3, 4}和{1, 2, 4}均不能构成新的可行解，作业3和2将被舍弃。

故最后的J={1, 4}，最终效益值=120(问题的最优解)。

4.3 带有限期的作业排序

■ 2. 贪心策略求解

□作业排序算法的概略描述

算法4.3 作业排序算法的概略描述

```
procedure GREEDY-JOB(D, J, n)
  //作业按p1≥p2≥…≥pn的次序输入，它们的期限值D(i)≥1,
   1≤i≤n, n≥1。J是在它们的截止期限完成的作业的集合//
  J←{1}
  for i←2 to n do
    if J∪{i}的所有作业能在它们的截止期限前完成
      then J←J∪{i}
    endif
  repeat
end GREEDY-JOB
```

4.3 带有限期的作业排序

■ 3. 最优解证明

定理4.2 算法4.3对于作业排序问题总是得到问题的一个最优解。

证明：

设J是由算法所得的贪心解作业集合，I是一个最优解的作业集合。

① 若I=J，则J就是最优解；

② 否则$I \not\subset J$且$J \not\subset I$，即至少存在两个作业a和b，使得$a \in J$且$a \notin I$, $b \in I$且$b \notin J$。

4.3 带有限期的作业排序

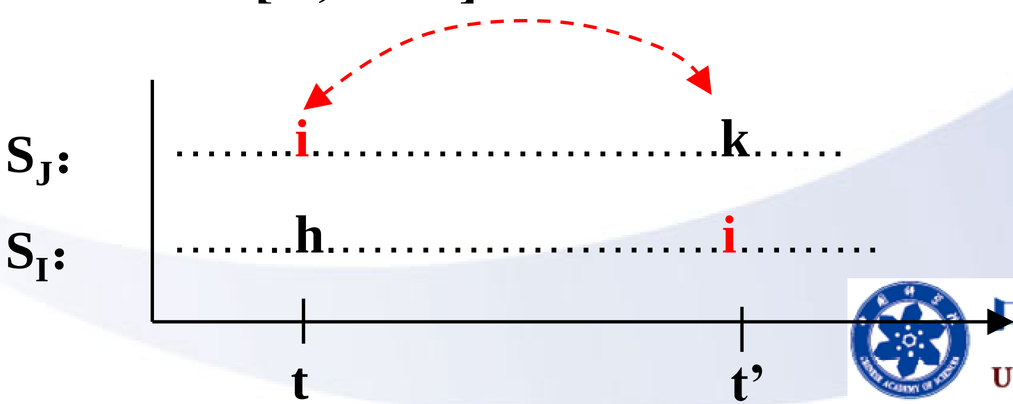
设S_J和S_I分别是J和I的可行的调度表。因为J和I都是可行解，故这样的调度表一定存在；

先对相同的作业进行调整

设i是既属于J又属于I的一个作业，并i设在调度表S_J中的调度时刻是[t, t+1]，而在S_I中的调度时刻是[t’, t’+1]。

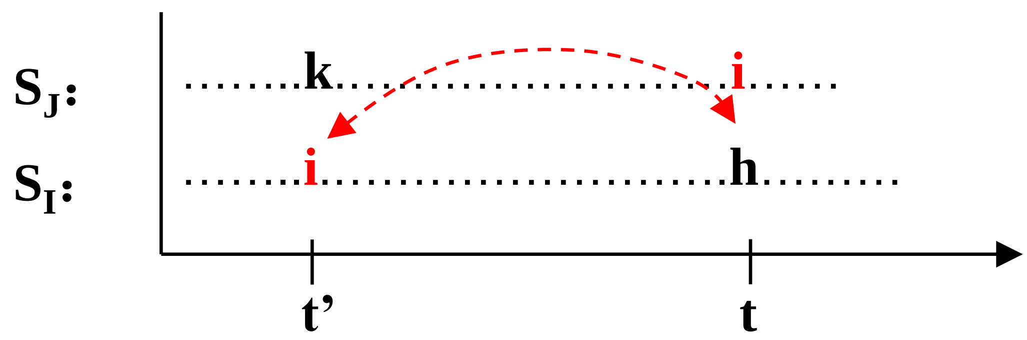
在S_J和S_I中作如下调整：

- 若$t<t'$，则将S_J中在[t’, t’+1]时刻调度的那个作业(如果有的话)与i相交换。如果J中在[t’, t’+1]时刻没有作业调度，则直接将i移到[t’, t’+1]调度。——新的调度表也是可行的。反之，



4.3 带有限期的作业排序

- 若t’<t，将S_I中在[t, t+1]时刻调度的那个作业(如果有的话)与i相交换。如果I中在[t, t+1]时刻没有作业调度，则直接将i移到[t, t+1]调度。——同样，新的调度表也是可行的。



- 对J和I中共有的所有作业作上述的调整。设调整后得到的调度表为S'_J和S'_I，则在S'_J和S'_I中J和I中共有的所有作业将在相同的时间被调度。

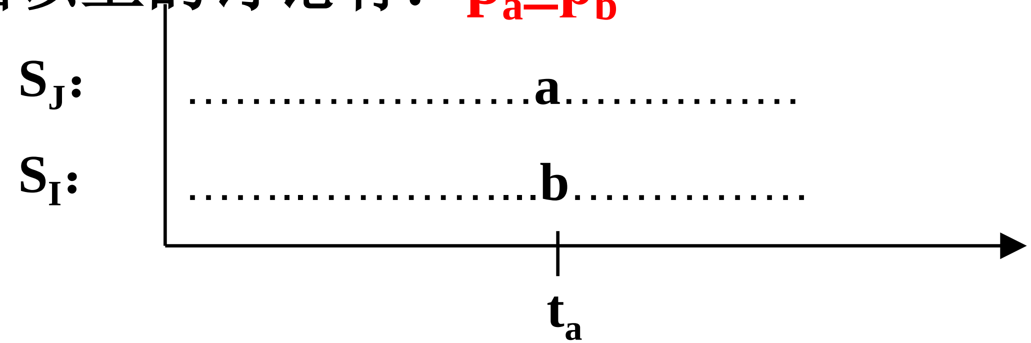
中国科学院大学
University of Chinese Academy of Sciences

4.3 带有限期的作业排序

对不同的作业进行调整

设a是使得a∈J且a∉I的一个具有最高效益值的作业，由算法处理规则可得：对于在I中而不在J中的作业所有b，有: $p_a \ge p_b$。

设a在S'_J中的调度时刻是$[t_a, t_a+1]$， b是S'_I中该时刻调度的作业。根据以上的讨论有：$p_a \ge p_b$



在S'_I中，去掉作业b，而去调度作业a，得到的是作业集合$I'=I-\{b\}\cup\{a\}$的 一个可行的调度表，且I'的效益值不小于I的效益值。而I'中比I少了一个与J不同的作业。

重复上述的转换，可使I在不减效益值的情况下转换成J。从而J至少有和I一样的效益值。所以J也是最优解。证毕。

中国科学院大学
University of Chinese Academy of Sciences

4.3 带有限期的作业排序

■ 4. 如何判断J的可行性

- 怎样才能称“J是可行的”？
  - J中的作业能够按照某种不违反任一作业时间期限的次序执行。
- 方法一：
  - 检验J中作业所有可能的排列，对于任一种次序排列的作业排列，判断这些作业是否能够在其期限前完成。
  - 对于所给出的一个排列$\sigma = i_1 i_2 \ldots i_k$，由于作业$i_j$完成的最早时间是j，因此只要判断出σ排列中的每个作业$d_{i_j} \geq j$，就可得知σ是一个允许的调度序列，从而J是一个可行解。

4.3 带有限期的作业排序

- **4. 如何判断J的可行性**
  - 方法一：
    - 反之，如果σ排列中有一个$d_{ij}<j$，则σ将是一个行不通的调度序列，因为至少作业i_j不能在其期限之前完成。
    - 若J中有k个作业，则将要检查**k!**个序列。
  - 方法二：
    - 检查J中作业的一个特殊序列就可判断J的可行性：按照作业期限的非降次序排列的作业序列。

4.3 带有限期的作业排序

定理4.3 设J是k个作业的集合，$\sigma=i_1i_2\ldots i_k$是J中作业的一种排列，它使得$d_{i1}\le d_{i2}\le\ldots\le d_{ik}$。J是一个可行解，当且仅当J中的作业可以按照σ的次序而又不违反任何一个期限的情况来处理。

证明：

① 如果J中的作业可以按照σ的次序而又不违反任何一个期限的情况来处理，则J是一个可行解。

② 若J是一个可行解，σ是否代表一个可行的调度序列？

J为一个可行解， 必存在序列$\sigma'=r_1r_2\ldots r_k$，使得$d_{rj}\ge j,\ 1\le j\le k$

- 若σ=σ’，则σ即是可行解。否则，
- 若σ≠σ’，令a是使得$r_a\ne i_a$的最小下标，并设$r_b=i_a$。则有：

$b>a$ 且 $d_{ra}\ge d_{rb}$ (?)

中国科学院大学 University of Chinese Academy of Sciences

4.3 带有限期的作业排序

■ 4. 如何判断J的可行性

$\sigma = i_1 \ \ i_2 \ \ldots \ i_a \ \ldots \ \ldots \ i_c \ \ldots \ i_k$ $(d_{i1} \leq d_{i2} \leq \ldots \leq d_{ik})$

$\sigma' = r_1 \ \ r_2 \ \ldots \ r_a \ \ldots \ r_b \ \ldots \ r_k$

- $a<b$
- $d_{ic} \geq d_{ia}$; $d_{ic}=d_{ra}$; $d_{ia}=d_{rb}$
- $d_{ra} \geq d_{rb}$

在σ'中调换r_a与r_b，所得的新序列$\sigma'' = s_1 s_2 \ldots s_k$的处理次序不违反任何一个期限。

重复上述过程，则可将σ'转换成σ且不违反任何一个期限。故σ是一个可行的调度序列。

故定理得证。

4.3 带有限期的作业排序

- 5. 带有限期的作业排序算法的实现
  - 对当前正在考虑的作业j，按限期大小采用一种“插入排序”的方式，尝试将其“插入”到一个按限期从小到大顺序构造的作业调度序列中，以此判断是否能够合并到当前部分解J中。
    - 如果可以，则插入到序列中，形成新的可行解，
    - 否则，舍弃该作业。
  - 具体如下：

4.3 带有限期的作业排序

■ 5. 带有限期的作业排序算法的实现

① 假设n个作业已经按照效益值从大到小的次序，即$p_1 \geq p_2 \geq \ldots \geq p_n$的顺序排列好，每个作业可以在单位时间内完成，并具有相应的时间期限；且至少有一个单位时间可以执行作业。

② 将作业1存入部分解J中，此时J是可行的；

③ 依次考虑作业2到n。假设已经处理了i-1个作业，其中有k个作业计入了部分解J中：J(1), J(2), …, J(k), 且有 $D(J(1)) \leq D(J(2)) \leq \ldots \leq D(J(k))$

④ 对当前正在考虑的作业i，将D(i)依次和D(J(k)), D(J(k-1)), …, D(J(1))相比较，直到找到位置q：使得 $D(J(q)) \leq D(i) < D(J(l)) \quad q < l \leq k$

4.3 带有限期的作业排序

■ 5. 带有限期的作业排序算法的实现

⑤ 若$D(J(l))>l$，$q<l\le k$，即说明q位置之后的所有作业均可推迟一个单位时间执行，而又不违反各自的执行期限。

⑥ 将q位置之后的所有作业后移一位，将作业i插入到位置q+1处，从而得到一个包含k+1个作业的新的可行解。

⑦ 若找不到这样的q，作业i将被舍弃。

⑧ 对i之后的其它作业重复上述过程直到n个作业处理完毕。

⑨ 最后J中所包含的作业集合是此时算法的贪心解，也是问题的最优解。

算法4.4 带有限期和效益的单位时间的作业排序贪心算法

```
procedure JS(D, J, n, k)        //D(1),…,D(n)是期限值。n≥1。作业已按p1≥p2≥…≥pn的顺序排序。J(i)是最优解中的第i个作业，1≤i≤k。终止时， D(J(i))≤D(J(i+1))， 1≤i<k//
  integer D(0:n), J(0:n), i, k, n, r
  D(0)←J(0)←0 //初始化//
  k←1; J(1)←1 //计入作业1//
  for i←2 to n do//按p的非增次序考虑作业。找i的位置并检查
    r←k
    while D(J(r))>D(i) and D(J(r))≠r do    r←r-1   repeat
    if D(J(r))≤D(i) and D(i)>r then //把i插入到J中//
      for m←k to r+1 by -1 do
          J(m+1) ←J(m)
      repeat
      J(r+1) ←i;  k←k+1
    endif
  repeat
end JS
```

从D(J(k))到 D(J(1))依次与D(i)比较来寻找插入位置r的过程

满足条件表示找到插入位置r

实现作业r+1到作业k依次往后移动一个位置

将作业i插入到r+1位置

4.3 带有限期的作业排序

■ 5. 带有限期的作业排序算法的实现

实例: $n=5$, $(p_1, p_2, p_3, p_4, p_5)=(10, 1, 15, 20, 5)$

$(d_1, d_2, d_3, d_4, d_5)=(1, 4, 3, 2, 4)$

排序: $(p_1', p_2', p_3', p_4', p_5')=(20, 15, 10, 5, 1)$

$(d_1', d_2', d_3', d_4', d_5')=(2, 3, 1, 4, 4)$

4.3 带有限期的作业排序

	0	1'	2'	3'	4'	5'
p		20	15	10	5	1
d	0	2	3	1	4	4

J(0)	J(1)	J(2)	J(3)	J(4)	J(5)
0	1'	2'			

D(0)=J(0)=0; k=1; J(1)=1;

for循环(i=2时)

r = k =1;

while (D(J(r)))$\not>$D(i) and D(J(r))) ≠r) 条件不满足,不执行循环

if (D(J(r)))≤D(i) and D(i)>r) then

{ for m=k=1 to r+1=2 by –1 do 不执行循环

J(r+1)=J(2)=i=2 ;   k=k+1=2;

}

	0	1’	2’	3’	4’	5’
p		20	15	10	5	1
d	0	2	3	1	4	4

J(0)	J(1)	J(2)	J(3)	J(4)	J(5)
0	3’	1’	2’		

k=2;

for循环(i=3时)

r = k = 2;

while (D(J(r)))>D(i) and D(J(r))) ≠ r) 执行两次循环后 r=0

if (D(J(r)))≤D(i) and D(i)>r) then

{ for m=k=2 to r+1=1 by –1 do 执行2次 J(3)=J(2); J(2)=J(1);

J(r+1)=J(1)=i=3 ; k=k+1=3;

}

	0	1’	2’	3’	4’	5’
p		20	15	10	5	1
d	0	2	3	1	4	4

J(0)	J(1)	J(2)	J(3)	J(4)	J(5)
0	3’	1’	2’	4’	

```
k=3;
for循环(i=4时)
  r = k = 3;
  while (D(J(r)))≯D(i) and D(J(r)))=r)条件不满足,不执行循环
   if  ( D(J(r)))≤D(i) and D(i)>r ) then
    { for  m=k=3 to r+1=4  by –1 do不执行循环;
       J(r+1)=J(4)=i=4 ;   k=k+1=4;
    }
```

	0	1'	2'	3'	4'	5'
p		20	15	10	5	1
d	0	2	3	1	4	4

J(0)	J(1)	J(2)	J(3)	J(4)	J(5)
0	3'	1'	2'	4'	

k=4;

for循环(i=5时)

r = k = 4;

while (D(J(r))≯D(i) and D(J(r)))== r) 条件不满足,不执行循环

if (D(J(r)))≤D(i) and D(i) ≯ r)条件不满足

作业5不能插入

4.3 带有限期的作业排序

■ 5. 带有限期的作业排序算法的实现

- □例: 设n=7, $(p_1, p_2, \ldots, p_n)$=(35, 30, 25, 20, 15, 10, 5), $(d_1, d_2, \ldots, d_n)$=(4, 2, 4, 3, 4, 8, 3),
- □算法GreedyJob的执行过程:

J(0)	J(1)	J(2)	J(3)	J(4)	J(5)
	d_1				
0	4				
	d_2	d_1			
0	2	4			

J(0)	J(1)	J(2)	J(3)	J(4)	J(5)
	d_2	d_1	d_3		
0	2	4	4		
	d_2	d_4	d_1	d_3	
0	2	3	4	4	
	d_2	d_4	d_1	d_3	d_6
0	2	3	4	4	8

4.3 带有限期的作业排序

■ 5. 带有限期的作业排序算法的实现

□计算时间分析

```
for i←2 to n do  ------------------→ 将循环n-1次①
    r←k
    while D(J(r))>D(i) and D(J(r)) ≠r do  --------→ 至多循环k次，
                                                  k是当前计入J中的作业数 ②
       r←r-1
    repeat
    if D(J(r))≤D(i) and D(i)>r then
       for i←k to r+1 by -1 do  ----→ 循环k-r次，r是插入点的位置 ③
          J(i+1) ←J(i)
       repeat
       J(r+1) ←i; k←k+1
    endif
repeat
```

中国科学院大学
University of Chinese Academy of Sciences

4.3 带有限期的作业排序

- **5. 带有限期的作业排序算法的实现**
  - 计算时间分析
    - 设s是最终计入J中的作业数(即k的最终值)，则算法JS所需要的总时间是O(sn)。$s \le n$，
    - 最坏情况：$T_{JS} = O(n^2)$，特例情况：$p_i=d_i=n-i+1$, $1 \le i \le n$
    - 最好情况：$T_{JS} = O(n)$，特例情况：$d_i=i$, $1 \le i \le n$

4.3 带有限期的作业排序

- **6. 一种“更快”的作业排序问题**
  - 使用不相交集合的**UNION**和**FIND**算法(见ppt 2.1.3节)，可以将JS的计算时间降低到数量级接近**O(n)**。
  - 对作业i分配的时间为**[α -1, α]**，其中α应尽量取大，且时间片**[α -1, α]**为空
  - 基本思想:
    - 尽可能推迟对作业i的处理。
    - 这样在安排作业处理次序时不必每有一个作业加入就需移动J中已有的作业。
    - 如果不存在这样的时间片，作业i被舍弃。

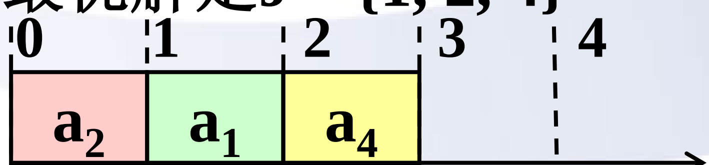
4.3 带有限期的作业排序

■ 6. 一种“更快”的作业排序问题

□ 例4.3 设n=5, $(p_1, \ldots, p_5) = (20, 15, 10, 5, 1)$, $(d_1, \ldots, d_5)=(2, 2, 1, 3, 3)$。

考虑的作业	可行解J	已分配的时间片	分配动作
1	∅	无	分配[1, 2]
2	{1}	[1, 2]	分配[0, 1]
3	{1, 2}	[0, 1] [1, 2]	舍弃
4	{1, 2}	[0, 1] [1, 2]	分配[2, 3]
5	{1, 2, 4}	[0, 1] [1, 2] [2, 3]	舍弃

➢ 最优解是J＝{1, 2, 4}



中国科学院大学
University of Chinese Academy of Sciences

4.3 带有限期的作业排序

- 6. 一种“更快”的作业排序问题
  - 例4.4 设n=7, $(p_1, p_2, \ldots, p_n)=(35, 30, 25, 20, 15, 10, 5)$, $(d_1, d_2, \ldots, d_n)=(4, 2, 4, 3, 4, 8, 3)$,

考虑的作业	可行解J	已分配的时间片	分配动作
1	0	无	分配[3, 4]
2	{1}	[3, 4]	分配[1, 2]
3	{1,2}	[1, 2], [3, 4]	分配[2, 3]
4	{1,2,3}	[1, 4]	分配[0, 1]
5	{1,2,3,4}	[0, 4]	舍弃
6	{1,2,3,4}	[0, 4]	[7, 8]
7	{1,2,3,4,6}	[0, 4], [7, 8]	舍弃

4.3 带有限期的作业排序

- 7. 具有不同执行时间的作业调度问题
  - 问题一：作业有开始时间和结束时间的要求
    - 在一台单CPU、资源无约束的机器上执行n个非抢占式作业，每个作业有不同的执行时间，
    - s_i是作业i的起始时间，f_i是作业i的结束时间,且$s_i<f_i$。
    - 若执行作业i，则在时间区间$[s_i, f_i)$内作业i占用CPU，其它作业必须等待。
    - 若区间$[s_i, f_i)$与区间$[s_j, f_j)$不相交($s_i \geq f_j$或$s_j \geq f_i$)，则称作业i与作业j是相容的。
    - 问题：
      - ✓求可相容的最大作业集合。

中国科学院大学
University of Chinese Academy of Sciences

4.3 带有限期的作业排序

- 7. 具有不同执行时间的作业调度问题
  - 问题一：作业有开始时间和结束时间的要求
    - 求解策略
      - ✓按作业结束时间的非降次序排序。
      - ✓从排序后的作业1开始依次考查 。
      - ✓若作业i能够和前一个被选中的作业相容，则保留，否则舍去。

4.3 带有限期的作业排序

■ 7. 具有不同执行时间的作业调度问题

```
procedure GreedyJobSelect(s, f, b, n)
    integer count, j;
    b[1]←true;  j←1;  count←1;
    for  i←2 to n do
      if s[i] ≥ f[j] then
        b[i]←true;  j←i;
        count ← count+1;
      else b[i]←false;
      endif
    repeat
    return count;
end GreedyJobSelect
```

//n个作业按照结束时间的非降次序排序，s、f分别是排好序的作业的开始时间表和结束时间表。b是一个大小为n的布尔数组，若作业i被选中，b[i]=true，否则b[i]=false//

i: 当前要处理的作业

j: 排序好的列表中的最后一个作业

4.3 带有限期的作业排序

- 7. 具有不同执行时间的作业调度问题
  - 问题二：作业只有执行时间要求
    - 在一台单CPU、资源无约束的机器上执行n个非抢占式作业，作业有不同的执行时间$t_1, t_2, \ldots t_n$, 但没有起止时间的要求。
    - 问题：
      - ✓求这些作业的最短平均完成时间
    - 分析：
      - ✓作业的完成时间 = 作业的等待时间+作业的执行时间
      - ✓作业的等待时间 = 该作业之前其它作业执行时间之和
      - ✓平均完成时间 = Σ所有作业的完成时间/n

中国科学院大学
University of Chinese Academy of Sciences

4.3 带有限期的作业排序

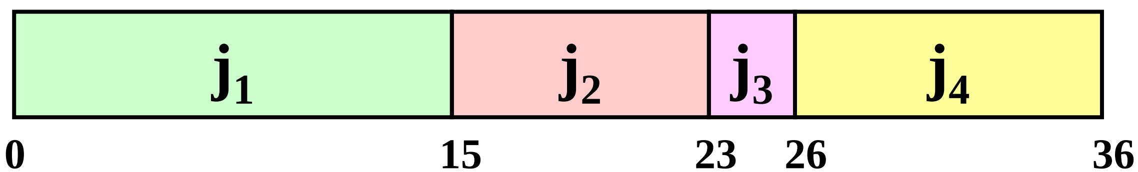
作业	时间
j_1	15
j_2	8
j_3	3
j_4	10

■ 7. 具有不同执行时间的作业调度问题

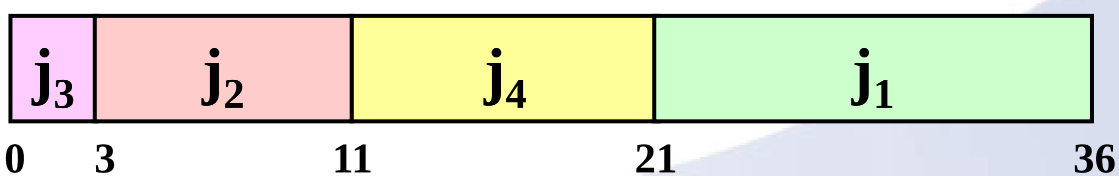
□问题二：作业只有执行时间要求

➢例，四个作业及其执行时间如下：

✓调度方式一：



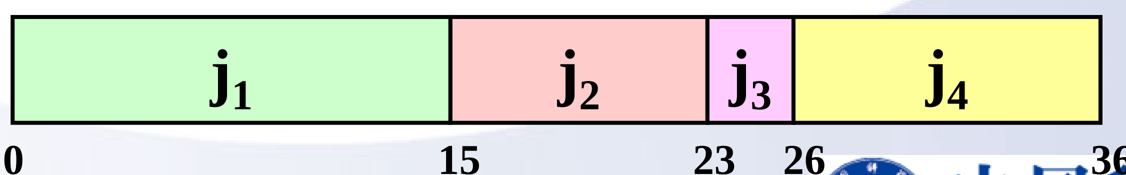
✓调度方式二：



4.3 带有限期的作业排序

作业	时间
j_1	15
j_2	8
j_3	3
j_4	10

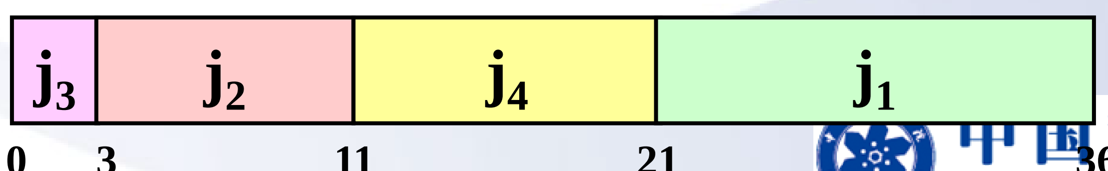
- 7. 具有不同执行时间的作业调度问题
  - 问题二：作业只有执行时间要求
    - 例，四个作业及其执行时间如下：
    - 调度方式一：
      - ✓ j_1完成时间：15
      - ✓ j_2完成时间：15+8=23
      - ✓ j_3完成时间：23+3=26
      - ✓ j_4完成时间：26+10=36
      - ✓ 平均完成时间 = (15+23+26+36)/4=25



4.3 带有限期的作业排序

作业	时间
j_1	15
j_2	8
j_3	3
j_4	10

- 7. 具有不同执行时间的作业调度问题
  - 问题二：作业只有执行时间要求
    - 调度方式二：
      - ✓j_1完成时间：3+8+10+15=36
      - ✓j_2完成时间：3+8=11
      - ✓j_3完成时间：3
      - ✓j_4完成时间：3+8+10=21
      - ✓平均完成时间 = (36+11+3+21)/4=17.75
    - 特点：
      - ✓作业按照执行时间的非降次序安排执行的顺序。



4.3 带有限期的作业排序

- 7. 具有不同执行时间的作业调度问题
  - 问题二：作业只有执行时间要求
    - 问题二的求解策略：
      - ✓按作业执行时间的非降次序依次执行。（短作业优先）
    - 证明：对于上述问题，按作业执行时间的非降次序执行，可以得到n个作业最短平均完成时间。
      - ✓（略）。

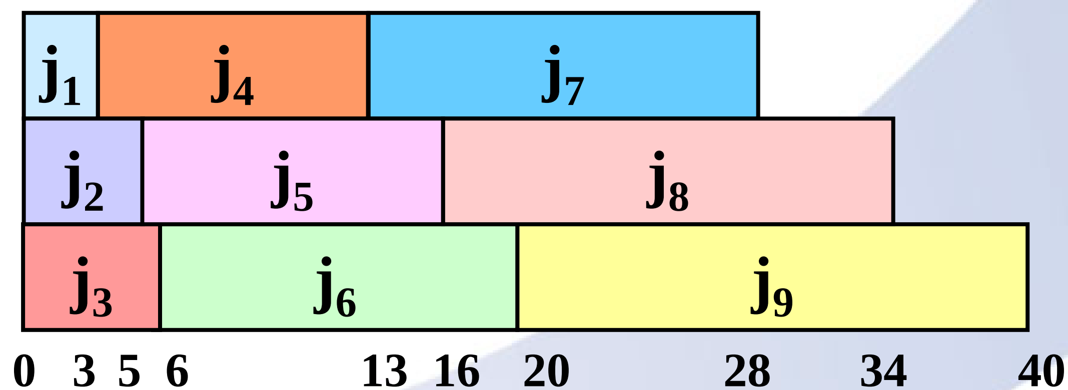
4.3 带有限期的作业排序

■ 7. 具有不同执行时间的作业调度问题

- □ 问题二的扩展：如果有多个CPU怎么办？
  - ➢ 仍按作业执行时间的非降次序，依次把作业轮换分配到各个CPU上执行即可。

    如，9个作业，3个CPU:

作业	时间
j_1	3
j_2	5
j_3	6
j_4	10
j_5	11
j_6	14
j_7	15
j_8	18
j_9	20



4.3 带有限期的作业排序

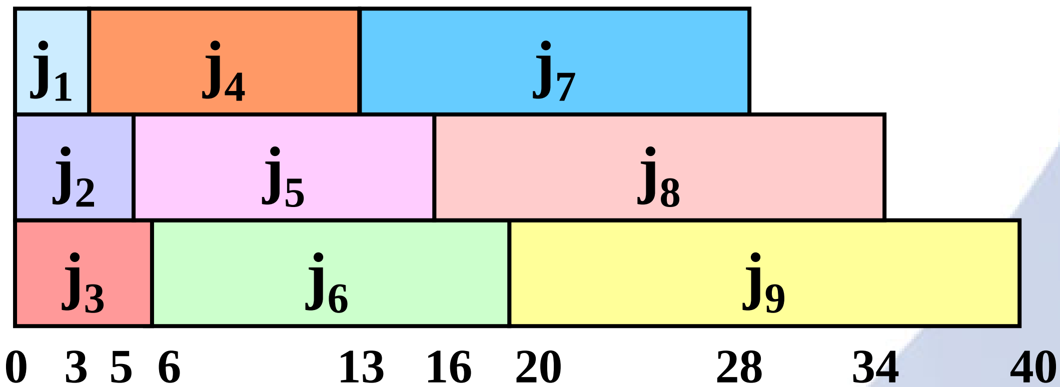
- 7. 具有不同执行时间的作业调度问题
  - 问题三：多机调度问题：
    - 给定n个作业、m个CPU，怎样调度使得n个作业能够在最短的时间内加工处理完毕？
    - 这个问题是NP完全问题，到目前为止还没有有效的解法。对于这一类问题，用贪心选择策略有时可以设计出较好的近似算法。

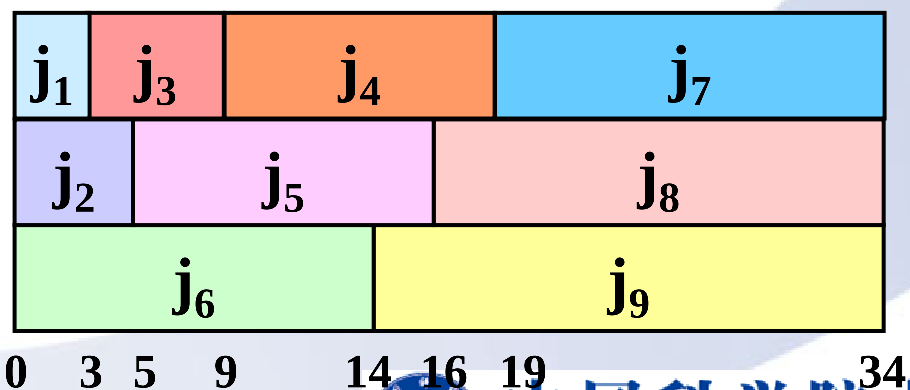
4.3 带有限期的作业排序

■ 7. 具有不同执行时间的作业调度问题

□问题三：多机调度问题：

作业	时间
j_1	3
j_2	5
j_3	6
j_4	10
j_5	11
j_6	14
j_7	15
j_8	18
j_9	20





4.3 带有限期的作业排序

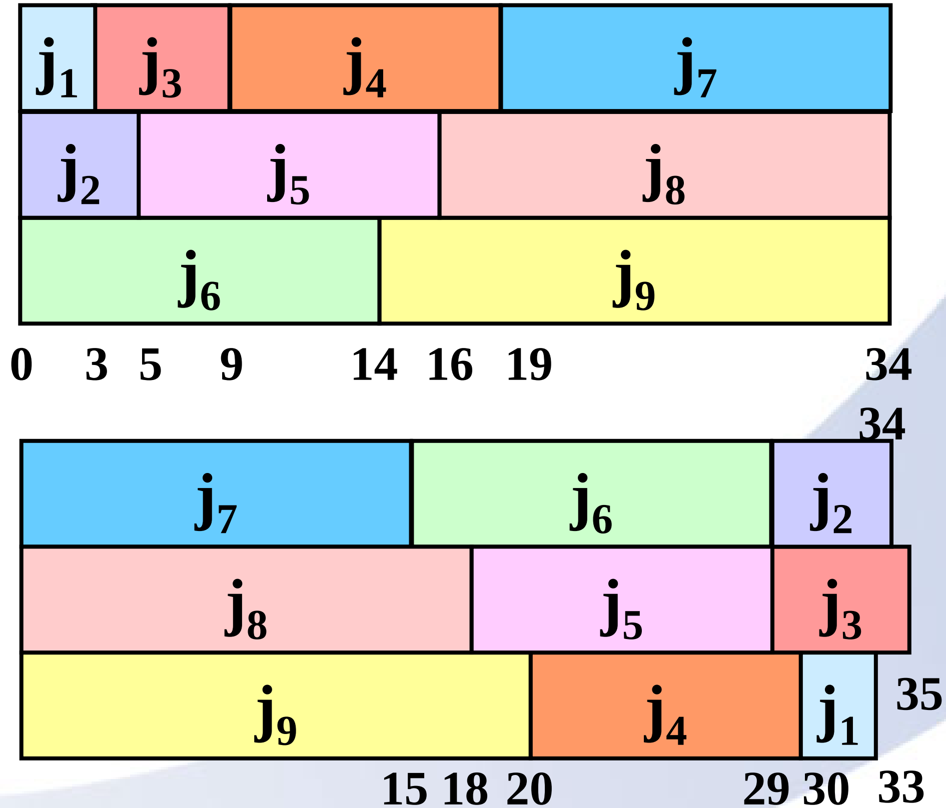
- **7. 具有不同执行时间的作业调度问题**
  - 问题三：多机调度问题的贪心算法
    - 采用长作业优先的贪心选择策略可以设计出解多机调度问题的较好的近似算法。
    - 当n≤m时，机器多任务少
      - ✓只要将机器i的[0, t_i]时间区间分配给作业i即可，此时算法只需要O(1)时间。
    - 当n>m时，
      - ✓首先将n个作业依其所需的处理时间从大到小排序。
      - ✓然后依此顺序将作业分配给空闲的处理机。
      - ✓此时算法所需的计算时间为O(nlogn)。

4.3 带有限期的作业排序

■ 7. 具有不同执行时间的作业调度问题

□问题三：多机调度问题的贪心算法

作业	时间
j_1	3
j_2	5
j_3	6
j_4	10
j_5	11
j_6	14
j_7	15
j_8	18
j_9	20



- ➢ 采用长作业优先的调度策略得到的时间为35
- ➢ 本问题的最优解是34

4.3 带有限期的作业排序

- **8. 活动安排问题**
  - 问题描述
    - 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，如演讲会场等，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$s_i<f_i$。
    - 若如果选择了活动i，则它在半开时间区间$[s_i, f_i)$内占用资源。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$s_i \ge f_j$或$s_j \ge f_i$时，活动i与活动j相容。
  - 问题：
    - 在所给的活动集合中选出最大的相容活动子集合

4.3 带有限期的作业排序

■ 8. 活动安排问题

```
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;
      }
}
```

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

4.3 带有限期的作业排序

■ 8. 活动安排问题

□讨论

- 由于输入的活动以其完成时间的非减序排列，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。
- 直观上，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。
- 也就是说，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。
- 算法greedySelector的效率极高。当输入的活动已按结束时间的非减序排列，算法只需O(n)的时间安排n个活动，使最多的活动能相容地使用公共资源。
- 如果所给出的活动未按非减序排列，可以用O(nlogn)的时间重排。

4.3 带有限期的作业排序

■ 8. 活动安排问题

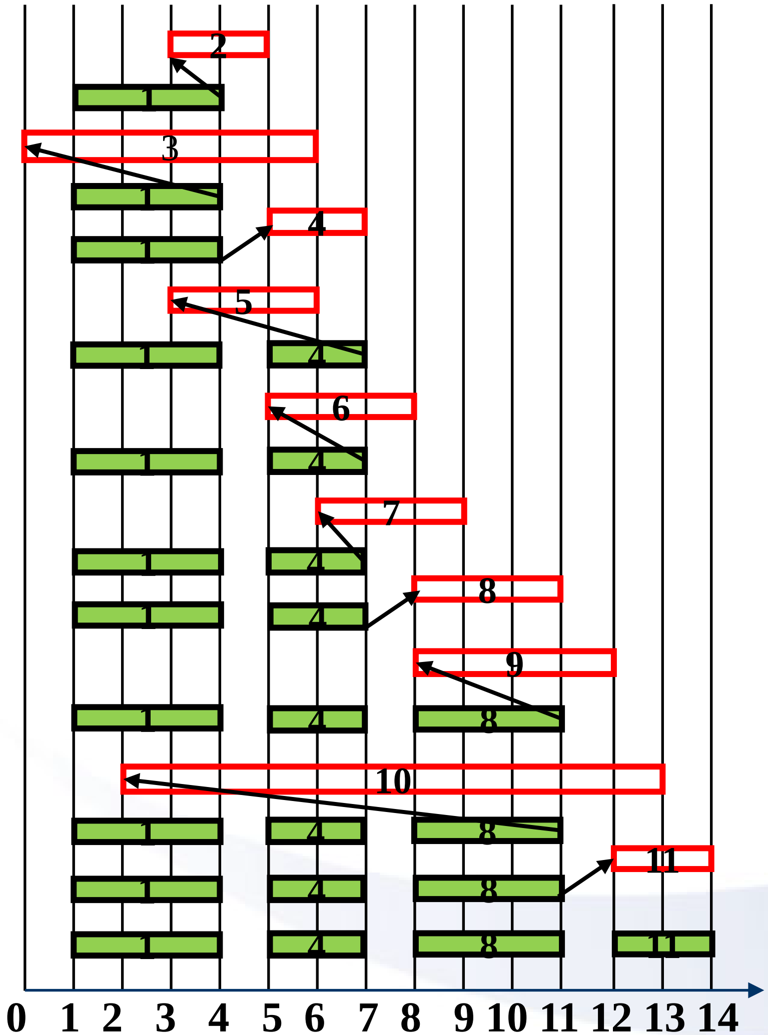
□例：设待安排的11个活动的开始时间和结束时间按结束时间的非减序排列如下：

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

4.3 带有限期的作业排序

■ 8. 活动安排问题

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



算法greedySelector 的计算过程如左图所示。
图中每行相应于算法的一次迭代。

绿色长条表示的活动是已选入集合A的活动，
空白长条表示的活动是当前正在检查相容性的活动。

4.3 带有限期的作业排序

- **8. 活动安排问题**
  - 若被检查的活动i的开始时间s_i小于最近选择的活动j的结束时间f_i，则不选择活动i，否则选择活动i加入集合A中。
  - 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greedySelector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归纳法证明。

作业-课后练习12

- 对作业排序问题证明：
  - 当且仅当子集合J中的作业可以按下述规则处理时，J才表示一个可行解。
  - 即如果J中的作业还没有分配处理时间，则将他分配在时间片[α -1, α]处理，其中α是使得$1\le r \le d_i$的最大整数r，且时间片[α -1, α]是空的。

中国科学院大学 University of Chinese Academy of Sciences

End