加工顺序问题

https://github.com/bjrjk

加工顺序问题

- 题目背景
- 符号定义
- 状态转移方程及详解
- 标记函数

https://github.com/bjrjk

题目背景

- 有n个工件需要在机器M_1和M_2上加工
- 每个工件都先在M_1上加工，然后在M_2上加工
- A_i， B_i分别表示工件i在M_1和M_2上所需的加工时间
- 安排工件的加工顺序，使得从第一个工件在M_1上加工开始到最后一个工件在M_2上加工完成，所需的总加工时间最短

https://github.com/bjrjk

符号定义

符号	意义
T	动态规划数组
A	工件在M_1上加工所需的时间数组
B	工件在M_2上加工所需的时间数组
S	机器M_1未加工的工件集合
U	工件的全集
t	机器M_2的剩余运行时间

https://github.com/bjrjk

题设

		S_{old}
	$\downarrow$	
		T_{old}

- 设集合S_{old}为机器M_1当前未加工的工件集合。
- 设集合$U=\{1,2,3,\ldots,n\}$为工件的全集。
- 初始时$S_{old}=U$。
- 截至当前，两台机器加工集合S_{old}中剩余的工件还需花费时间T_{old}秒。

https://github.com/bjrjk

		S_{old}
	↓	
		T_{old}

机器M_1

- 显然，为了使加工总时间最短，机器 M_1 不能有空闲时间。
- 因此，一旦机器M_1空闲，立即从未加工的工件集合S_{old}中选取一工件i让机器M_1开始加工。当工件i完成加工后，时间又过去了A_i秒，此时机器M_1又恢复到空闲状态，可以开始加工下一个工件。

https://github.com/bjrjk

S_{new}	$S_{new} = S_{old} - \{i\}$	S_{old}
	↓	
T_{new}	$T_{new} = T_{old} - A_i$	T_{old}

机器M_1

- 显然，为了使加工总时间最短，机器 M_1 不能有空闲时间。
- 因此，一旦机器M_1空闲，立即从未加工的工件集合S_{old}中选取一工件i让机器M_1开始加工。当工件i完成加工后，时间又过去了A_i秒，此时机器M_1又恢复到空闲状态，可以开始加工下一个工件。
- 此时机器M_1未加工的工件集合变为$S_{new} = S_{old} - \{i\}$，加工集合$S_{new}$中的工件还需花费时间$T_{new} = T_{old} - A_i$秒

https://github.com/bjrjk

S_{new}	$S_{new} = S_{old} - \{i\}$	S_{old}
t_{new}		t_{old}
		t_{old}
	$\downarrow$	
T_{new}	$T_{new} = T_{old} - A_i$	T_{old}

机器M_2(1)

- 机器M_1恰开始加工工件i的T_{old}时刻，机器M_2可能正在加工工件，也有可能处于空闲状态。设此时机器M_2还需加工t_{old}秒到达空闲状态。若M_2已经处于空闲，令$t_{old} = 0$。
- 机器M_1完成加工工件i的T_{new}时刻，机器M_2的剩余运行时间由于工件i加入M_2的任务队列中而产生了变化。定义工件i在机器M_1上完成加工后，机器M_2还需加工t_{new}秒到达空闲状态。

https://github.com/bjrjk

机器M_2(2)

$\mathrm{S}_{\mathrm{new}}$	$\mathrm{S}_{\mathrm{new}} = \mathrm{S}_{old} - \{i\}$	S_{old}
$\mathrm{t}_{\mathrm{new}}$		
	↓	
T_{new}	$T_{new} = T_{old} - A_i$	T_{old}

- 对$\mathrm{t}_{\mathrm{new}}$的取值，分如下两种情况讨论：

https://github.com/bjrjk

机器M_2(2)

$\mathrm{S_{new}}$	$\mathrm{S_{new}} = \mathrm{S}_{old} - \{i\}$	S_{old}
$\mathrm{t_{new}}$	$\mathrm{t_{new}} = B_i$	$\mathrm{t}_{old}, \mathrm{t}_{old} \le A_i$
	↓	
T_{new}	$T_{new} = T_{old} - A_i$	T_{old}

- 对$\mathrm{t_{new}}$的取值，分如下两种情况讨论：
- 当$\mathrm{t}_{old} \le A_i$时，即经过了A_i秒后，机器M_2处于空闲状态。此时，机器M_2只有任务i等待加工，故只需$\mathrm{t_{new}} = B_i$秒即可到达空闲状态。

https://github.com/bjrjk

S_{new}	$S_{new} = S_{old} - \{i\}$	S_{old}
t_{new}	$t_{new} = B_i$	$t_{old}, t_{old} \le A_i$
	$t_{new} = t_{old} - A_i + B_i$	$t_{old}, t_{old} > A_i$
	$\downarrow$	
T_{new}	$T_{new} = T_{old} - A_i$	T_{old}

机器M_2(2)

- 对t_{new}的取值，分如下两种情况讨论：
- 当$t_{old} \le A_i$时，即经过了A_i秒后，机器M_2处于空闲状态。此时，机器M_2只有任务i等待加工，故只需$t_{new} = B_i$秒即可到达空闲状态。
- 当$t_{old} > A_i$时，即经过了A_i秒后，机器M_2仍正在加工$U - S_{new}$集合中的任务。对于该集合的任务，机器M_2还需花费时间$t_{old} - A_i$秒。又多了一个新任务i，额外多出B_i的加工时间。因此机器M_2还需$t_{new} = t_{old} - A_i + B_i$秒。

https://github.com/bjrjk

机器M_2(2)

$\mathrm{S_{new}}$	$\mathrm{S_{new}} = \mathrm{S}_{old} - \{i\}$	S_{old}
$\mathrm{t_{new}}$	$\mathrm{t_{new}} = B_i$	$\mathrm{t}_{old}, \mathrm{t}_{old} \le A_i$
	$\mathrm{t_{new}} = \mathrm{t}_{old} - A_i + B_i$	$\mathrm{t}_{old}, \mathrm{t}_{old} > A_i$
	↓	
T_{new}	$T_{new} = T_{old} - A_i$	T_{old}

- 对$\mathrm{t_{new}}$的取值，分如下两种情况讨论：
- 当$\mathrm{t}_{old} \le A_i$时，即经过了A_i秒后，机器M_2处于空闲状态。此时，机器M_2只有任务i等待加工，故只需$\mathrm{t_{new}} = B_i$秒即可到达空闲状态。
- 当$\mathrm{t}_{old} > A_i$时，即经过了A_i秒后，机器M_2仍正在加工$U - \mathrm{S}_{new}$集合中的任务。对于该集合的任务，机器M_2还需花费时间$\mathrm{t}_{old} - A_i$秒。又多了一个新任务i，额外多出B_i的加工时间。因此机器M_2还需$\mathrm{t_{new}} = \mathrm{t}_{old} - A_i + B_i$秒。
- 因此综合两种情况，可得工件i完成加工后，机器M_2还需加工$\mathrm{t_{new}} = \max\{\mathrm{t}_{old} - A_i, 0\} + B_i$秒。

https://github.com/bjrjk

转移公式的引入(1)

S_{new}	$S_{new}=S_{old}-\{i\}$	S_{old}
t_{new}	$t_{new}=B_i$	$t_{old}, t_{old}\le A_i$
	$t_{new}=t_{old}-A_i+B_i$	$t_{old}, t_{old}>A_i$
	$\downarrow$	
T_{new}	$T_{new}=T_{old}-A_i$	T_{old}

- 综上所述，经观察，可以注意到：**加工时间只与机器M_1未加工的工件集合S、机器M_2还需加工的时间t有关。** 因此它们需要作为状态转移方程中的变量。
- 所以，定义动态规划数组$T_{S,t}$为 当机器M_1未加工的工件集合为S、机器M_2还需加工的时间为t时，将集合S中的全部工件加工完毕的所需时间。

https://github.com/bjrjk

转移公式的引入(2)

- 陈列前述公式如下：

$\mathrm{S}_{\mathrm{new}}$	$\mathrm{S}_{\mathrm{new}} = \mathrm{S}_{old} - \{i\}$	S_{old}
$\mathrm{t}_{\mathrm{new}}$	$\mathrm{t}_{\mathrm{new}} = B_i$	$\mathrm{t}_{old}, \mathrm{t}_{old} \leq A_i$
	$\mathrm{t}_{\mathrm{new}} = \mathrm{t}_{old} - A_i + B_i$	$\mathrm{t}_{old}, \mathrm{t}_{old} > A_i$
$\mathrm{t}_{\mathrm{new}}$	$\mathrm{t}_{\mathrm{new}} = \max\{\mathrm{t}_{old} - A_i, 0\} + B_i$	
	$\downarrow$	
T_{new}	$T_{new} = T_{old} - A_i$	T_{old}

https://github.com/bjrjk

转移公式的引入(3)

- 现展开式子
- $T_{new} = T_{old} - A_i$（自顶向下）
- $T_{old} = T_{new} + A_i$（自底向上）
- $T_{S_{old},t_{\text{old}}} = T_{S_{new},t_{new}} + A_i$
- 代入: $\begin{cases} \mathrm{S}_{\text{new}} = \mathrm{S}_{old} - \{i\} \\ \mathrm{t}_{\text{new}} = \max\{\mathrm{t}_{old} - A_i,\ 0\} + B_i \end{cases}$

https://github.com/bjrjk

转移公式的引入(4)

- $T_{S_{old},t_{\mathrm{old}}} = T_{S_{old}-\{i\},\max\{\mathrm{t}_{old}-A_i,0\}+B_i} + A_i$
- 化简，得
- $\boldsymbol{T_{S,t} = T_{S-\{i\},max\{t-A_i,0\}+B_i} + A_i}$
- **它是选取工件$\boldsymbol{i}$时，所需时间的转移公式。**
- $T_{S,t}$：当机器M_1未加工的工件集合为S、机器M_2还需加工的时间为t时，将集合S中的全部工件加工完毕的所需时间。

https://github.com/bjrjk

状态转移方程的边界条件

- $\boldsymbol{T_{S,t} = T_{S-\{i\},max\{t-A_i,0\}+B_i} + A_i}$
- 显然，转移方程的初始条件为：
- 当机器M_1未加工的工件集合为$\emptyset$、机器M_2还需加工的时间为t时，所需时间为$T_{\emptyset,t} = \mathrm{t}$。
- 转移方程的终止条件为：
- 当机器M_1未加工的工件集合为U、机器M_2还需加工的时间为0时，所需时间为$T_{U,0}$。

https://github.com/bjrjk

从贪心到动态规划(1)

- $T_{S,t} = T_{S-\{i\},max\{t-A_i,0\}+B_i} + A_i$
- 此处的工件i是任意选取的。
- 如何取到最优值呢？

https://github.com/bjrjk

从贪心到动态规划(2)

- 最直接的贪心法：遍取未加工工件集合S中的工件i，使当前阶段的$T_{S,t}$取得最小值。即令
- $\boldsymbol{T_{S,t} = \min_{i \in S} \{T_{S-\{i\},max\{t-A_i,0\}+B_i} + A_i\}}$
- 需要证明最优子结构：当前阶段的$T_{S,t}$取得最小值、下一阶段的$T_{S,t}$也取得最小值……直到终止条件也取得最小值。

https://github.com/bjrjk

最优子结构的证明(1)

- 设P是所给作业的最优调度，它所需的加工时间为$T_{S,t} = A_{P_1} + \mathrm{T}'$。
- 其中$T' = T_{S-\{P_1\},\max\{t-A_{P_1},0\}+\mathrm{B}_{P_1}}$为安排调度作业$P_2, P_3, \ldots, P_n$的所需时间。
- 假设P不是其所给流水作业的规模缩小1的子最优调度，而P'是子最优调度。

https://github.com/bjrjk

最优子结构的证明(2)

- 设$T'' = T_{S-\{P'_1\},\max\{t-A_{P'_1},0\}+\mathrm{B}_{P'_1}}$是安排调度作业$P'_2, P'_3, \ldots, P'_n$所需的时间。
- 显然有$T'' < \mathrm{T}'$。
- 则$T'_{S,t} = A_{P_1} + T'' < A_{P_1} + T' = T_{S,t}$。矛盾。
- 故作业调度问题具有最优子结构性质。

https://github.com/bjrjk

状态转移方程

- $\begin{cases} T_{\emptyset,t} = t \\ T_{S,t} = \min\limits_{i\in S}\{T_{S-\{i\},max\{t-A_i,0\}+B_i} + A_i\} \end{cases}$
- $T_{S,t}$：当机器M_1未加工的工件集合为S、机器M_2还需加工的时间为t时，将集合S中的全部工件加工完毕的所需时间。
- 求解$T_{U,0}$时，采用状态压缩的技巧减少DP数组使用的空间
- 时间复杂度$O(tn2^n)$，空间复杂度$O(t2^n)$

https://github.com/bjrjk

状态转移方程中状态变量的确定(1)

- 观察->理论?
- 事实上S, t两变量能够确定加工时间的原因如下:

https://github.com/bjrjk

状态转移方程中状态变量的确定(1)

- 观察->理论?
- 事实上S, t两变量能够确定加工时间的原因如下:
- 两机器是流水作业，计算加工的总时间不能简单的将两机器的运行时间相加得到。需以两机器的时间轴之一为主，另一时间轴作为辅助，计算总运行时间。

https://github.com/bjrjk

状态转移方程中状态变量的确定(2)

- 机器M_1未加工的工件集合为S，它能够确定主时间轴——机器M_1的运作时间，但它不包含任何关于确定工件加工顺序的信息。

https://github.com/bjrjk

状态转移方程中状态变量的确定(2)

- 机器M_1未加工的工件集合为S，它能够确定主时间轴——机器M_1的运作时间，但它不包含任何关于确定工件加工顺序的信息。
- 机器M_2还需加工的时间为t，它能够确定辅时间轴——机器M_1在运作完毕后，机器M_2完成M_1递交给它的任务所需花费的额外时间。

https://github.com/bjrjk

状态转移方程中状态变量的确定(2)

- 机器M_1未加工的工件集合为S，它能够确定主时间轴——机器M_1的运作时间，但它不包含任何关于确定工件加工顺序的信息。
- 机器M_2还需加工的时间为t，它能够确定辅时间轴——机器M_1在运作完毕后，机器M_2完成M_1递交给它的任务所需花费的额外时间。
- 关于工件加工顺序的信息，在状态转移过程中进行第一维变量的决策过程中（选取下一加工的工件），被巧妙地蕴含在第二维变量里（机器M_1与机器M_2同时运行所抵消的时间）。

https://github.com/bjrjk

状态转移方程中状态变量的确定(3)

- 两变量所共同组成的状态恰好涵盖了求得总运行时间的所有要素（加工工件的集合及加工的顺序），因此上述两状态能够唯一确定加工时间。
- 事实上，对于动态规划问题，几乎不可能存在一个一般性的思路，一劳永逸的解决全部此类问题。而是需要根据具体问题进行具体分析。

https://github.com/bjrjk

回溯路径的记录

- $\begin{cases} T_{\emptyset,t} = t \\ T_{S,t} = \min\limits_{i \in S} \{T_{S-\{i\},max\{t-A_i,0\}+B_i} + A_i\} \end{cases}$
- 站在状态(S,t)处，想要回溯到上一状态，需要记录哪些值？
- 为了输出工件，需要记录$\boldsymbol{i}$，从而可以推导出上一状态的第一维变量。
- 记录$max\{t-A_i,0\}+B_i$，上一个状态的第二维变量。
- 一边回溯，一边输出$\boldsymbol{i}$即可。

https://github.com/bjrjk