

# Введение в теорию расписаний

Рассматриваются два множества:

$M = \{M_1, M_2, \dots, M_m\}$  — машины (станки, процессоры, бригады, ...)

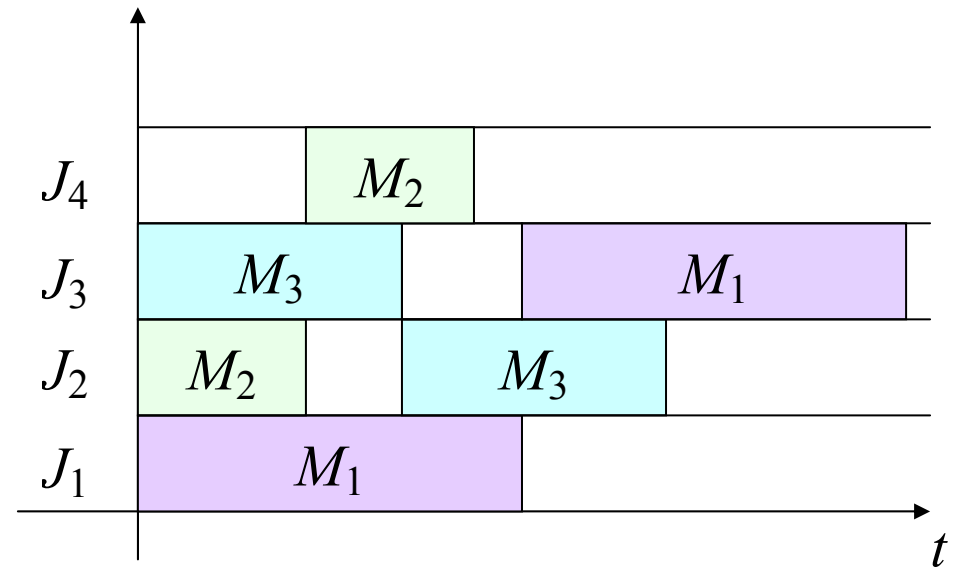
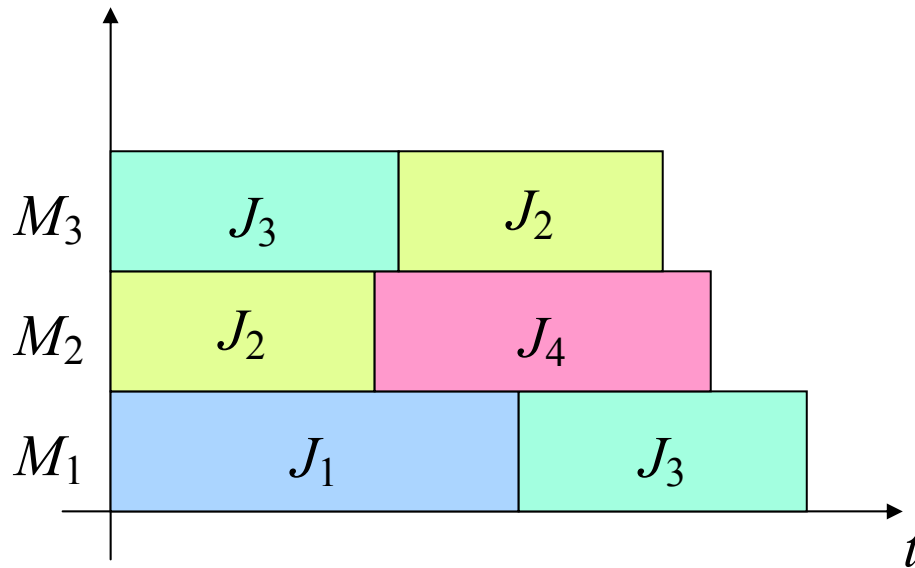
$J = \{J_1, J_2, \dots, J_n\}$  — работы (задания, пакеты задач, ...)

● *Расписание* — указание, на каких машинах и в какое время должны выполняться работы.

В каждый момент времени каждая машина выполняет не более одной работы, и каждая работа выполняется на одной машине или не выполняется вовсе.

## Два типа диаграмм Ганта

Одно решение, представленное на двух диаграммах



## Характеристики работ

Работы состоят из операций:  $J_i = \{O_{i_1}, O_{i_2}, \dots, O_{i_{n_i}}\}$

Операция  $O_{i_j}$  требует  $p_{ij}$  времени и может выполняться на одной из машин множества  $\mu_{ij} \subseteq \{M_1, \dots, M_m\}$ .

Если  $|\mu_{ij}| = 1, \forall i, j$ , то получаем модель с предписаниями.

Если  $|\mu_{ij}| = m, \forall i, j$ , то получаем модель с параллельными машинами.

Для работы  $J_i$  известны:

$r_i \geq 0$  — время появления первой операции  $O_{i_1}$

$d_i \geq 0$  — директивное время окончания последней операции  $O_{i_{n_i}}$

$w_i \geq 0$  — важность (вес, ценность) работы  $J_i$

# Классификация задач теории расписаний

## Краткая запись задачи $\alpha \mid \beta \mid \gamma$

$\alpha$  — характеристики машин;  $\beta$  — характеристики работ;

$\gamma$  — целевая функция задачи;

### Варианты для $\beta$ :

$\beta_1 = pmtn$  (preemption) разрешаются прерывания;

$\beta_2 = prec$  (precedence relations) условия предшествования на множестве работ (цепи, деревья, сети);

$\beta_3 = r_i$  — время поступления на обслуживание

$\beta_4 \in \{p_{ij}=1; p_{ij} \in \{0,1\}; p_{ij} = p_{ij}(t), \dots\}$  — уточнения для времени выполнения операций.

$\beta_5 = d_i$  — директивные сроки окончания работ;

$\beta_6 = p\text{-batching}$  ( $s\text{-batching}$ ) — работы разбиваются на группы, и в каждой группе берется максимум (сумма) времён выполнения работ;

## Характеристики машин

Поле  $\alpha$  состоит из двух частей  $\alpha = \alpha_1 \alpha_2$ :

$\alpha_1$  — характеристики машин,

$\alpha_2$  — число машин.

Если  $\alpha_1 \in \{\emptyset, P, Q, R\}$ , то  $n_i = 1 \ \forall J_i$ , то есть каждая работа состоит ровно из одной операции.

$\alpha_1 = \emptyset$  — для каждой работы задана машина для ее выполнения,

$\alpha_1 = P$  — машины параллельны и одинаковы  $p_{ij} = p_i$ ,

$\alpha_1 = Q$  — машины параллельны, но различаются скоростями  $p_{ij} = p_i / s_j$ ,

$\alpha_1 = R$  — машины параллельны, длительности выполнения работ произвольны, но  $p_{ij} = p_i / s_{ij}$ .

Если  $\alpha_1 \in \{G, X, J, F, O\}$ , то  $n_i \geq 1$ , то есть у каждой работы может быть несколько операций.

$\alpha_1 = J$  (*job shop*, рабочий цех) — у каждой операции своя машина  $|\mu_{ij}| = 1$  и линейный порядок выполнения операций  $O_{i_1} \rightarrow O_{i_2} \rightarrow \dots \rightarrow O_{i_{n_i}}$ .

$\alpha_1 = F$  (*flow shop*, потоковая линия) — машины упорядочены  $M_1, M_2, \dots, M_m$  и каждая работа проходит все машины в этом порядке,  $n_i = m$  и  $\mu_{ij} = M_j, \forall i$ .

$\alpha_1 = O$  (*open shop*, открытая линия) — каждая работа состоит из  $m$  операций ( $n_i = m$ ), но  $\mu_{ij} = \{M_1, \dots, M_m\}$  и на множестве операций нет условий предшествования,

$\alpha_1 = X$  (*mixed shop*, смешанный цикл) — смесь  $J$  и  $O$ ,

$\alpha_1 = G$  (*general case*) — произвольный порядок предшествования на операциях (как в календарном планировании).

## Целевые функции

Обозначим через  $c_i$  — время окончания работы  $J_i$ . Рассматриваются два типа минимизируемых целевых функций:

$$f(c) = \max_i f_i(c_i), \quad f(c) = \sum_{i=1}^n f_i(c_i).$$

### Примеры целевых функций:

$C_{\max} = \max_{i=1, \dots, n} c_i$  — время окончания всех работ;

$L_{\max} = \max_{i=1, \dots, n} (c_i - d_i)$  — запаздывание относительно директивных сроков;

$D_{\max} = \max_{i=1, \dots, n} |c_i - d_i|$  — отклонение от директивных сроков;

$F_{\max} = \max_{i=1, \dots, n} (\max\{0, d_i - c_i\})$  — опережение директивных сроков;

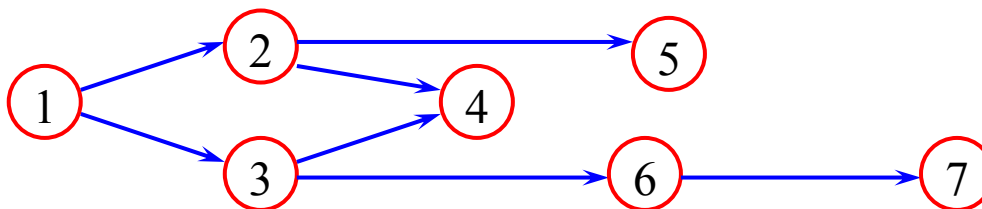
$\sum_{i=1}^n w_i c_i$  — взвешенная сумма окончания работ.

# Примеры задач теории расписаний

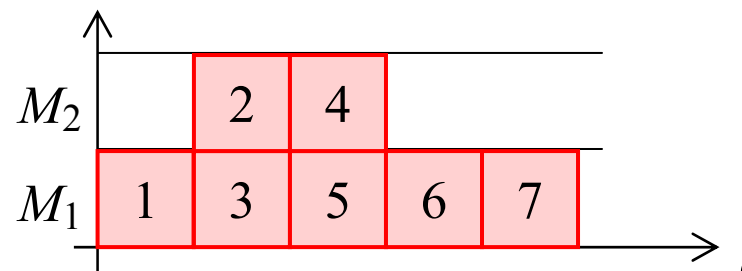
**Пример 1.**  $P \mid pres, p_i = 1 \mid C_{\max}$

Задача поиска расписания с минимальным временем окончания всех работ на  $m$  параллельных машинах с длительностями работ  $p_i = 1$  и условиями предшествования, то есть предполагается известным ориентированный граф без циклов, вершинами которого являются работы, а дуги задают частичный порядок выполнения работ.

Если  $n = 7$ ,  $m = 2$  и условия предшествования заданы графом:



то одно из допустимых решений имеет вид





## Пример 2. $1 \mid r_i, pmtn \mid L_{\max}$

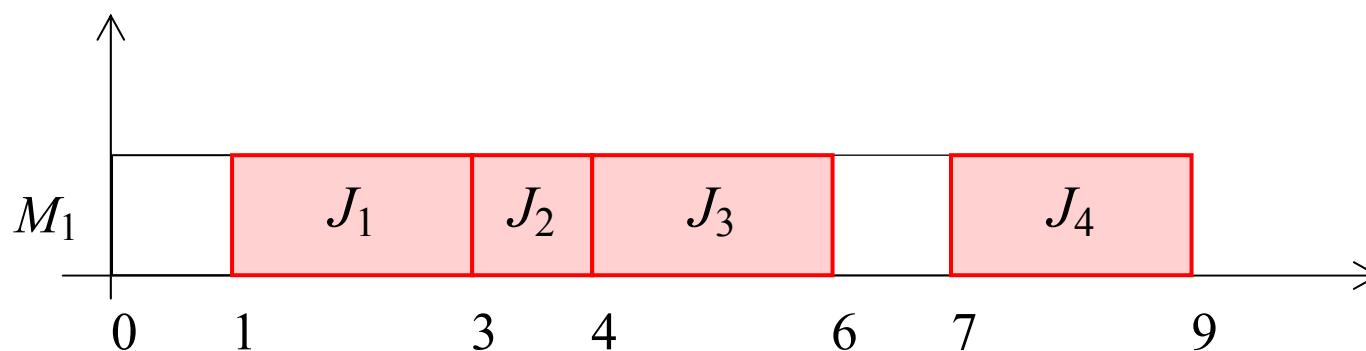
Задача на одной машине с возможностью прерывания работ, директивными сроками окончания работ и произвольными временами появления работы. Требуется найти расписание  $\{c_i\}_{i=1}^n$ , минимизирующее максимальное запаздывание, то есть

$$L_{\max} = \max_{i=1, \dots, n} (c_i - d_i) \rightarrow \min$$

Для  $n=4$  и

$i$	1	2	3	4
$p_i$	2	1	2	2
$r_i$	1	2	2	7
$d_i$	2	3	4	8

Одно из допустимых решений имеет вид:



$$L_{\max} = \max \{3 - 2; 4 - 3; 6 - 4; 9 - 8\} = 2.$$

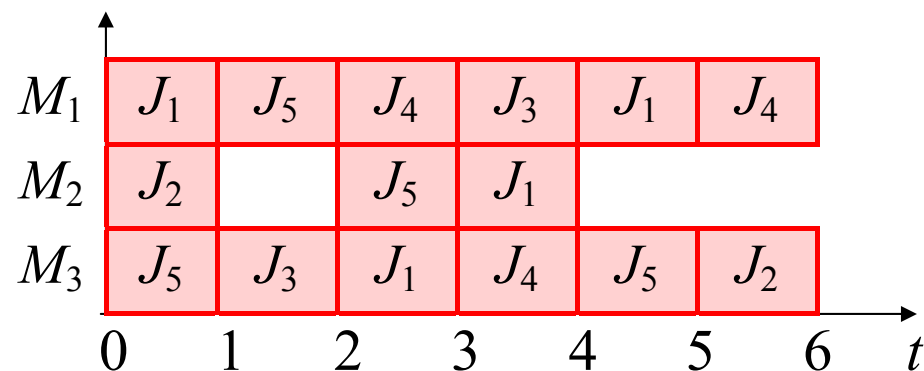
### Пример 3. $J3 \mid p_{ij} = 1 \mid C_{\max}$

Задача поиска расписания с минимальным временем окончания всех работ на трех машинах, образующих систему *job shop* — рабочий цех; длительности всех операций равны 1; у каждой работы свое множество операций; для каждой операции указана машина для ее выполнения.

При  $n = 5$ ,  $m = 3$  и матрице

	Машины				
$J_1$	$M_1$	$M_3$	$M_2$	$M_1$	
$J_2$	$M_2$	$M_3$	—	—	
$J_3$	$M_3$	$M_1$	—	—	
$J_4$	$M_1$	$M_3$	$M_1$	—	
$J_5$	$M_3$	$M_1$	$M_2$	$M_3$	

Одно из допустимых решений задачи имеет вид:



Заметим, что машина  $M_1$  обязана работать не менее 6 единиц времени (2 для  $J_1$ , 1 для  $J_3$ , 2 для  $J_4$ , 1 для  $J_5$ ), то есть нашли оптимум!

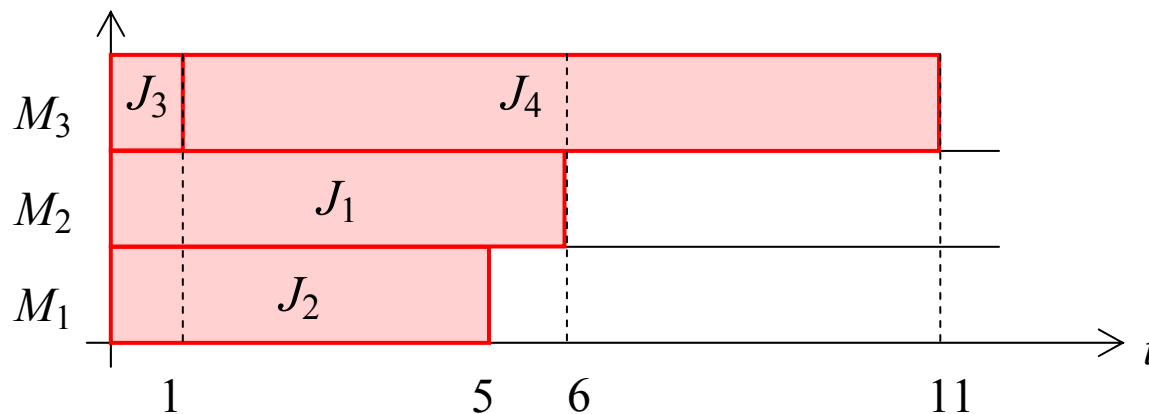
#### Пример 4. $R3 \mid d_i \mid D_{\max}$

Задача поиска расписания, минимизирующего максимальное отклонение времен завершения работ от директивных сроков на трех параллельных машинах.

При  $n = 4$ ,  $m = 3$  и матрице длительностей выполнения работ  $p_{ij}$

	$M_1$	$M_2$	$M_3$	$d_i$
$J_1$	10	6	1	5
$J_2$	5	20	3	5
$J_3$	9	30	1	6
$J_4$	6	5	10	7

Одно из допустимых решений задачи имеет вид



$$D_{\max} = \max \{ |5 - 6|; |5 - 5|; |6 - 1|; |7 - 11| \} = 5$$

$J_1$ 
 $J_2$ 
 $J_3$ 
 $J_4$

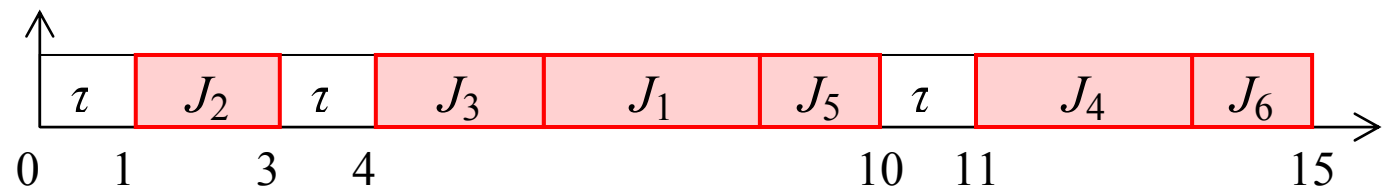
### Пример 5. $1 \mid s\text{-batch} \mid \sum w_i c_i$

Задача собрать работы в группы для обработки на одной машине так, чтобы минимизировать взвешенную сумму окончания всех работ. В каждой группе время окончания работ равно времени окончания последней работы в группе. Длительность выполнения всей группы работ равна сумме длительностей работ. При переходе от одной группы к другой машина требует переналадки  $\tau$  (простой.)

При  $n = 6$ ,  $m = 1$ ,  $\tau = 1$  и

$i$	1	2	3	4	5	6
$p_i$	3	2	2	3	1	1
$w_i$	1	2	1	1	4	4

Одно из допустимых решений при разбиении на 3 группы:  $\{J_2\}$ ,  $\{J_3, J_1, J_5\}$ ,  $\{J_4, J_6\}$  имеет вид



$$\sum_{i=1}^6 w_i c_i = w_2 \cdot 3 + (w_3 + w_1 + w_5) \cdot 10 + (w_4 + w_6) \cdot 15.$$

# Задачи теории расписаний на одной машине

Первые публикации появились в 1955 – 1956 гг (Jackson, Smith)

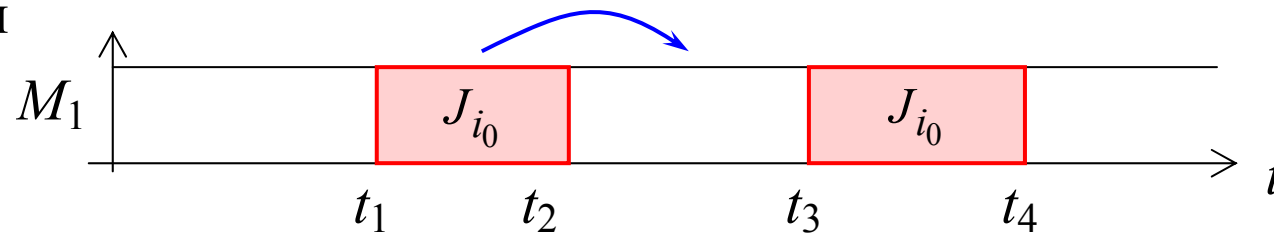
Рассмотрим задачу с  $r_i \equiv 0$  и минимизируемой функцией

$f_{\max}(c_i) = \max_{i=1, \dots, n} f_i(c_i)$ ,  $f_i$  — монотонно возрастающая функция; прерываний

работ разрешены, то есть  $1 \mid pmtn \mid f_{\max}$

**Теорема 1.** Среди оптимальных решений найдется решение без прерывания и простоя машины.

**Доказательство.** Пусть в оптимальном решении работа  $J_{i_0}$  выполнялась с прерыванием



Тогда изменим расписание, сохранив  $c_{i_0} = t_4$ , а работы из интервала  $[t_2, t_3]$  сдвинем влево к  $t_1$ . Так как  $f_i$  — монотонно возрастающая функция, то новое решение также будет оптимальным. ■

## Задача 1 | *prec* | $f_{\max}$

Решение задается перестановкой  $\pi = (\pi_1, \dots, \pi_n)$ . Величина  $\pi_i$  задает номер работы, стоящей на  $i$ -м месте в перестановке  $\pi$ . Отношения предшествования задаются матрицей  $A$ :  $a_{ij} = 1$ , если работа  $J_i$  предшествует работе  $J_j$  и  $a_{ij} = 0$  в противном случае.

### Идея алгоритма

Пусть  $N = \{1, \dots, n\}$  — множество всех работ и  $P(N) = \sum_{i \in N} p_i$ . Тогда в оптимальном решении последней работой будет работа, которая не имеет последователей и дает  $\min_{i \in N} f_i(P(N))$ .

## Алгоритм Лаулера

1. For  $i := 1, \dots, n$  do  $n(i) := \sum_{j=1}^n a_{ij}$ ;

2.  $S := \{1, \dots, N\}$ ;  $p := \sum_{i \in S} p_i$ ;

3. For  $k := n, \dots, 1$  do

Найти  $j \in S$ , для которого  $n(j) = 0$  и  $f_j(p) = \min_{i \in S} f_i(p)$ ;

Положить  $S \setminus \{j\}$ ;  $\pi_k := j$ ;  $p := p - p_j$ ;

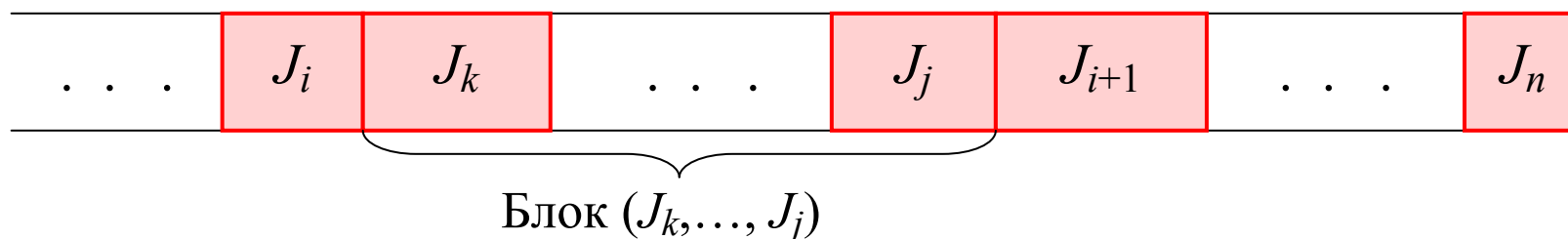
For  $i := 1, \dots, n$  do

if  $a_{ij} = 1$  then  $n(i) := n(i) - 1$ .

Трудоемкость алгоритма  $T \approx O(n^2)$ .

**Теорема 2.** Алгоритм Лаулера строит оптимальную перестановку  $\mathcal{P}$ .

**Доказательство.** Перенумеруем все работы так, чтобы  $\mathcal{P}(i) = i$ ,  $i = 1, \dots, n$ . Предположим, что  $\mathcal{P}$  не является оптимальным решением, и пусть  $\sigma = (\sigma(1), \dots, \sigma(n))$  — оптимальное решение. Найдем в нем первый номер с конца, где  $j = \sigma(i) \neq i$  и  $\sigma(i+1) = i+1$ :



Согласно алгоритму Лаулера, работа  $J_i$  может быть поставлена сразу перед  $J_{i+1}$ , так как у нее нет последователей в блоке  $(J_k, \dots, J_j)$ . Но  $f_i(p) \leq f_j(p)$ ,  $p = \sum_{l=1}^i p_l$ . Значит, вставка  $i$  перед  $i+1$  не увеличит целевую функцию и новое решение также является оптимальным. Действуя аналогично, мы уберем все нарушения, переходя от одного оптимального решения к другому, и в итоге получим  $\mathcal{P}$ . ■



## Задача 1 | $prec, pmtn, r_i | f_{\max}$

По-прежнему  $f_{\max} = \max_{i=1, \dots, n} f_i(c_i)$  и  $f_i(x)$  — монотонно возрастающие функции. Времена прихода работ  $r_i \geq 0$  могут не быть согласованными с частичным порядком, то есть  $a_{ij} = 1$  ( $i \rightarrow j$ ), но  $r_j < r_i + p_i$ . Поэтому сначала модифицируем величины  $r_i$ . Занумеруем работы так, что  $i < j$  при ( $i \rightarrow j$ ) и упорядочим пары  $e = (i \rightarrow j)$  по возрастанию  $j$ . Если всего пар  $|E|$  штук, то алгоритм пересчета величин  $r_i$  может быть записан следующим образом.

### Алгоритм Modify $r_i$

For  $e := 1, \dots, |E|$  do

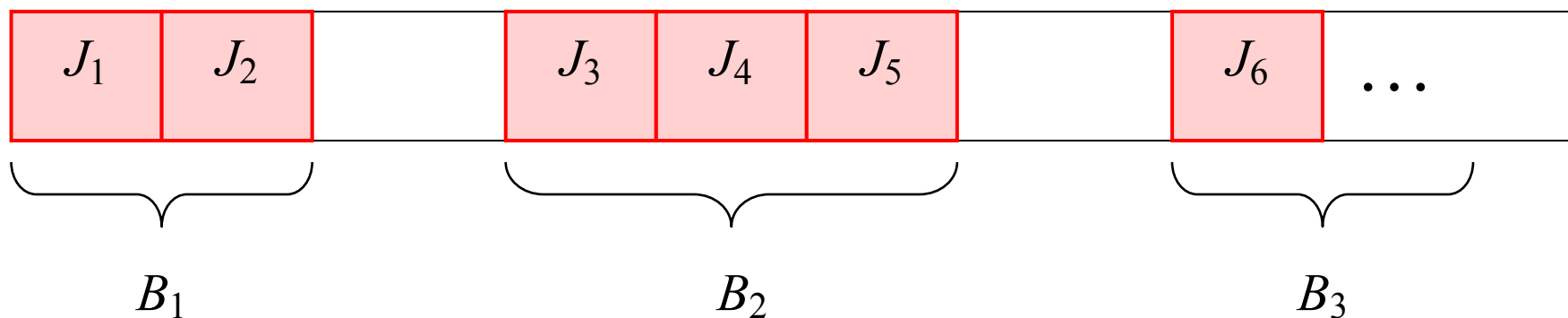
$r_j := \max \{ r_j, r_i + p_i \};$

## Разбиение на блоки

Упорядочим работы так, чтобы

$$r_1 \leq r_2 \leq \dots \leq r_n .$$

Этот порядок порождает допустимое расписание. Оно разбивается на блоки. Блок — это максимальное подмножество работ, которое выполняется без простоя машины:



# Алгоритм построения блоков

## Алгоритм Blocks $\{1, 2, \dots, n\}$

1.  $i := 1, j := 1;$

2. While  $i \leq n$  do

$t := r_i; B_j := \emptyset;$

While  $(r_i \leq t) \ \& \ (i \leq n)$  do

$B_j := B_j \cup \{i\};$

$t := t + p_i;$

$c_i := t;$

$i := i + 1;$

2.3  $j := j + 1;$

Трудоемкость алгоритма  $T \approx O(n)$ .

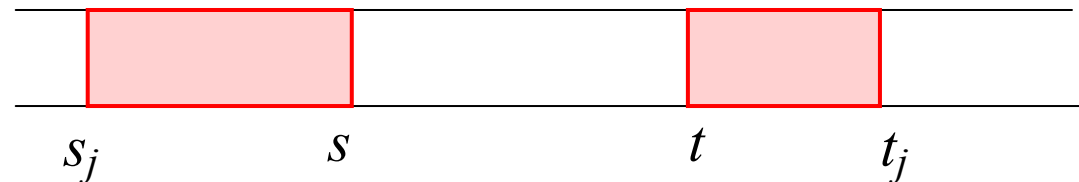
## Параметры блоков

Для блока  $B_j$  определим:  $s_j = \min_{i \in B_j} r_i$  — начало блока;

$p(B_j) = \sum_{i \in B_j} p_i$  — длительность блока;  $t_j = t(B_j) = s_j + p(B_j)$  — окончание блока.

**Теорема 3.** Для задачи  $1 \mid pres, pmtn, r_i \mid f_{\max}$  существует оптимальное расписание, в котором машина работает без простоев в интервалах  $[s_j, t_j]$ ,  $j = 1, \dots, K$ , где  $K$  — число блоков.

Доказательство. Рассмотрим оптимальное расписание и предположим, что в интервале  $[s_j, t_j]$  машина простаивает с  $s$  по  $t$ :



Рассмотрим первый такой интервал (самый левый).

Покажем, что  $\exists$  работа  $J_{i_0}$  такая, что  $r_{i_0} \leq s$ , но  $c_{i_0} > s$ . Предположим, что такой работы нет. Рассмотрим множество работ  $T$ , стартующих позже  $s$ :  $T = \{J_i \mid s_i > s\}$ . Для них

$$r = \min \{r_i \mid i \in T\} > s,$$

так как нет работы  $J_{i_0}$ . Но тогда алгоритм Blocks должен был дать простой машины в интервале  $[s, r]$ . Получили противоречие. Значит работа  $J_{i_0}$  существует. Сдвинем ее начало в  $s$  и сократим интервал  $[s, t]$  на  $p_{i_0}$ . Если  $t - s > p_{i_0}$ , то повторяем процедуру до тех пор, пока не покроем весь интервал. Но  $[s, t]$  был первым интервалом. Аналогично поступим со вторым и т.д. ■

## Оптимальное расписание для блока

Каждый блок можно рассматривать отдельно. Пусть  $f_{\max}^*(B)$  — оптимальное решение для блока  $B$  и  $f_{\max}^*(B \setminus \{j\})$  — оптимальное решение для  $B \setminus \{j\}$ . Так как  $f_i$  — монотонно неубывающие функции, то  $f_{\max}^*(B) \geq f_{\max}^*(B \setminus \{j\})$  и

$$f_{\max}^*(B) \geq \max_{j \in B} f_{\max}^*(B \setminus \{j\}) \quad (*)$$

В блоке  $B$  одна из работ заканчивается последней. Обозначим ее через  $J_l$ . Она не имеет последователей в  $B$  и  $f_l(t(B)) = \min\{f_j(t(B)) \mid j \in B \text{ и } j \text{ не имеет последователей в } B\}$ . Очевидно, что

$$f_{\max}^*(B) \geq f_l(t(B)) \quad (**)$$

Удалим работу  $J_l$  из  $B$  и найдем оптимальное решение для этой подзадачи. Оно снова будет иметь блочную структуру. Простой машины в интервале  $[s_j, t_j]$  будет соответствовать времени выполнения работы  $J_l$  и

$$f_{\max}^*(B) = \max\{f_{\max}^*(B \setminus \{J_l\}), f_l(t(B))\}.$$

В силу неравенств (\*) и (\*\*) это значение будет оптимальным. Применяя алгоритм рекурсивно, получаем оптимальное решение задачи.

## Общая схема алгоритма 1 | $prec, pmtn, r_i$ | $f_{\max}$

1.  $S := \{1, \dots, n\}$
2.  $f_{\max}^* := \text{Decompose}(S)$

### Procedure Decompose ( $S$ )

1. If  $S = \emptyset$  then return  $-\infty$
2. If  $S = \{i\}$  then return  $f_i(r_i + p_i)$   
else 2.1. Call Blocks ( $S$ )

$f := -\infty$

For all blocks  $B$  do

Найти  $l$ :  $f_l(t(B)) = \min\{f_j(t(B)) \mid j \in B \text{ и } j \text{ не имеет в } B \text{ последователей}\}$ ;

$h := \text{Decompose}(B \setminus \{J_l\})$

$f := \max\{f, h, f_l(t(B))\}$

2.4. return  $f$



Трудоемкость алгоритма  $T = O(n^2)$

Число прерываний не более  $(n - 1)$ , т.к. каждое прерывание дает разбиение на блоки.

Если  $r_i = 0$  для всех  $i \in S$ , то получаем алгоритм Лаулера.

**Упражнение.** Разработать точный полиномиальный алгоритм для задачи  $1 \mid prec, p_i = 1, r_i \mid f_{\max}$ .