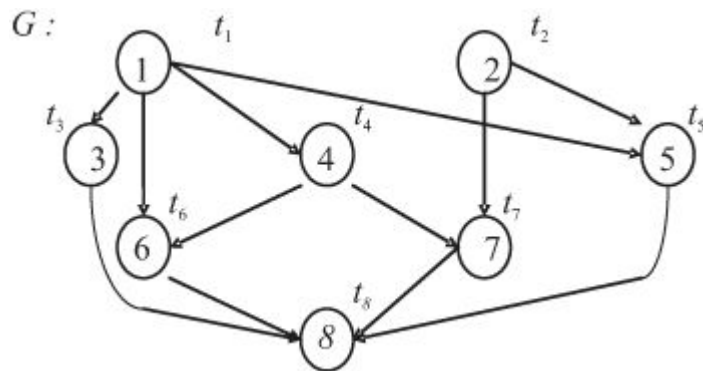


Лекция 5

Алгоритмы определения ПМВНР и ранних и поздних сроков выполнения операций

Алгоритм определения ПМВНР

Матрица следования



1									2
2									3
3	1								1
4	1								2
5	1	1							4
6	1			1					4
7		1		1					2
8			1		1	1	1		1

$S = A^T$, где A – матрица смежности.

Определение. Путь максимальной длины $T_{кр}$ в информационном графе G назовём **критическим**.

Внимание. Критических путей может быть несколько.

Определение. Пусть в G существуют прямые связи – ребра – (a, b) , (b, c) , но не существует прямой связи (a, c) . Тогда связь (a, c) называется **транзитивной**.

Алгоритм 1 дополнения треугольной матрицы S

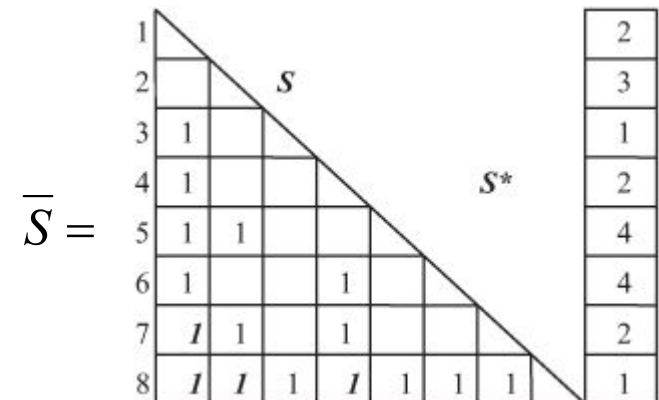
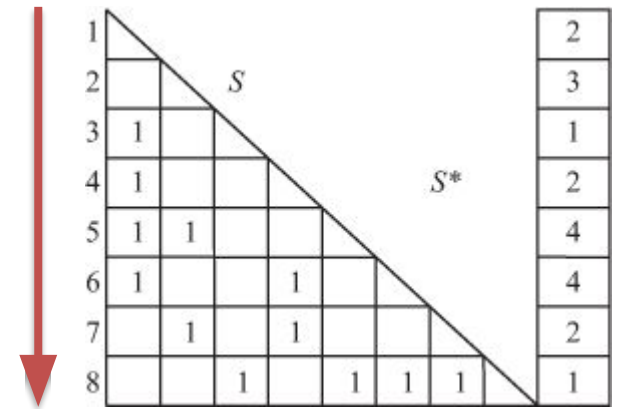
ТРАНЗИТИВНЫМИ СВЯЗЯМИ

1. Организуем просмотр сверху вниз строк матрицы следования S.

2. В очередной i-й строке организуем просмотр элементов в порядке увеличения j номеров столбцов.

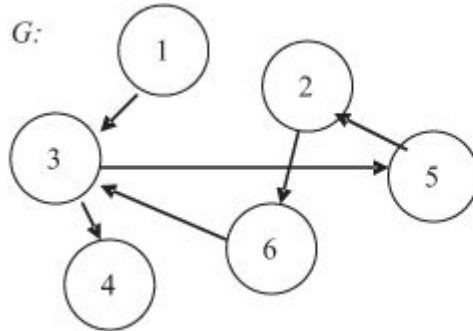
3. Если $(i, j)=1$, складываем строки i и j по операции дизъюнкции.

4. Если исходная матрица следования S не треугольная, последовательный просмотр её строк производится неоднократно до установления факта неизменности окончательной матрицы.)



Определение контуров

Пусть задан граф G :



S :

1					
2				1	
3	1				1
4			1		
5			1		
6		1			

После первого шага преобразования S принимает вид:

1					
2			1		1
3	1	1			1
4	1	1	1		1
5	1	1	1		1
6		1	1		1

После второго шага преобразования S принимает вид:

1					
2	1	I	1		1
3	1	1	I		1
4	1	1	1		1
5	1	1	1		I
6	1	1	1		1

Контур:

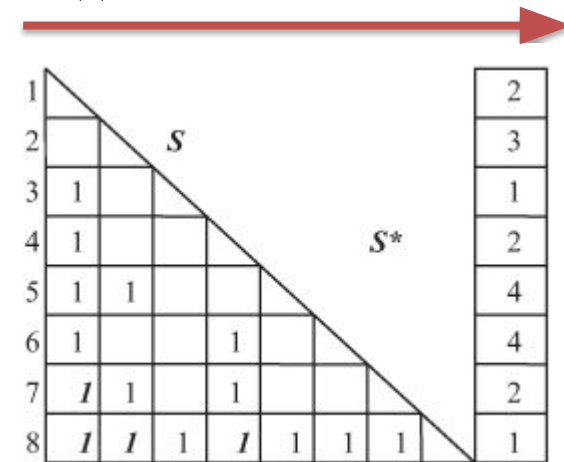
$2 \rightarrow 6 \rightarrow 3 \rightarrow 5 \rightarrow 2$

Полное множество взаимно независимых работ (ПМВНР)

Определение. Работы a и b будем называть взаимно независимыми, если в матрице следования S выполняется условие $(a, b) = (b, a) = 0$.

1. Организуем просмотр слева направо столбцов матрицы следования S .
2. В очередном j -м столбце организуем просмотр элементов в порядке увеличения i номеров строк, $i > j$. ПМВНР = $\{j\}$.

Пример. Столбец 2. Просматриваем строки начиная с 2-й.



Определение. Работы $\{i\}$, $i = 1, \dots, n$, образуют полное множество взаимно независимых работ (ПМВНР), если для любой работы $k \notin \{i\}$, существует задающая или транзитивная связь $(\mu, k) = 1$ или $(k, \nu) = 1$, $\mu, \nu \in \{i\}$, но между работами $\mu, \nu \in \{i\}$ связей нет, т.е. $(\mu, \nu) = 0$

3. Работа $\{i\} \in$ ПМВНР, если $(i, k) = (k, i) = 0$ для $k \in$ ПМВНР.

Полное множество взаимно независимых работ (ПМВНР)

Порядок рассмотрения:

В столбце 2: нулевые элементы $\{2,3,4,6\}$

Т.к. $(6,4)=1$, то множество $\{2,3,4,6\}$ разбивается на два:
 $\{2,3,4\}$, $\{2,3,6\}$

В столбце 3: нулевые элементы $\{3,4,5,6,7\}$

Т.к. $(6,4)=1$, то множество $\{3,4,5,6,7\}$

разбивается на два: $A1=\{3,4,5,7\}$, $A2=\{3,5,6,7\}$

Т.к. $(7,4)=1$, то множество $A1=\{3,4,5,7\}$

разбивается на два: $A11=\{3,4,5\}$, $A12=\{3,5,7\} \subset A2$,

1									2
2									3
3	1								1
4	1								2
5	1	1							4
6	1			1					4
7	1	1		1					2
8	1	1	1	1	1	1			1

Столбец 1. ПМВНР = $\{1, 2\}$

Столбец 2. ПМВНР = $\{2, 3, 4\}$, $\{2, 3, 6\}$

Столбец 3. ПМВНР = $\{3, 4, 5\}$, $\{3, 5, 6, 7\}$

~~Столбец 4. ПМВНР = $\{4, 5\}$~~

~~Столбец 5. ПМВНР = $\{5, 6, 7\}$~~

~~Столбец 6. ПМВНР = $\{6, 7\}$~~

~~Столбец 7. ПМВНР = $\{7\}$~~

Столбец 8. ПМВНР = $\{8\}$

Итого ПМВНР:

$\{1,2\}$, $\{2,3,4\}$, $\{3,4,5\}$,
 $\{2,3,6\}$, $\{3,5,6,7\}$, $\{8\}$.

Алгоритмы определения ранних и поздних сроков выполнения операций

Ранние и поздние сроки выполнения работ

Ранний срок τ_{1i} окончания выполнения работы – минимальный срок окончания выполнения работы.

Поздний срок $\tau_{2i}(T)$ окончания выполнения работы – максимальный срок окончания выполнения работы при ограничении на время выполнения алгоритма $T > T_{кр}$, где $T_{кр}$ – минимальное время выполнения алгоритма.

При $T = T_{кр}$ ранние сроки окончания выполнения работ, составляющих критические пути, совпадают с поздними сроками окончания их выполнения.

Алгоритм нахождения ранних сроков окончания

выполнения работ

1. Полагаем первоначально $\tau_{11} = \tau_{12} = \dots = \tau_{1m} = 0$.
2. Производя циклический обзор строк матрицы следования S , находим очередную из необработанных строк. Если все строки обработаны, выполнение алгоритма заканчивается.
3. Пусть j — номер найденной необработанной строки. Если j -я строка не содержит единичных элементов, полагаем $\tau_{1j} = t_j$. Переходим к выполнению шага 6.
4. Если j -я строка содержит единичные элементы, выбираем элементы множества $\{\tau_{11}, \dots, \tau_{1m}\}$, соответствующие номерам единичных элементов j -й строки.

Если все выбранные таким образом элементы, образующие множество $\{\tau_{1j(v)}\}$, $v = 1, \dots, k_j$, отличны от нуля, полагаем $\tau_{1j} = \max \tau_{1jv} + t_j$.

Если хотя бы один из выбранных элементов нулевой (соответствующий ранний срок ещё не найден), выполняем шаг 2.

Обработанную j -ю строку метим, чтобы исключить её повторную обработку.

Переходим к выполнению шага 2.

Примечани

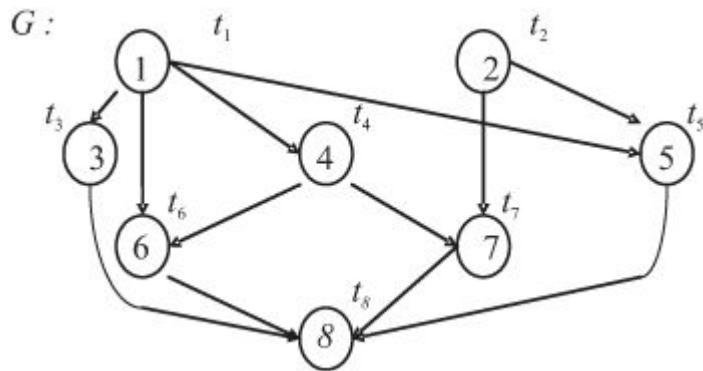
я

Если граф G не содержит контуров, зацикливание при этом невозможно.

Если матрица S треугольная, то никогда не складываются условия для многократного циклического обзора строк. Тогда ранние сроки окончания выполнения работ находятся за один последовательный просмотр строк матрицы S .

$$T_{кр} = \max \{ \tau_{11}, \dots, \tau_{1m} \}.$$

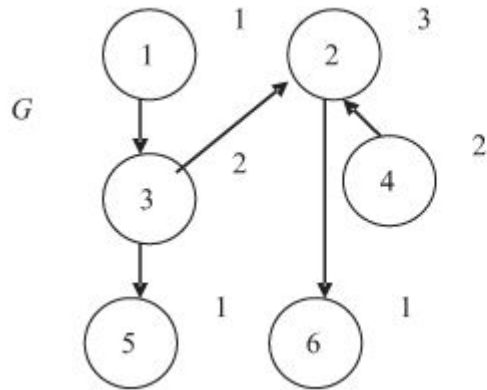
Пример:



1									2
2									3
3	1								1
4	1								2
5	1	1							4
6	1		1						4
7		1	1						2
8			1	1	1	1			1

$$\tau_{11} = t_1 = 2, \tau_{12} = t_2 = 3, \tau_{13} = \tau_{11} + t_3 = 3, \tau_{14} = \tau_{11} + t_4 = 4, \tau_{15} = \max \{ \tau_{11}, \tau_{12} \} + t_5 = 7, \tau_{16} = \max \{ \tau_{11}, \tau_{14} \} + t_6 = 8, \tau_{17} = \max \{ \tau_{12}, \tau_{14} \} + t_7 = 6, \tau_{18} = \max \{ \tau_{13}, \tau_{15}, \tau_{16}, \tau_{17} \} + t_8 = 9; T_{кр} = 9.$$

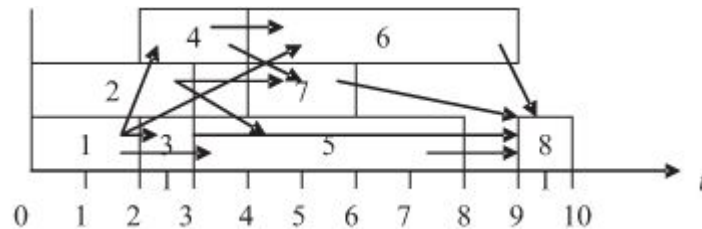
Пример



	S					T				
1										1
2			1	1						3
3	1									2
4										2
5				1						1
6		1								1

$$\begin{aligned} \tau_{11} &= 1, \\ \tau_{13} &= \tau_{11} + t_3 = 3, \\ \tau_{14} &= 2, \\ \tau_{15} &= \tau_{13} + t_5 = 4, \\ \tau_{12} &= \max \{ \tau_{13}, \tau_{14} \} + t_2 = 6, \\ \tau_{16} &= \tau_{12} + t_6 = 7. \\ T_{кр} &= 7. \end{aligned}$$

Временная диаграмма выполнения работ при ранних сроках окончания:



Алгоритм нахождения поздних сроков окончания

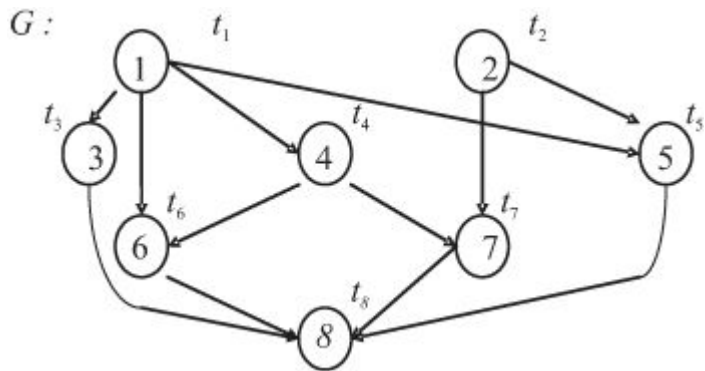
выполнения работ при заданном значении T

1. Полагаем первоначально $\tau_{21}(T) = \dots = \tau_{2m}(T) = 0$.
2. Производя циклический обзор справа налево столбцов матрицы S, находим очередной из не обработанных ещё столбцов. Если все столбцы обработаны, выполнение алгоритма заканчивается.
3. Пусть j — номер найденного необработанного столбца. Если j-й столбец не содержит единичных элементов, полагаем $\tau_{2j}(T) = T$. Переходим к выполнению шага 6.
4. Если j-й столбец содержит единичные элементы, выбираем элементы множества $\{ \tau_{21}(T), \dots, \tau_{2m}(T) \}$, соответствующие номерам единичных элементов j-го столбца.
5. Если все выбранные таким образом элементы $\{ \tau_{2jv}(T) \} \subset \{ \tau_{21}(T), \dots, \tau_{2m}(T) \}$, $v = 1, \dots, k_j$, отличны от нуля, полагаем
$$\tau_{2j}(T) = \min \{ \tau_{2jv}(T) - t_{jv} \}.$$
В противном случае выполняем шаг 2.
6. Обработанный j-й столбец метим с целью исключения его повторной обработки. Переходим к выполнению шага 2.

Пример

1

Для $T = 10$:



1								2
2								3
3	1							1
4	1							2
5	1	1						4
6	1			1				4
7		1		1				2
8			1		1	1	1	1

$$\tau_{28}(10) = 10,$$

$$\tau_{27}(10) = \tau_{28}(10) - t_8 = 9,$$

$$\tau_{26}(10) = \tau_{28}(10) - t_8 = 9,$$

$$\tau_{25}(10) = \tau_{28}(10) - t_8 = 9,$$

$$\tau_{24} = \min \{ \tau_{26}(10) - t_6, \tau_{27}(10) - t_7 \} = 5,$$

$$\tau_{23}(10) = \tau_{28}(10) - t_8 = 9,$$

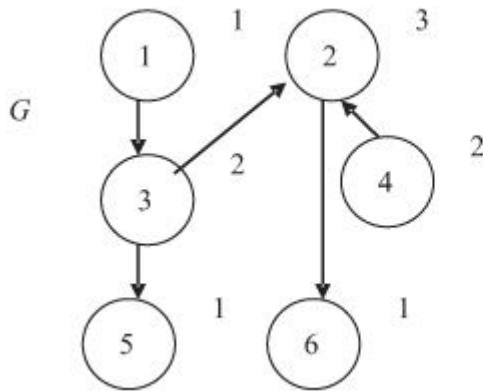
$$\tau_{22}(10) = \min \{ \tau_{25}(10) - t_5, \tau_{27}(10) - t_7 \} = 5,$$

$$\tau_{21}(10) = \min \{ \tau_{23}(10) - t_3, \tau_{24}(10) - t_4, \tau_{25}(10) - t_5, \tau_{26}(10) - t_6 \} = 3.$$

Пример

2

Для $T = 10$:



	S					T
1						1
2			1	1		3
3	1					2
4						2
5			1			1
6	1					1

$$\tau_{26}(10) = \tau_{25}(10) = 10.$$

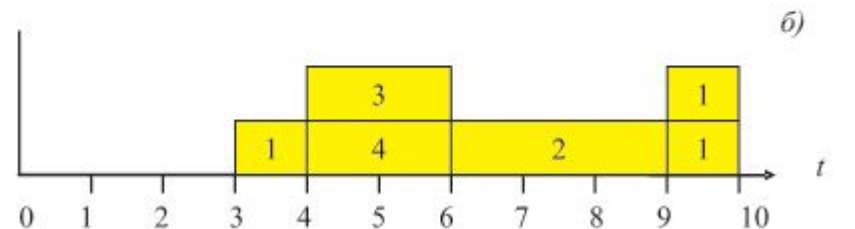
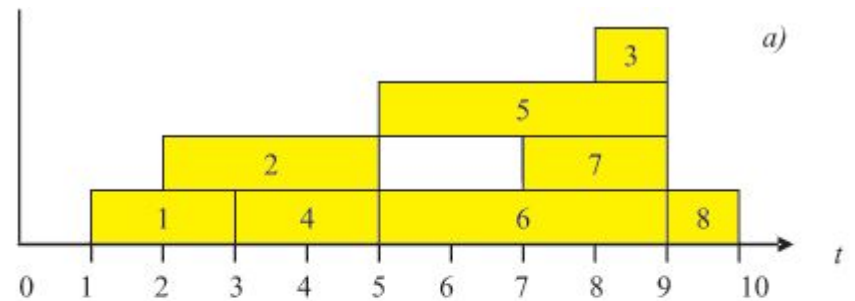
$$\tau_{22}(10) = \tau_{26}(10) - t_6 = 9.$$

$$\tau_{24}(10) = \tau_{22}(10) - t_2 = 6.$$

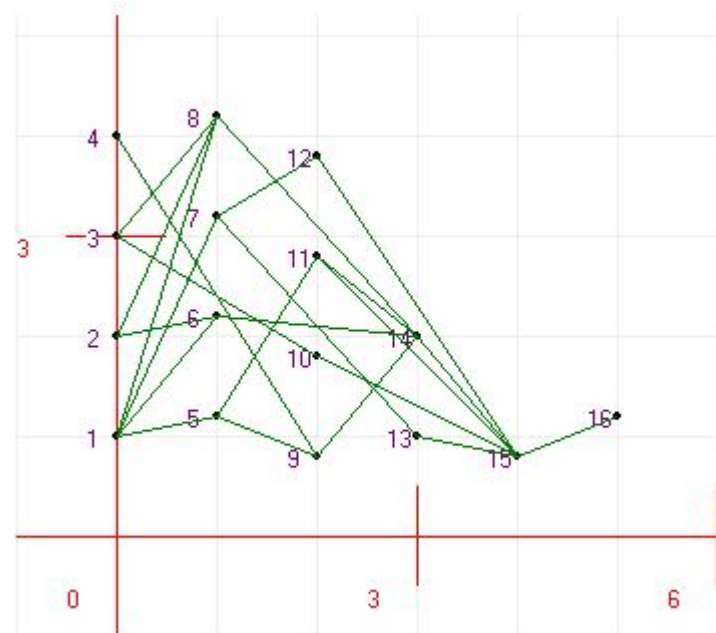
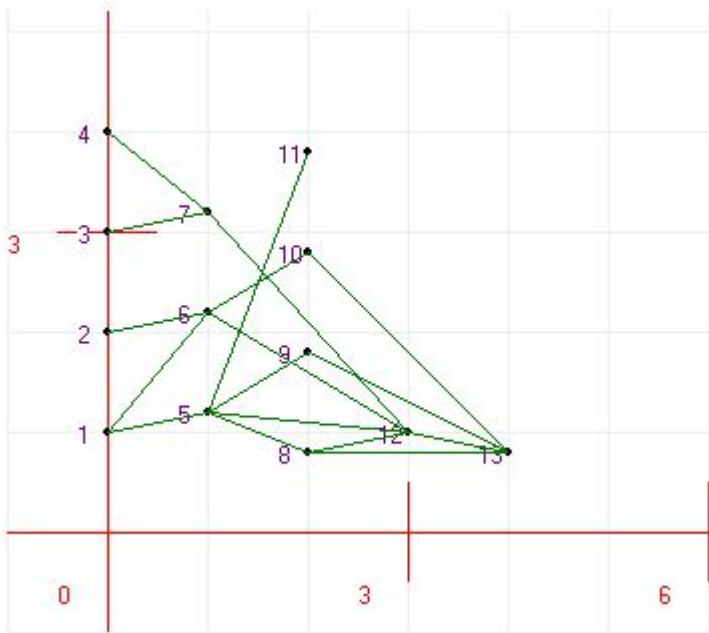
$$\tau_{23}(10) = \min \{ \tau_{22}(10) - t_2, \tau_{25}(10) - t_5 \} = 6.$$

$$\tau_{21}(10) = \tau_{23}(10) - t_3 = 4.$$

Диаграммы выполнения работ при поздних сроках окончания в примерах 1 и 2.



Самостоятельная работа



$t_1=1, t_2=2, t_3=1, t_4=3, t_5=2, t_6=5, t_7=2, t_8=2, t_9=1, t_{10}=6, t_{11}=1, t_{12}=2, t_{13}=4. \quad T = T_{кр} + 2$

$t_1=4, t_2=1, t_3=3, t_4=3, t_5=1, t_6=4, t_7=6, t_8=2, t_9=5, t_{10}=1, t_{11}=1, t_{12}=2, t_{13}=4, t_{14}=2, t_{15}=1, t_{16}=2. \quad T = T_{кр} + 3$

1. Найдите ранние и поздние сроки выполнения работ.
2. Постройте диаграммы по ранним и поздним срокам выполнения работ

Алгоритмы поиска наименьших ресурсов: процессов и времени

Плотность загрузки вычислительной системы

Пусть τ_j — произвольное значение момента окончания выполнения j -й работы:

$$\tau_{1j} \leq \tau_j \leq \tau_{2j}(T) \quad (\tau_j \in [\tau_{1j}, \tau_{2j}(T)]), \quad j = 1, \dots, m,$$

Меняя значения $\{\tau_j\}$, $j = 1, \dots, m$, но соблюдая при этом порядок следования работ, мы получим множество допустимых расписаний выполнения работ.

Определение. Функция

$$F(\tau_1, \tau_2, \dots, \tau_m, t) = \sum_{j=1}^m f(\tau_j, t),$$

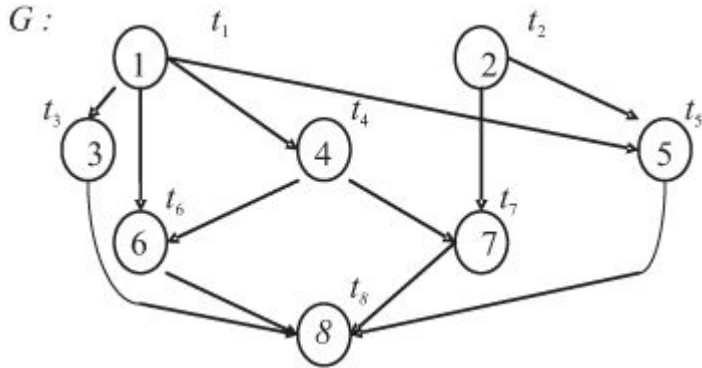
где:

$$f(\tau_j, t) = \begin{cases} 1 & \text{при } t \in [\tau_j - t_j, \tau_j] \\ 0 & \text{в противном случае,} \end{cases}$$

называется **плотностью загрузки**, найденной для значений τ_1, \dots, τ_m

Для заданных τ_1, \dots, τ_m значение функции F в каждый момент времени t совпадает с числом **одновременно** (параллельно) выполняющихся в этот момент работ.

Пример

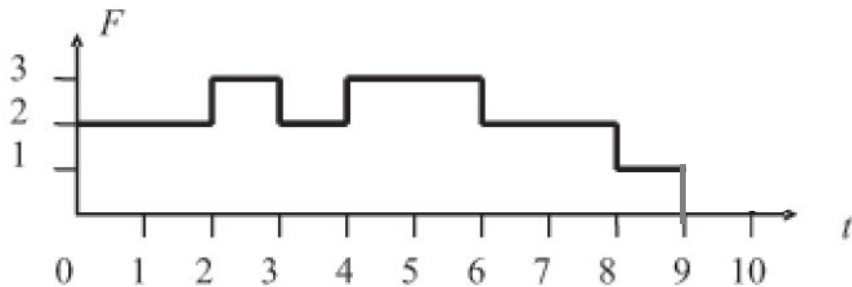


1								2
2								3
3	1							1
4	1							2
5	1	1						4
6	1			1				4
7		1		1				2
8			1		1	1	1	1

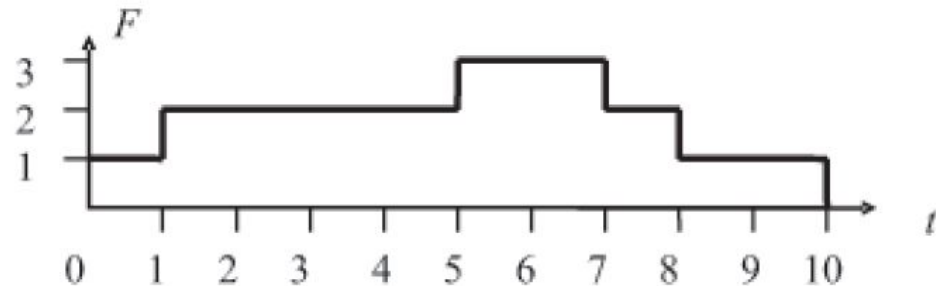
- $\tau_{11} = 2,$ $\tau_{21} = 3,$
- $\tau_{12} = 3,$ $\tau_{22} = 5,$
- $\tau_{13} = 3,$ $\tau_{23} = 9,$
- $\tau_{14} = 4,$ $\tau_{24} = 5,$
- $\tau_{15} = 7,$ $\tau_{25} = 9,$
- $\tau_{16} = 8,$ $\tau_{26} = 9,$
- $\tau_{17} = 6,$ $\tau_{27} = 9,$
- $\tau_{18} = 9;$ $\tau_{28} = 10,$
- $T_{кр} = 9.$
- $T = 10.$

$$F(\tau_1, \tau_2, \dots, \tau_m, t) = \sum_{j=1}^m f(\tau_j, t), \quad f(\tau_j, t) = \begin{cases} 1 & \text{при } t \in [\tau_j - t_j, \tau_j] \\ 0 & \text{в противном случае,} \end{cases}$$

$F(2, 3, 3, 4, 7, 8, 6, 9, t):$



$F(2, 4, 3, 4, 8, 9, 7, 10, t):$

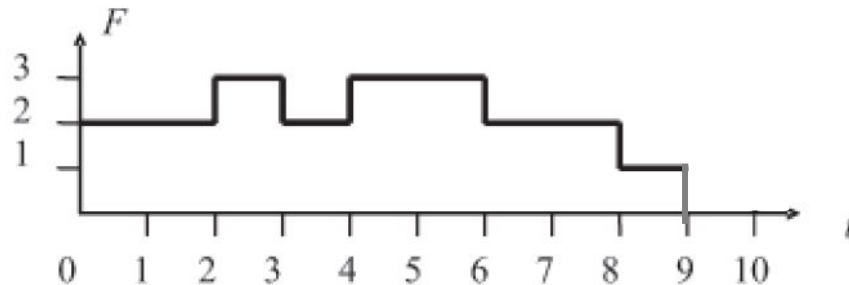
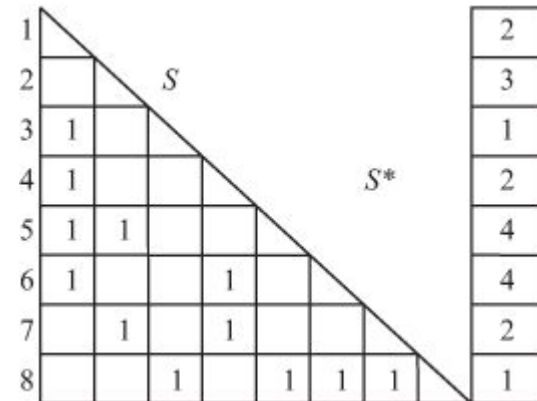


Пример

j		Промежуточное значение								
		0,5	1,5	2,5	3,5	4,5	5,5	6,5	7,5	8,5
1	[0,2]	x	x							
2	[0,3]	x	x	x						
3	[2,3]			x						
4	[2,4]			x	x					
5	[3,7]				x	x	x	x		
6	[4,8]					x	x	x	x	
7	[4,6]					x	x			
9	[8,9]									x
Итого		2	2	3	2	3	3	2	1	1

$$f(\tau_j, t) = \begin{cases} 1 & \text{при } t \in [\tau_j - t_j, \tau_j] \\ 0 & \text{в противном случае,} \end{cases}$$

$F(2, 3, 3, 4, 7, 8, 6, 9, t)$



Максимальная ширина

графа

Дано:

граф G ,

L – число ПМВНР,

r_i – число работ, образующих i -е полное множество;

$i = 1, \dots, L$,

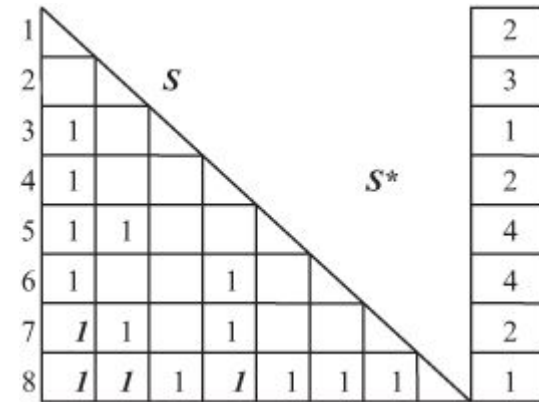
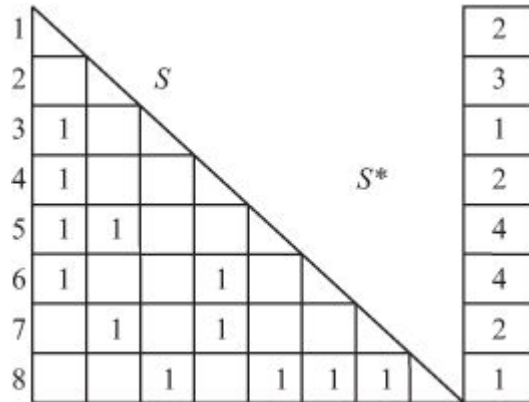
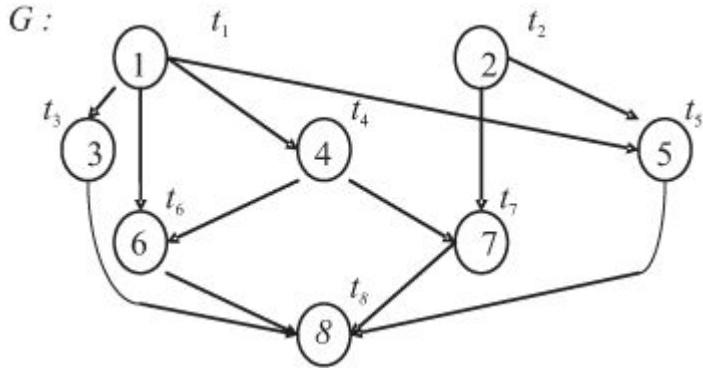
Пусть $R = \max \{r_1, \dots, r_l\}$

Тогда:

$$R = \max_{\tau_1, \tau_2, \dots, \tau_m} F(\tau_1, \dots, \tau_m, t)$$

Лемма. Минимальное число n процессоров одинаковой специализации и производительности (т.е. в однородной ВС), способных выполнить данный алгоритм за время $T \geq T_{кр}$, не превышает $R = \max \{r_1, \dots, r_l\}$, где r_i , $i = 1, \dots, L$, — число работ, входящих в i -е ПМВНР, которое составлено по взвешенному графу G , соответствующему этому алгоритму.

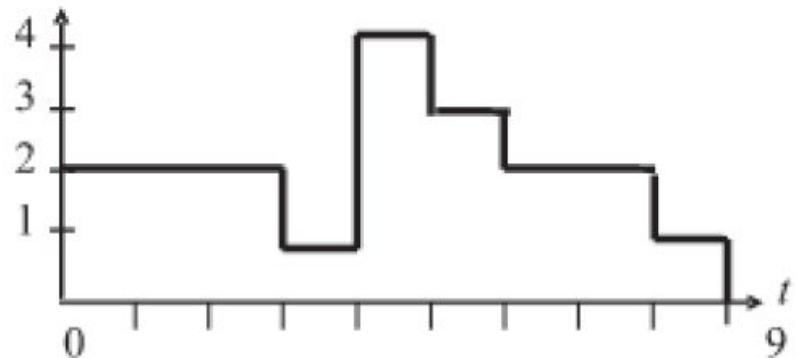
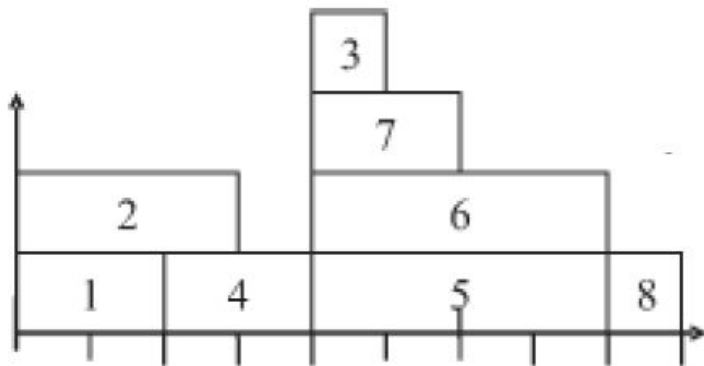
Пример



ПМВНР:

- {1,2}, {2,3,4}, {3,4,5},
- {2,3,6}, **{3,5,6,7}**, {8}.

Существует допустимое расписание, например, $\tau_1 = 2, \tau_2 = 3, \tau_3 = 5, \tau_4 = 4, \tau_5 = 8, \tau_6 = 8, \tau_7 = 6, \tau_8 = 9$ такое, при котором максимальное значение плотности загрузки F равно четырём



Загрузка отрезка для допустимого расписания

Определение. Функция

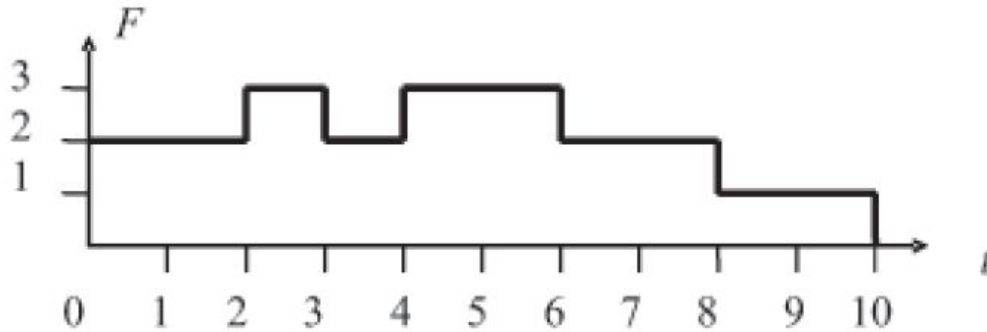
$$\Phi(\tau_1 \tau_m, \theta_1, \theta_2) = \int_{\theta_1}^{\theta_2} F(\tau_1 \tau_m, t) dt$$

называется загрузкой отрезка $[\theta_1, \theta_2] \subset [0, T]$ для заданного допустимого расписания τ_1, \dots, τ_m .

Функция Φ определяет объём работ (суммарное время их выполнения) на фиксированном отрезке их выполнения при заданном допустимом расписании.

Приме

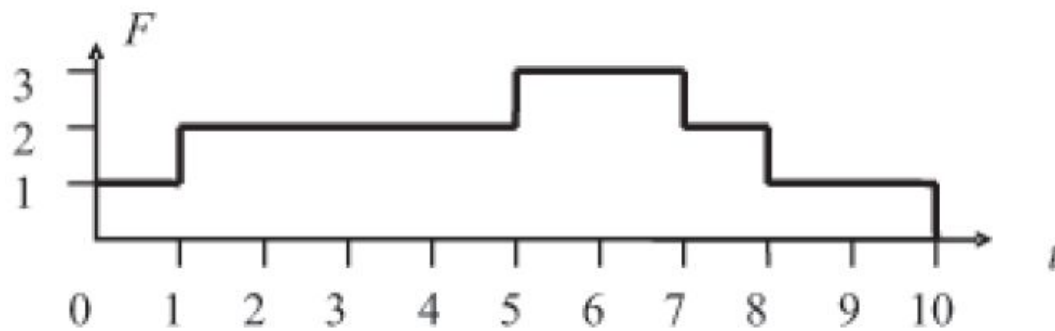
р



Для отрезка времени

$$[0, 4] \subset [0, 10]:$$

$$\Phi = 9$$



Для отрезка времени $[1, 3]:$

$$\Phi = 4$$

Для отрезка времени $[2, 5]:$

$$\Phi = 6$$

Минимальные оценки работ и вычислительных ресурсов

Определение.

Функция

$$\varphi^{(T)}(\theta_1, \theta_2) = \min_{\tau_1 \dots \tau_m} \Phi(\tau_1 \dots \tau_m, \theta_1, \theta_2)$$

называется минимальной загрузкой отрезка $[\theta_1, \theta_2] \subset [0, T]$.

Функция $\varphi^{(T)}$ определяет минимально возможный объём работ, который при данном T и при различных допустимых значениях (расписаниях) τ_1, \dots, τ_m должен быть выполнен на отрезке времени $[\theta_1, \theta_2] \subset [0, T]$.

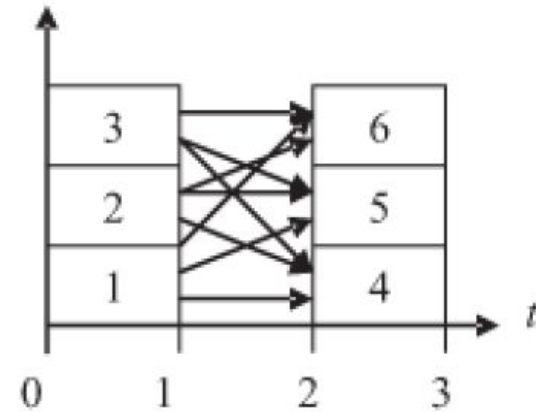
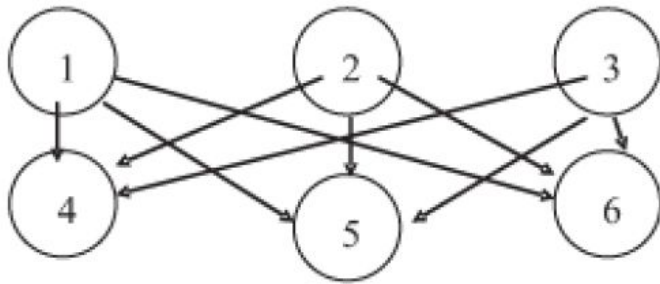
Теорема 1. Для того чтобы T было наименьшим временем выполнения данного алгоритма на однородной вычислительной системе, состоящей из n процессоров, либо чтобы n процессоров было достаточным для выполнения данного алгоритма за время T , необходимо, чтобы для данного отрезка времени $[\theta_1, \theta_2] \subset [0, T]$ выполнялось соотношение:

$$\varphi^{(T)}(\theta_1, \theta_2) \leq n(\theta_2 - \theta_1)$$

Приме

р

Пусть $T=3$.



Из теоремы имеем:

$$\varphi^{(T)}(\theta_1, \theta_2) \leq n(\theta_2 - \theta_1)$$

Откуда получаем общее соотношение:

$$n \geq \frac{\varphi^{(T)}(\theta_1, \theta_2)}{\theta_2 - \theta_1}$$

Для получения полной оценки надо перебрать все отрезки $[\theta_1, \theta_2] \subset [0, T]$ и найти:

$$\max_{[\theta_1, \theta_2] \subset [0, T]} \frac{\varphi^{(T)}(\theta_1, \theta_2)}{\theta_2 - \theta_1}$$

Продолжение примера

$$n \geq \max_{[\theta_1, \theta_2] \subset [0, T]} \frac{\varphi^{(T)}(\theta_1, \theta_2)}{\theta_2 - \theta_1} \quad (*)$$

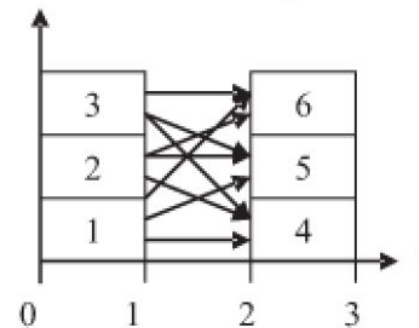
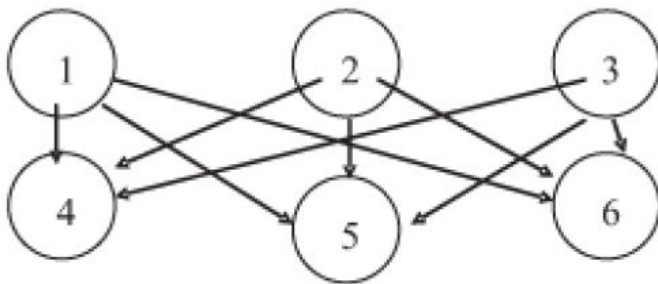
Проанализируем все возможные отрезки $[\theta_1, \theta_2] \subset [0, 3]$:

$$\phi^{(3)}(0, 1) = \phi^{(3)}(1, 2) = \phi^{(3)}(2, 3) = 0,$$

$$\phi^{(3)}(0, 2) = \phi^{(3)}(1, 3) = 3,$$

$$\phi^{(3)}(0, 3) = 6.$$

Минимальное n , удовлетворяющее условию (*): $n = 2$.



Но за время $T=3$ минимально достаточное число процессоров $n = 3$.

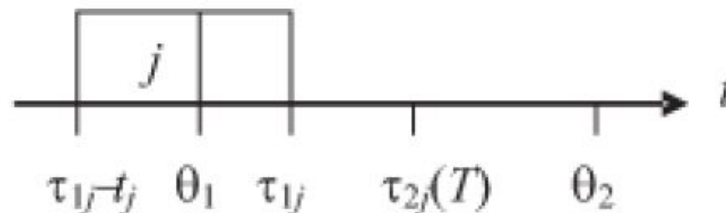
Условный объём части работы j на отрезке времени

$[\theta_1, \theta_2]$

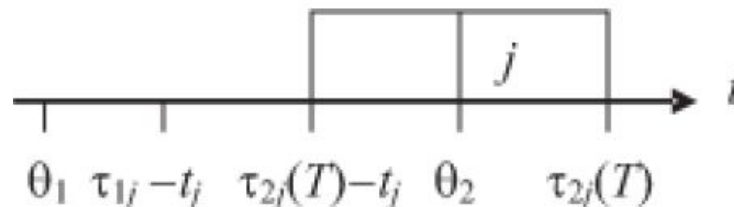
Определим
функцию:

$$\xi(x) = \begin{cases} x & \text{при } x \geq 0 \\ 0 & \text{при } x < 0 \end{cases}$$

- 1) Значение $\xi(\tau_{1j} - \theta_1)$ характеризует условный объём части работы j на отрезке времени $[\theta_1, \theta_2]$ при условии $\tau_{1j} - t_j \leq \theta_1$ и при максимальном смещении времени выполнения работы j влево.



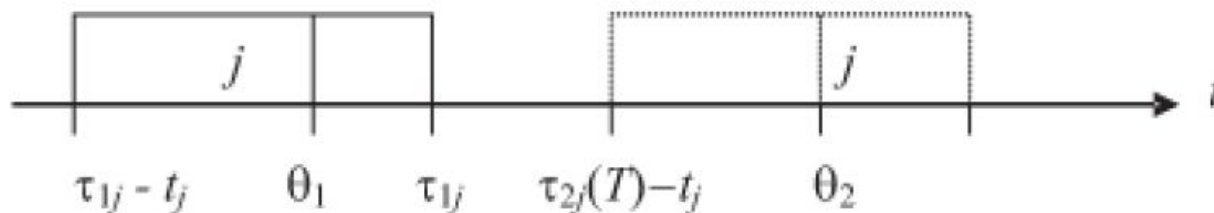
- 2) Значение $\xi(\theta_2 - \tau_{2j}(T) + t_j)$ характеризует аналогичный объём работы j при максимальном смещении времени выполнения работы j вправо.



Условный объём части работы j на отрезке времени

$[\theta_1, \theta_2]$

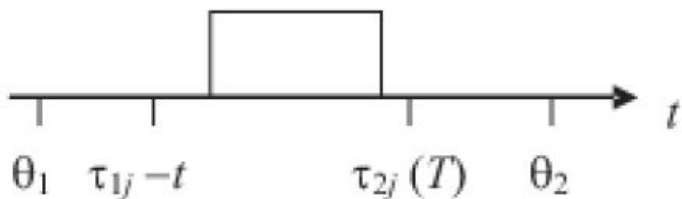
Если для работы j оба указанных выше значения функции ξ отличны от нуля, но не превышают значение t_j и $\theta_2 - \theta_1$, то максимально разгрузить отрезок $[\theta_1, \theta_2]$ от работы j можно смещением времени его выполнения в сторону, обеспечивающую меньшее из двух указанных выше значений ξ



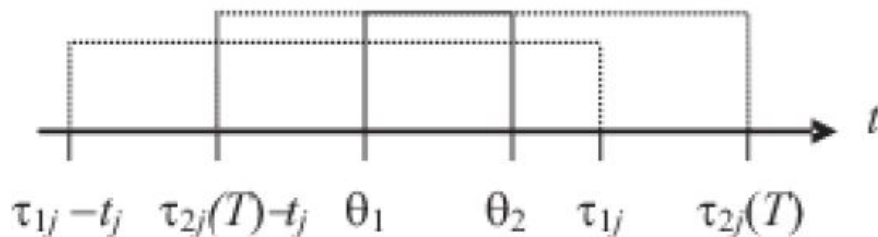
Условный объём части работы j на отрезке времени $[\theta_1, \theta_2]$

Два случая, когда работа j не может быть хотя бы частично смещена с отрезка $[\theta_1, \theta_2]$:

а) $\theta_1 \leq \tau_{1j} - t_j < \tau_{2j}(T) \leq \theta_2$, в этом случае $t_j \leq \theta_2 - \theta_1$ и объём работы j , выполняемой на отрезке, совпадает с объёмом t_j всей этой работы;



б) $\tau_{1j} \geq \theta_2 \wedge \tau_{2j}(T) - t_j \leq \theta_1$, в этом случае очевидно, что $t_j \geq \theta_2 - \theta_1$ и объём части работы j , выполняемой на отрезке $[\theta_1, \theta_2]$ совпадает со значением $\theta_2 - \theta_1$.



Алгоритм нахождения значения функции $\phi^{(T)}(\theta_1, \theta_2)$.

1. Предполагаем, что для каждой работы $j = 1, \dots, m$, известны значения $t_j, \tau_{1j}, \tau_{2j}(T)$. Полагаем равным нулю значение переменной ϕ .
2. Организуем последовательный анализ работ $j = 1, \dots, m$.
3. Для каждой работы j полагаем
$$\phi := \phi + \min \{ \xi (\tau_{1j} - \theta_1), \xi (\theta_2 - \tau_{2j}(T) + t_j), t_j, \theta_2 - \theta_1 \}.$$

После перебора всех работ $\phi = \phi^{(T)}(\theta_1, \theta_2)$.

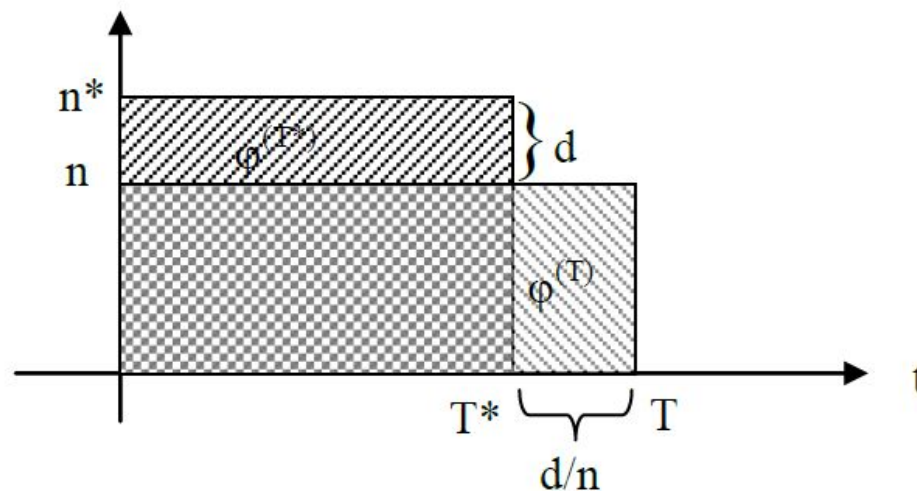
Алгоритм нахождения значения функции $\phi^{(T)}(\theta_1, \theta_2)$.

Теорема 2. Пусть заданный алгоритм выполняется на ВС, состоящей из n^* процессоров, и T^* — текущее значение оценки снизу времени выполнения алгоритма. Пусть на отрезке времени $[\theta_1, \theta_2] \in [0, T^*]$ выполняется соотношение:

$$\phi^{(T^*)}(\theta_1, \theta_2) - n(\theta_2 - \theta_1) = d > 0.$$

Тогда минимальное время T выполнения алгоритма на n процессорах удовлетворяет соотношению:

$$T \geq T^* + \frac{d}{n}$$



Нижняя оценка минимального числа процессоров, необходимого для выполнения алгоритма за заданное время

1. Первоначально полагаем $n = 0$

2. Организуем перебор всех отрезков $[\theta_1, \theta_2]$ $[0, T]$ в порядке

$[0, 1];$

$[0, 2]; [1, 2];$

$[0, 3]; [1, 3]; [2, 3];$

.....

$[0, T]; [1, T]; \dots, [T - 1, T]$

3. Для очередного анализируемого отрезка времени $[\theta_1, \theta_2]$ находим значение

$$n' = \left\lceil \frac{\varphi^{(T^*)}(\theta_1, \theta_2)}{\theta_2 - \theta_1} \right\rceil$$

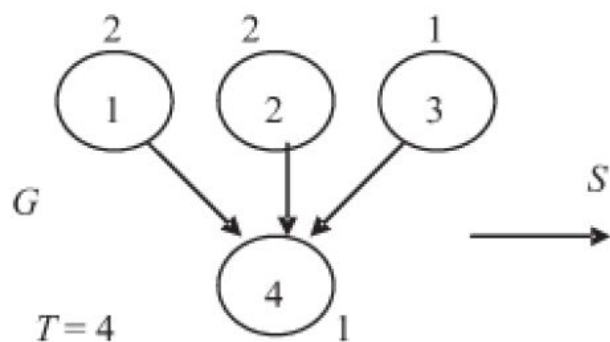
4. Если $n' > n$, выполняем операцию $n := n'$.

После перебора всех отрезков окажется найденным значение n , которое равно максимальному из значений, удовлетворяющих условию:

$$T \geq T^* + \frac{d}{n}$$

Пример

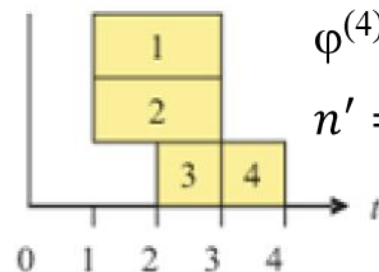
Задача: Найти $\phi^{(4)}(\theta_1, \theta_2)$ и n' .



S

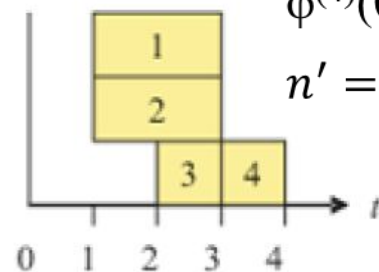
1				2
2				2
3				1
4	1	1	1	1

$\varphi[\theta_1, \theta_2]$	Номер узла j	τ_{1j}	τ_{2j}	$\tau_{1j} - \theta_1$	$\theta_2 - \tau_{2j} + t_j$	t_j	$\theta_2 - \theta_1$	min
$\varphi[0,1]$	1	2	3	2-0	1-3+2	2	1	0
	2	2	3	2-0	1-3+2	2	1	0
	3	1	3	1-0	1-3+1	1	1	0
	4	3	4	3-0	1-4+1	1	1	0
Итого $\varphi^{(4)}[0,1] = \sum min = 0+0+0+0=0$								
$\varphi[0,2]$	1	2	3	2-0	2-3+2	2	2	1
	2	2	3	2-0	2-3+2	2	2	1
	3	1	3	1-0	2-3+1	1	2	0
	4	3	4	3-0	2-4+1	1	2	0
Итого $\varphi^{(4)}[0,2] = \sum min = 1+1+0+0=2$								



$$\varphi^{(4)}(0,1) = 0$$

$$n' = \frac{\varphi^{(4)}(0,1)}{1-0} = 0$$



$$\varphi^{(4)}(0,2) = 2$$

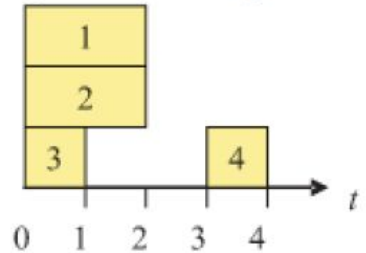
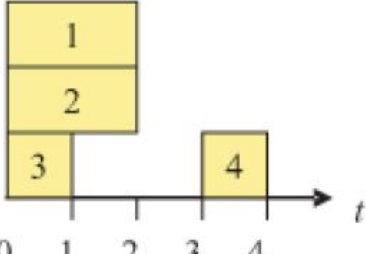
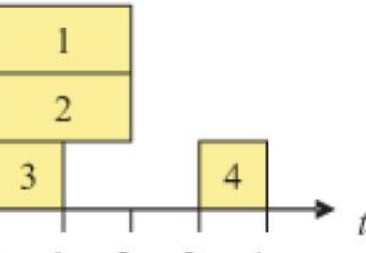
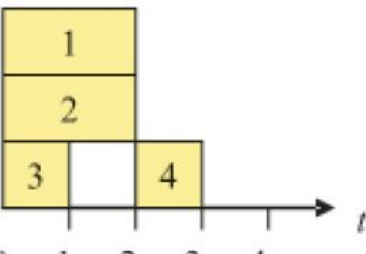
$$n' = \frac{\varphi^{(4)}(0,2)}{2-0} = 1$$

Пример

$\varphi[1,2]$	1	2	3	2-1	2-3+2	2	1	1
	2	2	3	2-1	2-3+2	2	1	1
	3	1	3	1-1	2-3+1	1	1	0
	4	3	4	3-1	2-4+1	1	1	0
	Итого $\varphi^{(4)}[1,2]=\sum \min=1+1+0+0=2$							
$\varphi[0,3]$	1	2	3	2-0	3-3+2	2	3	2
	2	2	3	2-0	3-3+2	2	3	2
	3	1	3	1-0	3-3+1	1	3	1
	4	3	4	3-0	3-4+1	1	3	0
	Итого $\varphi^{(4)}[0,3]=\sum \min=2+2+1+0=5$							
$\varphi[1,3]$	1	2	3	2-1	3-3+2	2	2	1
	2	2	3	2-1	3-3+2	2	2	1
	3	1	3	1-1	3-3+1	1	2	0
	4	3	4	3-1	3-4+1	1	2	0
	Итого $\varphi^{(4)}[1,3]=\sum \min=1+1+0+0=2$							
$\varphi[2,3]$	1	2	3	2-2	3-3+2	2	1	0
	2	2	3	2-2	3-3+2	2	1	0
	3	1	3	1-2	3-3+1	1	1	0
	4	3	4	3-2	3-4+1	1	1	0
	Итого $\varphi^{(4)}[2,3]=\sum \min=0+0+0+0=0$							

	$\varphi^{(4)}(1,2) = 2$ $n' = 2$
	$\varphi^{(4)}(0,3) = 5$ $n' = 2$
	$\varphi^{(4)}(1,3) = 2$ $n' = 1$
	$\varphi^{(4)}(2,3) = 0$ $n' = 0$

Пример

$\varphi[0,4]$	1	2	3	2-0	4-3+2	2	4	2	 $\varphi^{(4)}(0,4) = 6$ $n' = 2$
	2	2	3	2-0	4-3+2	2	4	2	
	3	1	3	1-0	4-3+1	1	4	1	
	4	3	4	3-0	4-4+1	1	4	1	
	Итого $\varphi^{(4)}[0,4] = \sum \min = 2+2+1+1 = 6$								
$\varphi[1,4]$	1	2	3	2-1	4-3+2	2	3	1	 $\varphi^{(4)}(1,4) = 3$ $n' = 1$
	2	2	3	2-1	4-3+2	2	3	1	
	3	1	3	1-1	4-3+1	1	3	0	
	4	3	4	3-1	4-4+1	1	3	1	
	Итого $\varphi^{(4)}[1,4] = \sum \min = 1+1+0+1 = 3$								
$\varphi[2,4]$	1	2	3	2-2	4-3+2	2	2	0	 $\varphi^{(4)}(2,4) = 1$ $n' = 1$
	2	2	3	2-2	4-3+2	2	2	0	
	3	1	3	1-2	4-3+1	1	2	0	
	4	3	4	3-2	4-4+1	1	2	1	
	Итого $\varphi^{(4)}[2,4] = \sum \min = 0+0+0+1 = 1$								
$\varphi[3,4]$	1	2	3	2-3	4-3+2	2	1	0	 $\varphi^{(4)}(3,4) = 0$ $n' = 0$
	2	2	3	2-3	4-3+2	2	1	0	
	3	1	3	1-3	4-3+1	1	1	0	
	4	3	4	3-3	4-4+1	1	1	0	
	Итого $\varphi^{(4)}[3,4] = \sum \min = 0+0+0+0 = 0$								

$$n = \max n' = 2.$$

Нижняя оценка минимального времени выполнения данного алгоритма на ВС

1. Первоначально полагаем:

$$T = \max \left\{ \left\lceil \frac{1}{n} \sum_{j=1}^m t_j \right\rceil, T_{кр} \right\}$$

2. Организуем перебор всех отрезков $[\theta_1, \theta_2] [0, T]$ в той же последовательности, что и в предыдущем алгоритме.

3. Для очередного анализируемого отрезка времени $[\theta_1, \theta_2]$ находим значение:

$$d = \Psi^{(T)}(\theta_1, \theta_2) - n(\theta_2 - \theta_1)$$

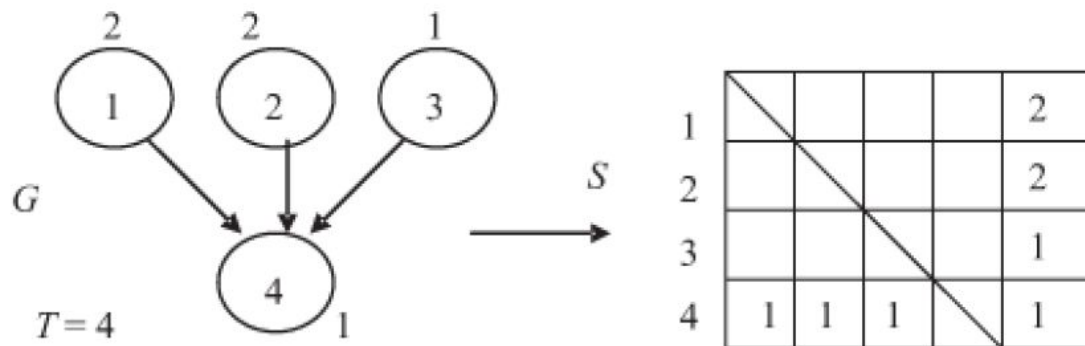
4. Если $d > 0$, выполняем операцию $T := T + \lceil d/n \rceil$.

5. Полагаем $\tau_{2j}(T) := \tau_{2j}(T) + \lceil d/n \rceil$, $j = 1, \dots, m$.

После перебора всех отрезков $[\theta_1, \theta_2]$ окажется найденным окончательное значение T — нижняя оценка минимального времени выполнения данного алгоритма на данной ВС.

Пример

Пусть число процессоров: $n = 2$



Первоначально находим:

$$T = \max \left\{ \frac{1}{2} \cdot (2 + 2 + 1 + 1), T_{\text{кр}} = 3 \right\} = 3$$

$$T_{\text{кр}} = T = 3$$

Находим для $T_{\text{кр}}$ новые поздние сроки:

$$\tau_{21}=2, \tau_{22}=2, \tau_{23}=2, \tau_{24}=3$$

Пример

$\varphi[\theta_1, \theta_2]$	Значение $\varphi[\theta_1, \theta_2]$	$d = \varphi[\theta_1, \theta_2] - n(\theta_2 - \theta_1)$	$T = T + \lfloor d/n \rfloor$	Диаграмма (построены не верно)
$\varphi^{(3)}[0, 1]$	2	$2 - 2(1 - 0) = 0$	3	
$\varphi^{(3)}[0, 2]$	5	$5 - 2(2 - 0) = 1 > 0$	$3 + \lfloor 1/2 \rfloor = 4$	
$\varphi^{(4)}[1, 2]$	2	$2 - 2(2 - 1) = 0$	4	

Пример

$\varphi[\theta_1, \theta_2]$	Значение $\varphi[\theta_1, \theta_2]$	$d = \varphi[\theta_1, \theta_2] - n(\theta_2 - \theta_1)$	$T = T + \lfloor d/n \rfloor$	Диаграмма (построены не верно)
$\varphi^{(4)}[0,3]$	2	$2 - 2(3 - 0) = -4 < 0$	4	
$\varphi^{(4)}[1,3]$	2	$2 - 2(3 - 1) = -1$	4	
$\varphi^{(4)}[2,3]$	0	$0 - 2(3 - 2) = -2$	4	

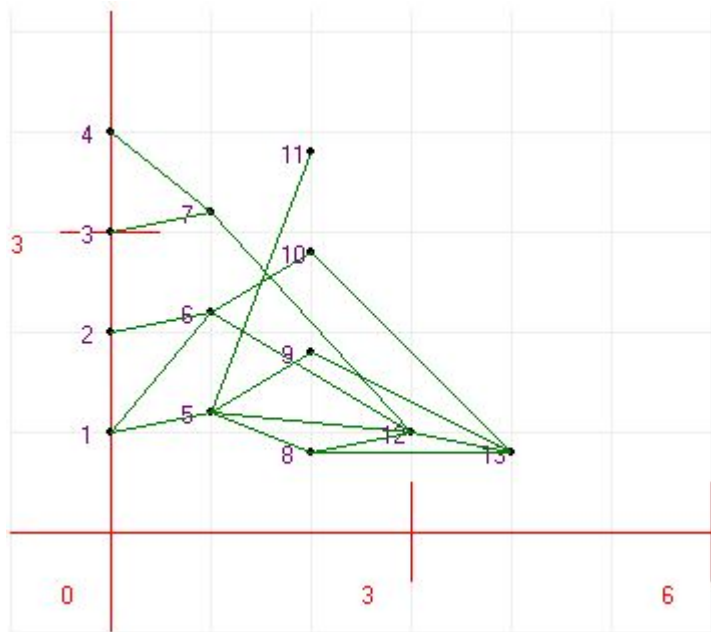
Пример

$\varphi[\theta_1, \theta_2]$	Значение $\varphi[\theta_1, \theta_2]$	$d = \varphi[\theta_1, \theta_2] - n(\theta_2 - \theta_1)$	$T = T + \lfloor d/n \rfloor$	Диаграмма (построены не верно)
$\varphi^{(4)}[0,4]$	6	$6 - 2(4 - 0) = -2$	4	
$\varphi^{(4)}[1,4]$	3	$3 - 2(4 - 1) = -3$	4	
$\varphi^{(4)}[2,4]$	1	$1 - 2(4 - 2) = -3$	4	
$\varphi^{(4)}[3,4]$	2	$2 - 2(4 - 3) = 0$	4	

$T = 4$

Самостоятельная работа

ИТОВАЯ



$t_1=1, t_2=2, t_3=1, t_4=3, t_5=2, t_6=5, t_7=2, t_8=2,$
 $t_9=1, t_{10}=6, t_{11}=1, t_{12}=2, t_{13}=4.$

$$T = T_{\text{кр}} + 2$$

1. Постройте параллельную форму графа;
2. Распределите операции по ярусам (группам);
3. Проведите оптимизацию групп;
4. Найдите ПМВНР;
5. Найдите ранние сроки выполнения работ;
6. Найдите поздние сроки выполнения работ при $T = T_{\text{кр}} + 3$;
7. Постройте F – функцию плотности загрузки ВС при T ;
8. Найдите минимальное число процессоров n' , необходимое для выполнения алгоритма за время $T = T_{\text{кр}}$;
9. Найдите минимальное время выполнения данного алгоритма на ВС с числом процессоров равным: $n=2$: