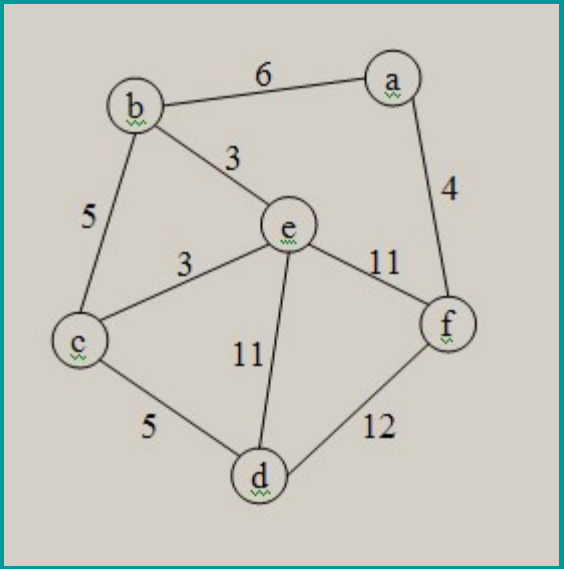
Презентация к лекции по Алгоритмам и структурам данных

Шумков Е.А.

Задача. найти вручную минимальные пути на графе из вершины А с помощью алгоритма Дейкстры



Задача. Сгенерировать одномерный массив случайных целых чисел длиной 30 и найти третий по величине элемент массива с конца

Структура данных «Куча»

В компьютерных науках куча — это специализированная структура данных типа дерево, которая удовлетворяет свойству кучи: если B является узлом-потомком узла A, то ключ(A) ≥ ключ(B). Из этого следует, что элемент с наибольшим ключом всегда является корневым узлом кучи, поэтому иногда такие кучи называют max-кучами (в качестве альтернативы, если сравнение перевернуть, то наименьший элемент будет всегда корневым узлом, такие кучи называют min-кучами). Не существует никаких ограничений относительно того, сколько узлов-потомков имеет каждый узел кучи, хотя на практике их число обычно не более двух. Куча является максимально эффективной реализацией абстрактного типа данных, который называется очередью с приоритетом. Кучи имеют решающее значение в некоторых эффективных алгоритмах на графах, таких, как алгоритм Дейкстры на d-кучах и сортировка методом пирамиды.

*** Кучи обычно реализуются в виде массивов, что исключает наличие указателей между её элементами.**

•Высота кучи определяется как высота двоичного дерева. То есть она равна количеству рёбер в самом длинном простом пути, соединяющем корень кучи с одним из её листьев.

Есть еще понятие куча в динамическом распределении памяти (но это другое…)

Операции над Кучами

- ***найти максимум*** **или** ***найти минимум*****: найти максимальный элемент в max-куче или минимальный элемент в min-куче, соответственно**
- ***удалить максимум*** **или** ***удалить минимум*****: удалить корневой узел в max- или min-куче, соответственно**
- ***увеличить ключ*** **или** ***уменьшить ключ*****: обновить ключ в max- или min-куче, соответственно**
- ***добавить*****: добавление нового ключа в кучу.**
- ***слияние*****: соединение двух куч с целью создания новой кучи, содержащей все элементы обеих исходных.**

Варианты Куч

2-3 куча
Двуродительская куча
Двоичная куча
Биномиальная куча
Очередь Бродала
Фибоначчиева куча
Спаренная куча
… и другие кучи

Применение структуры данных Куча

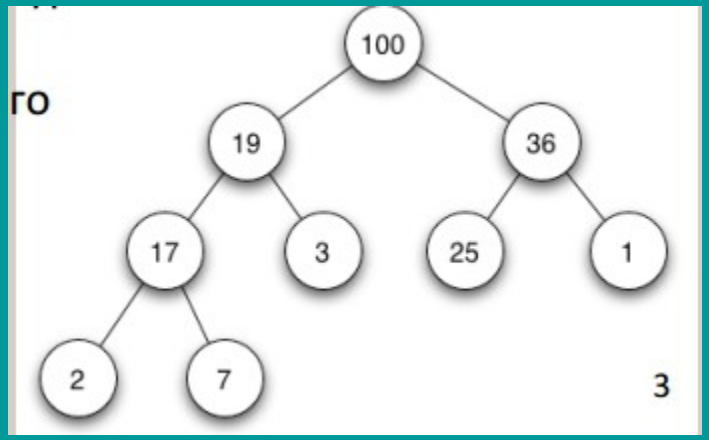
*** Пирамидальная сортировка: один из лучших применяемых методов сортировки, не имеющий квадратичных наихудших сценариев.**
*** Алгоритмы поиска: при использовании кучи поиск минимума, максимума, того и другого, медианы или k-го наибольшего элемента может быть сделан за линейное время (часто даже за константное время).**
***Алгоритмы на графах:**
Применение кучи в качестве структуры данных для внутреннего обхода даёт сокращение времени выполнения на полиномиальный порядок.
Примерами таких проблем являются алгоритм построения минимального остовного дерева Прима и проблема кратчайшего пути Дейкстры.

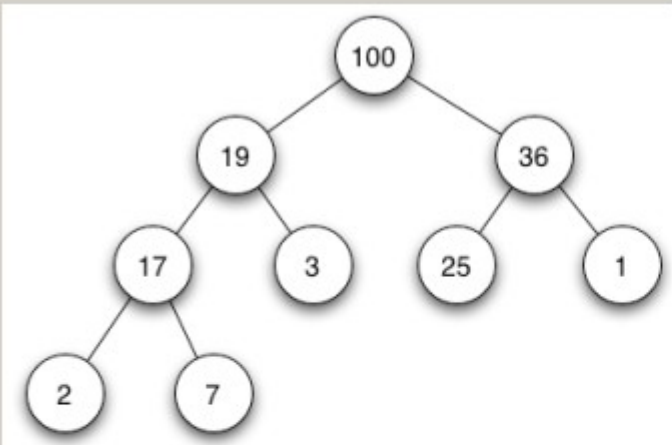
Полная и почти полная бинарная куча может быть представлена очень эффективным способом с помощью индексного массива. Первый (или последний) элемент будет содержать корень. Следующие два элемента массива содержат узлы-потомки корня. Следующие четыре элемента содержат четверых потомков от двух узлов — потомков корня, и т. д. Таким образом, потомки узла уровня n будут располагаться на позициях 2n и 2n+1 для массива индексируемого с единицы. (2n+1 и 2n+2 если с нуля)

Структура данных «Двоичная куча»

Двоичная куча (или сортирующее дерево) – это такое двоичное дерево, для которого Выполнены три условия:

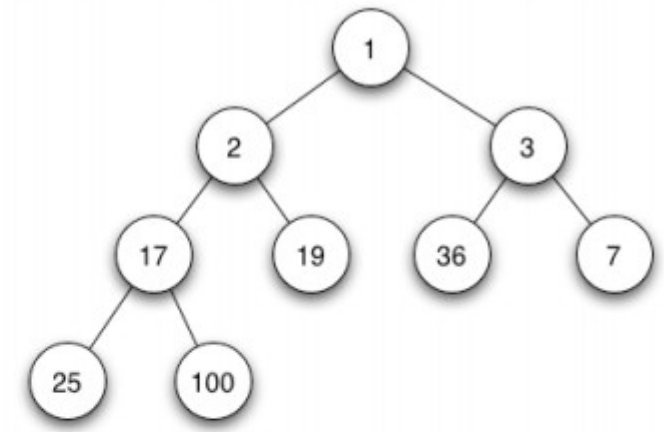
1) **Значение в любой вершине не меньше, чем значения ее потомков**
2) **Глубина всех листьев (расстояние до корня) отличается не более чем на 1 слой**
3) **Последний слой заполняется слева направо без «дырок»**





max-heap

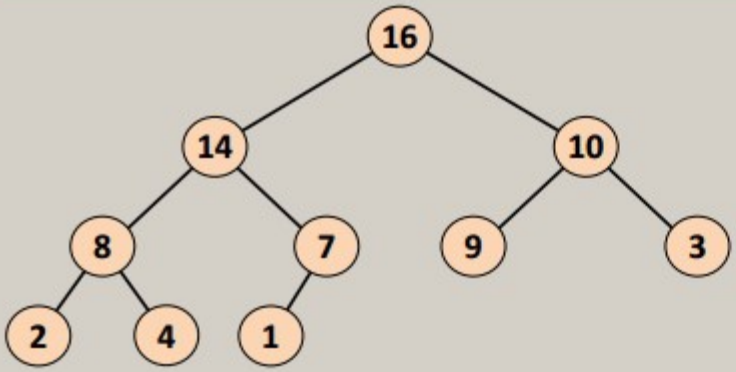
Приоритет любой вершины **не меньше (≥)**, приоритета потомков



min-heap

Приоритет любой вершины **не больше (≤)**, приоритета потомков

Реализация бинарной кучи на основе массива



max-heap (10 элементов)

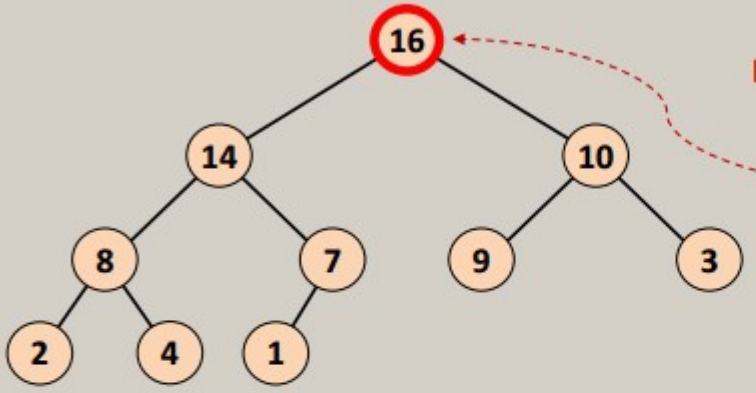
Массив *H*[1..14] приоритетов (ключей):

16	14	10	8	7	9	3	2	4	1				

- Корень дерева храниться в ячейке $H[1]$ – максимальный элемент
- Индекс родителя узла *i*: $Parent(i) = \lfloor i/2 \rfloor$
- Индекс левого дочернего узла: $Left(i) = 2i$
- Индекс правого дочернего узла: $Right(i) = 2i + 1$

H[Parent(i)]>=H[i]

Поиск максимального элемента



max-heap (10 элементов)

Максимальный элемент хранится в корне max-heap

Массив *H*[1..14] приоритетов (ключей):

16	14	10	8	7	9	3	2	4	1				

- Корень дерева храниться в ячейке $H[1]$ – максимальный элемент
- Индекс родителя узла *i*: $Parent(i) = \lfloor i/2 \rfloor$
- Индекс левого дочернего узла: $Left(i) = 2i$
- Индекс правого дочернего узла: $Right(i) = 2i + 1$

Восстановление свойств Кучи

Если в куче изменяется один из элементов, то она может перестать удовлетворять свойству упорядоченности. Для восстановления этого свойства служит процедура Heapify. Она восстанавливает свойство кучи в дереве, у которого левое и правое поддеревья удовлетворяют ему. Эта процедура принимает на вход массив элементов *A* и индекс *i*. Она восстанавливает свойство упорядоченности во всём поддереве, корнем которого является элемент *A*[*i*].

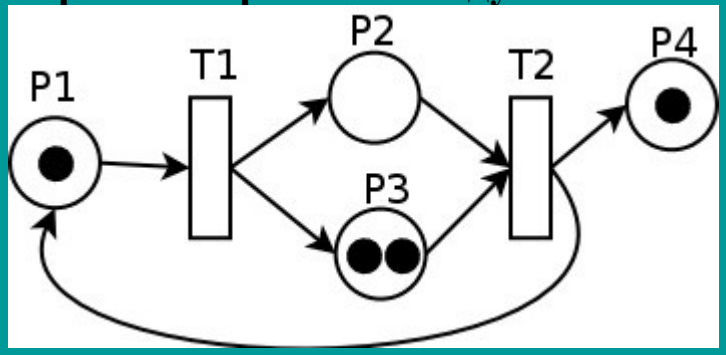
```
Heapify(A, i)
  left ← 2i
  right ← 2i+1
  heap_size - количество элементов в куче
  largest ← i
  if left ≤ A.heap_size и A[left] > A[largest]
    then largest ← left
  if right ≤ A.heap_size и A[right] > A[largest]
    then largest ← right
  if largest ≠ i
    then Обменять A[i] ↔ A[largest]
         Heapify(A, largest)
```

WIKIPEDIA

Использованы материалы презентации доцента Курносова М.Г.

Сети Петри

Сети Петри — математический аппарат для моделирования динамических дискретных систем. Впервые описаны Карлом Петри в 1962 году.



Сеть Петри представляет собой двудольный ориентированный мультиграф, состоящий из вершин двух типов — позиций и переходов, соединённых между собой дугами. Вершины одного типа не могут быть соединены непосредственно. В позициях могут размещаться метки (маркеры, фишки), способные перемещаться по сети.

Событием называют срабатывание перехода, при котором метки из входных позиций этого перехода перемещаются в выходные позиции. События происходят мгновенно либо разновременно, при выполнении некоторых условий.

Как и стандартные UML диаграммы, BPMN и EPC, сети Петри предоставляют возможность графически иллюстрировать процессы включающие выбор, итерации и одновременное выполнение. Но в отличие от данных стандартов, у сетей Петри четкая математическая формулировка и за ними стоит развитая математическая теория.

Сеть Петри есть мультиграф, так как он допускает существование кратных дуг от одной вершины графа к другой. Так как дуги являются направленными, то это ориентированный мультиграф. Вершины графа можно разделить на два множества (позиции и переходы) таким образом, что каждая дуга будет направлена от элемента одного множества (позиций или переходов) к элементу другого множества (переходов или позиций); следовательно, такой граф является двудольным ориентированным мультиграфом. || Википедия.ру

Виды сетей Петри:

Временна́я сеть Петри — переходы обладают весом, определяющим продолжительность срабатывания (задержку).

Стохастическая сеть Петри — задержки являются случайными величинами.

Функциональная сеть Петри — задержки определяются как функции некоторых аргументов, например, количества меток в каких-либо позициях, состояния некоторых переходов.

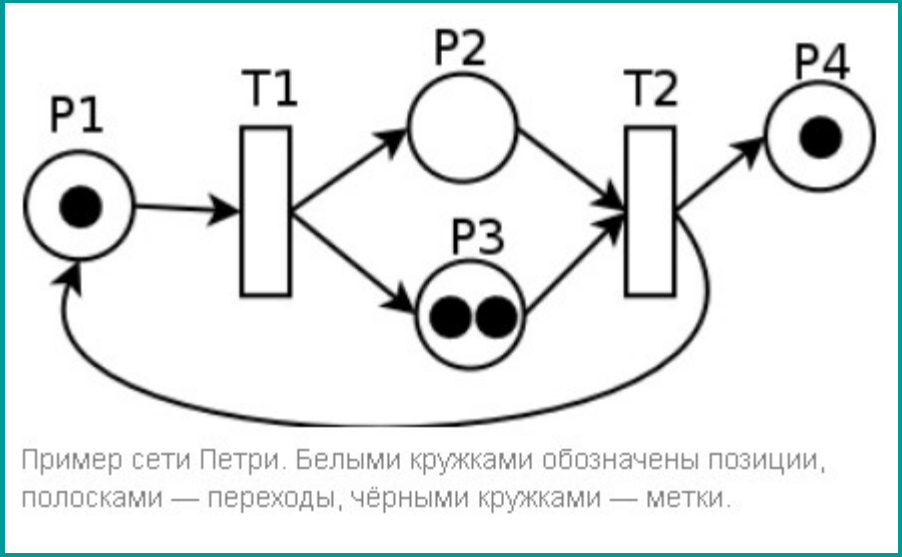
Цветная сеть Петри — метки могут быть различных типов, обозначаемых цветами, тип метки может быть использован как аргумент в функциональных сетях.

Ингибиторная сеть Петри — возможны ингибиторные дуги, запрещающие срабатывания перехода, если во входной позиции, связанной с переходом ингибиторной дугой, находится метка.

Иерархическая сеть — содержит не мгновенные переходы, в которые вложены другие, возможно, также иерархические, сети. Срабатывание такого перехода характеризует выполнение полного жизненного цикла вложенной сети.

Сеть Петри состоит из 4-х элементов:
-Множество позиций P;
-Множество переходов T;
-Входная функция I;
-Выходная функция O;

Структура сети Петри определяется ее позициями, переходами, входной и выходной функцией



Пример сети Петри. Белыми кружками обозначены позиции, полосками — переходы, чёрными кружками — метки.

https://itmodeling.fandom.com

Выполнением сети Петри управляют количество и распределение фишек в сети. Сеть Петри выполняется посредством запусков переходов. Переход запускается удалением фишек из его входных позиций и образованием новых фишек, помещаемых в его выходные позиции. Переход запускается, если он разрешен. Переход называется разрешенным, если каждая из его входных позиций имеет число фишек по крайней мере равное числу дуг из позиции в переход. Фишки во входной позиции, которые разрешают переход, называются его разрешающими фишками. Например, если позиции p1 и p2 служат входами для перехода t1, тогда t1 разрешен, если p1 и p2 имеют хотя бы по одной фишке. Для перехода t3 с входным комплектом {p3,p3,p3} позиция p3 должна иметь не менее 3 фишек для разрешения перехода t3.

Алгоритм работы сети Петри

>Переход t3 I(t3) = {p2} и O(t3) = {p3,p4} разрешен каждый раз, когда в p2 будет хотя бы одна фишка. Переход t3 запускается удалением одной фишки из позиции p2 и помещением одной фишки в позицию p3 и p4 (его выходы). Переход t4, в котором I(t4) = {p4,p5} и O(t4) = {p5,p6,p6} запускается удалением по одной фишке из позиций p4 и p5, при этом одна фишка помещается в p5 и две в p6 (рис. 2).

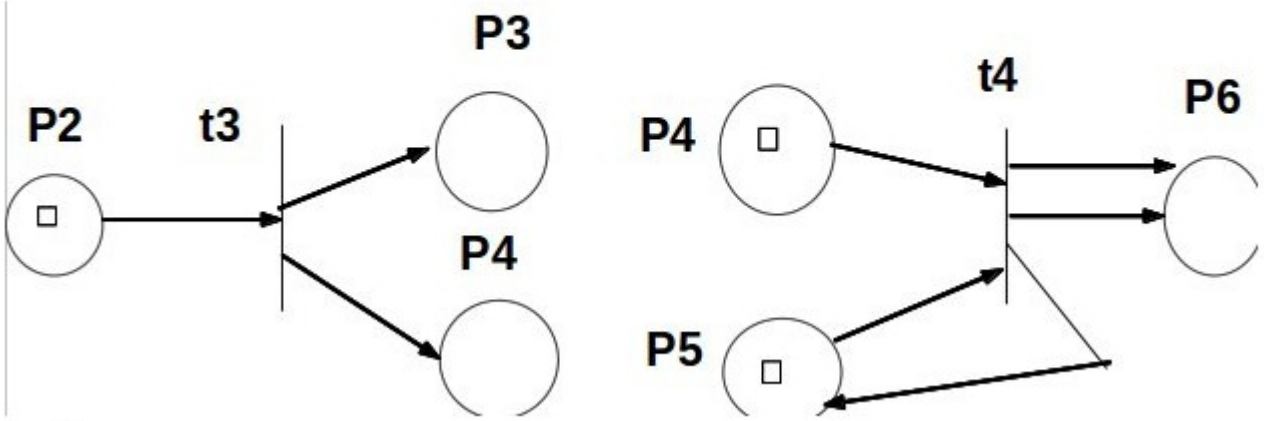


Рис. 2

https://itmodeling.fandom.com

Важное применение – технологические линии, в т.ч. РТК

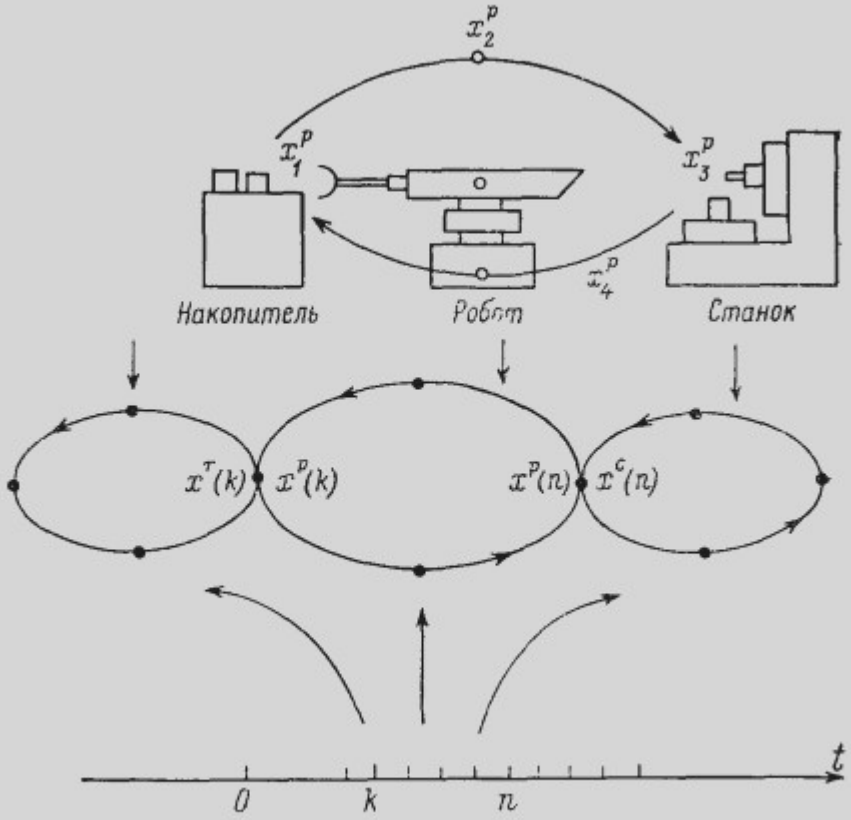
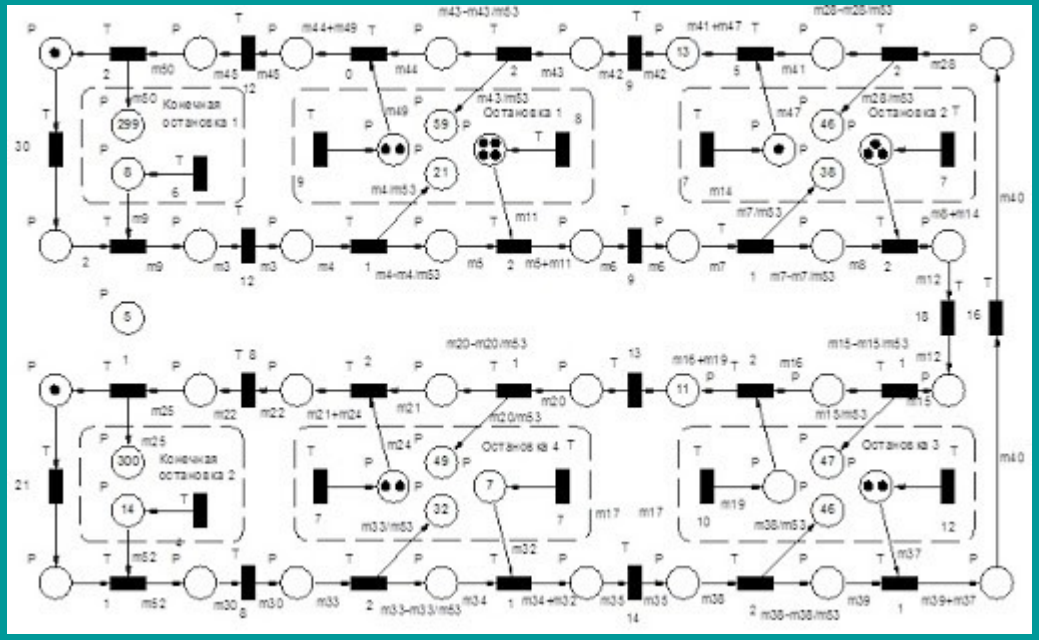
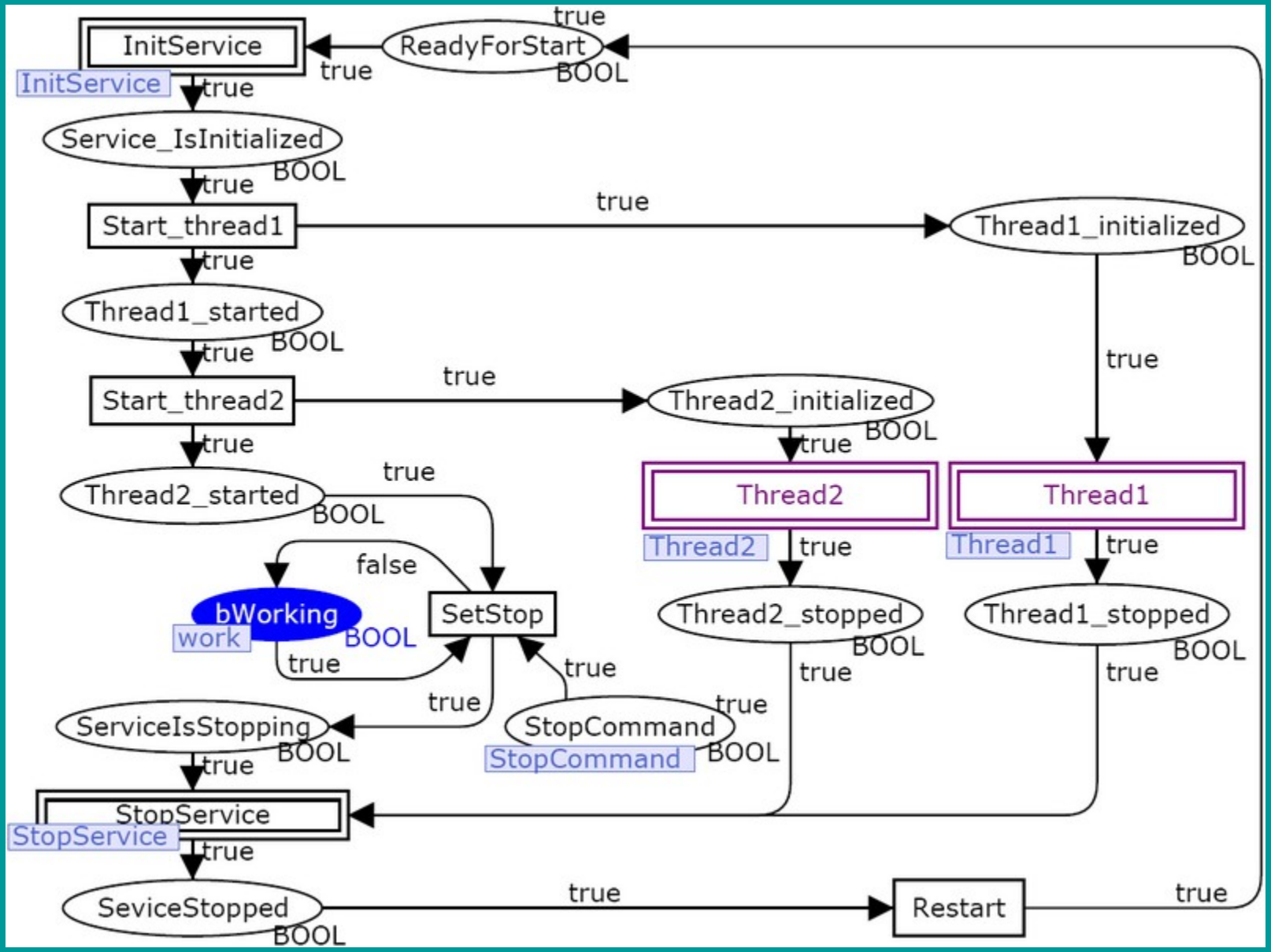


Рис. 1.9. Циклические траектории элементов прозводственной системы.

Пример применения сети Петри – моделирование движения автобусов



Пример применения сети Петри – моделирование параллельного программирования



Литература по сетям Петри:

-***Котов В. Е.*** **Сети Петри. — М: Наука, 1984. — 160 с.**

-***Питерсон Дж.*** **Теория сетей Петри и моделирование систем. — М: Мир, 1984. — 264 с.**