

4. Доминаторы и постдоминаторы

4.1 Доминаторы

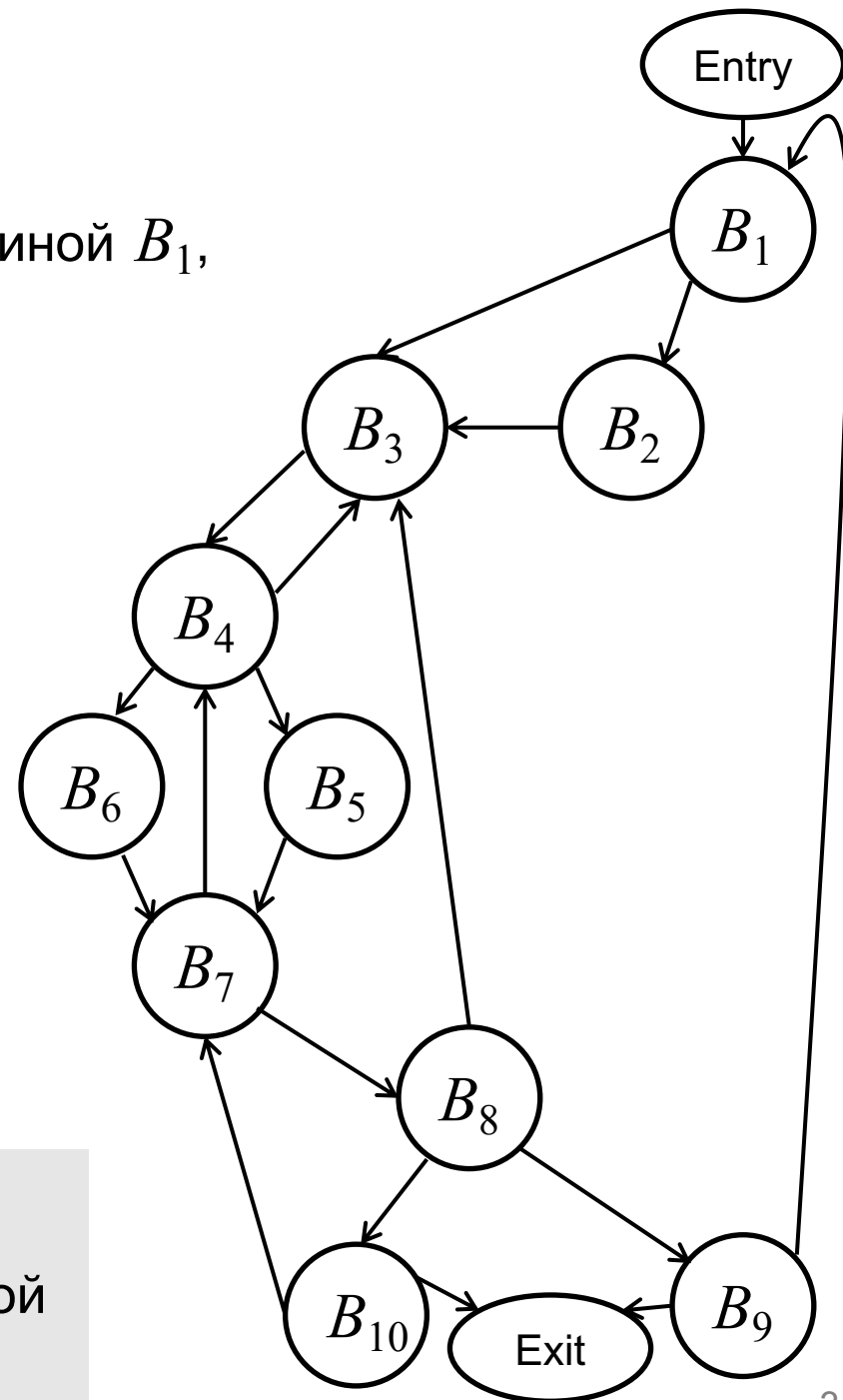
4.1.1 Определение

- ◇ В ГПУ вершина d является *доминатором* вершины n (этот факт записывается как $d \text{ dom } n$ или $d \in \text{Dom}(n)$), если любой путь от вершины Entry до вершины n проходит через вершину d .
- ◇ **Замечание.** Из определения 4.2.1 следует, что каждая вершина n является доминатором самой себя, так как путь от Entry до n проходит через n .

4.1 Доминаторы

4.1.2 Примеры доминаторов

- ◇ Рассмотрим ГПУ с входной вершиной B_1 , показанный на рисунке.
- ◇ B_1 является доминатором всех узлов, включая себя самого.
- ◇ B_2 является доминатором только себя самого.
- ◇ B_3 является доминатором всех вершин, кроме B_1 и B_2 .
- ◇ B_4 является доминатором всех вершин, кроме B_1 , B_2 и B_3 .
- ◇ B_5 и B_6 являются доминаторами только себя.

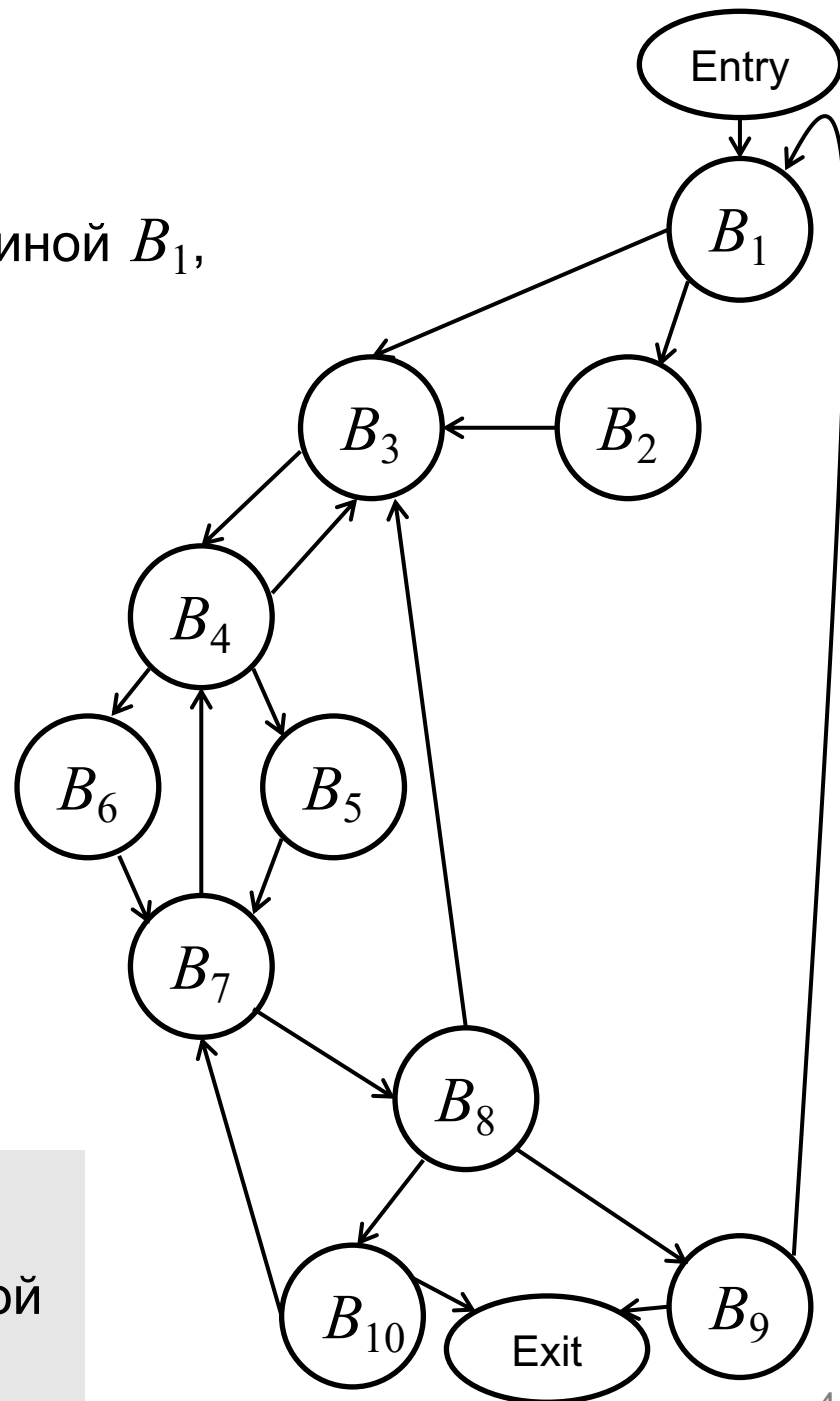


(Определение. Вершина d является доминатором вершины n , если любой путь от $Entry$ до n проходит через d .)

4.1 Доминаторы

4.1.2 Примеры доминаторов

- ◇ Рассмотрим ГПУ с входной вершиной B_1 , показанный на рисунке.
- ◇ B_7 является доминатором вершин B_7, B_8, B_9 и B_{10} .
- ◇ B_8 является доминатором вершин B_8, B_9 и B_{10} .
- ◇ B_9 и B_{10} являются доминаторами только себя.



(Определение. Вершина d является доминатором вершины n , если любой путь от $Entry$ до n проходит через d .)

4.1 Доминаторы

4.1.3 Свойства отношения *dom*

- ◇ 1. Отношение *dom* рефлексивно, антисимметрично и транзитивно, т.е. является отношением частичного порядка.
- (1) *Рефлексивность*: $a \text{ dom } a$.
- (2) *Антисимметричность*: если $a \text{ dom } b$ и $b \text{ dom } a$,
то $a = b$.
- (3) *Транзитивность*: если $a \text{ dom } b$ и $b \text{ dom } c$, то $a \text{ dom } c$.
- ◇ 2. Для любой вершины n ГПУ каждый ациклический путь от *Entry* до n проходит через все доминаторы n , причем на всех таких путях доминаторы проходятся в *одном и том же порядке*

4.1 Доминаторы

4.1.3 Свойства отношения *dom*

- ◇ 3. Вершина s ГПУ является *строгим доминатором* вершины n ($s \text{ sdom } n$), если $s \text{ dom } n$ и $s \neq n$.
- ◇ 4. Вершина i ГПУ является *непосредственным доминатором* вершины n ($i \text{ idom } n$), если
 - (1) $i \text{ sdom } n$
 - (2) не существует вершины m , такой, что $i \text{ sdom } m$ и $m \text{ sdom } n$.
- ◇ 5. У каждой вершины n , за исключением *Entry*, существует единственный непосредственный доминатор.

4.1 Доминаторы

4.1.3 Свойства отношения dom

- ◇ 6. Пусть n, d – вершины ГПУ,
 $Pred(n) = \{p_1, p_2, \dots, p_k\}$ и $d \neq n$.
Тогда $d dom n$ тогда и только тогда, когда $\forall i: d dom p_i$.
- ◇ 7. Множество строгих доминаторов вершины n является пересечением множеств доминаторов всех ее предшественников.

4.1 Доминаторы

4.1.4 Алгоритм вычисления доминаторов

- ◇ **Задача поиска всех доминаторов** вершин ГПУ формулируется как задача анализа потока данных в прямом направлении.

Значением потока данных на входе в блок B является множество вершин (базовых блоков), являющихся доминаторами B .

Операцией сбора является операция пересечения множеств.

Передаточная функция f_B добавляет вершину B к рассматриваемому множеству вершин.

Граничное условие: единственным доминатором вершины $Entry$ является она сама.

4.1 Доминаторы

4.1.4 Алгоритм вычисления доминаторов

◇ Алгоритм

Вход: граф потока $G = \langle N, E \rangle$ с входным узлом $Entry$.

Выход: для каждой вершины $n \in N$ множество $D(n)$ ее доминаторов.

Метод: найти решение следующей задачи потока данных (вершины n соответствуют базовым блокам):

$$\forall n \in N \quad D(n) = Out[Pred(n)].$$

4.1 Доминаторы

4.1.4 Алгоритм вычисления доминаторов

Область определения	Множество подмножеств базовых блоков
Направление обхода	<i>Forward</i>
Передаточная функция	$f_B(x) = x \cup \{B\}$
Граничное условие	$Out [Entry] = Entry$
Операция сбора (\wedge)	\cap
Система уравнений	$Out[B] = f_B (In[B])$ $In[B] = \bigcap_{P \in Pred(B)} Out[P]$
Начальное приближение	$Out [B] = N$

4.1 Доминаторы

4.1.4 Замечания о сходимости алгоритма вычисления доминаторов

- ГПУ является приводимым, если применяя два следующих шаблона преобразований, можно «свернуть» исходный ГПУ к единственной вершине:
 - У вершины-цикла с «дугой-петлёй», ведущей в саму эту вершину, «дуга-петля» удаляется;
 - Две вершины a и b , такие, что у b имеется единственный предок – вершина a , заменяются на a . Вершина b удаляется, а её потомки становятся потомками a .
- По сути, приводимый ГПУ соответствует хорошо структурированной программе, в которой все переходы могут быть реализованы с помощью операторов `if`, `while/for`, `do-while` и `break` (нет переходов внутрь циклов и ветвлений).
- Большинство программ имеют приводимый ГПУ (возможно, кроме отдельных функций, где имеются веские причины использовать такие переходы, например, при реализации интерпретатора).
- Естественные циклы и приводимость ГПУ более подробно рассматриваются в следующих лекциях, здесь они упоминаются только для оценки сходимости итеративного алгоритма поиска доминаторов.

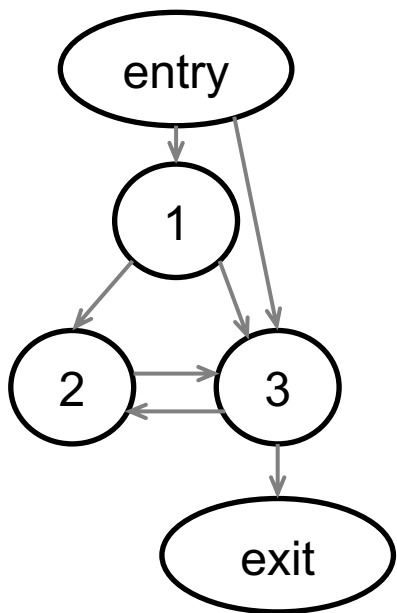
4.1 Доминаторы

4.1.4 Замечания о сходимости алгоритма вычисления доминаторов

- Если входной ГПУ приводим, корректное множество доминаторов для каждого узла получается уже после 1-й итерации алгоритма при посещении узлов в глубину в порядке рассмотренной нумерации «на обратном ходе».
- Это следует из того, что в приводимый граф состоит из естественных циклов, а заголовок цикла (узел, в который входит обратная дуга) доминирует все узлы цикла. Т.е. все обратно направленные дуги в программе по нумерации (которые идут от вершины с большим номером к меньшему) – это обратные дуги циклов. Для обратных дуг на 2-й итерации мы могли бы получить изменение во множестве доминаторов для заголовка цикла, если бы *latch*-узел (тот, из которого исходит обратная дуга) не имел узла-заголовка в множестве своих доминаторов. Но заголовок цикла доминирует все узлы цикла, включая и *latch*-узел. Так что пересечения с полным множеством на 1-й итерации будет достаточно, потому что всех предков, кроме доступных по обратным дугам, мы уже обошли раньше заголовка на 1-й же итерации, а предков, доступных по обратной дуге, заголовок цикла доминирует, и они не "уменьшат" результат от пересечения остальных предков.

4.1 Доминаторы

4.1.4 Сходимость алгоритма вычисления доминаторов



n	D ⁰ (n)	D ¹ (n)	D ² (n)
Entry	Entry	Entry	Entry
1	N	1, Entry	1, Entry
2	N	2, 1, Entry	2, Entry
3	N	3, Entry	3, Entry

Пример неприводимого, но «правильно» (по нашему алгоритму) пронумерованного графа, на котором не удастся вычислить доминаторы за один обход в порядке нумерации: на 1-й итерации при вычислении $D(2)$, когда делали пересечение по всем его предкам, мы считали, что $D(3) = N$, и поэтому получили $D(2) = \{2\} \cup (\{Entry, 1\} \cap N) = \{Entry, 1, 2\}$. Только после этого мы посетили вершину 3, рассмотрели в качестве ее предка дугу из Entry в 3, взяли пересечение с $D(Entry) = \{Entry\}$, после чего $D(3) = \{Entry, 3\}$. Это была последняя вершина на этой итерации. Теперь, мы должны учесть этот новый $D(3)$ в качестве предка 2 при вычислении $D(2)$, и для этого нужна еще одна итерация.

4.1 Доминаторы

4.1.4 Алгоритм вычисления доминаторов

◇ **Пример.** Применим алгоритм к ГПУ на рисунке в предположении, что блоки посещаются в порядке их номеров.

◇ **Первая итерация**

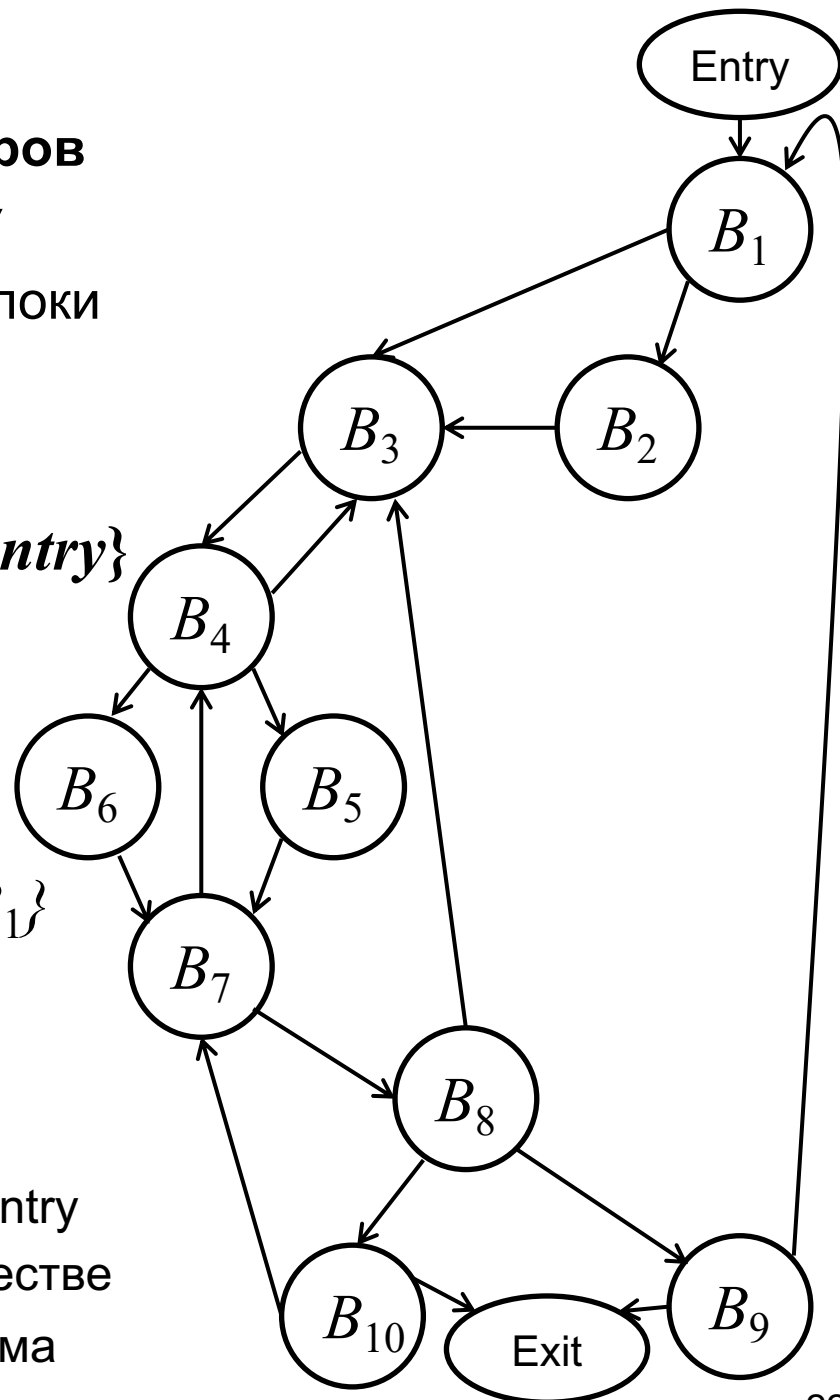
Граничное условие: $D(Entry) = \{Entry\}$

$Pred(B_1) = \{Entry, B_9\}$

$D(B_1) = \{B_1\} \cup (D(Entry) \cap$
 $\cap D(B_9) =$

$= \{B_1\} \cup (\{Entry\} \cap N) = \{Entry, B_1\}$

Замечание. Далее в этом примере для краткости мы не будем включать Entry в множества $D(B_i)$, однако в дальнейшем Entry будет нужно в дереве доминаторов в качестве $Idom(B_1)$ для корректной работы алгоритма построения границы доминирования.



4.1 Доминаторы

4.1.4 Алгоритм вычисления доминаторов

◇ **Пример.** Применим алгоритм к ГПУ на рисунке в предположении, что блоки посещаются в порядке их номеров.

◇ **Первая итерация**

$$Pred(B_2) = \{B_1\}$$

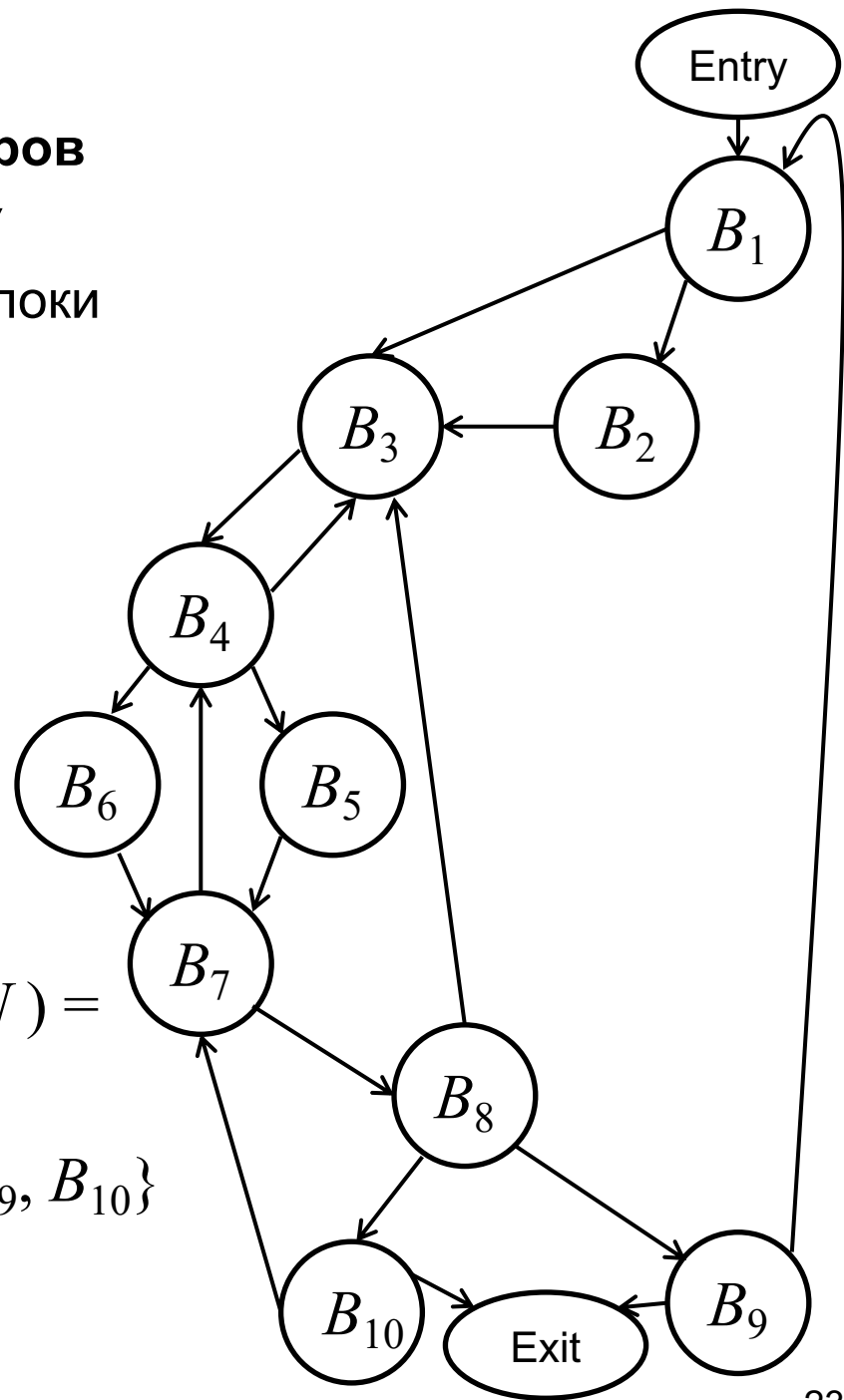
$$D(B_2) = \{B_2\} \cup D(B_1) = \{B_1, B_2\}$$

$$Pred(B_3) = \{B_1, B_2, B_4, B_8\}$$

$$D(B_3) = \{B_3\} \cup (D(B_1) \cap D(B_2) \cap D(B_4) \cap D(B_8)) =$$

$$= \{B_3\} \cup (\{B_1\} \cap \{B_1, B_2\} \cap N \cap N) = \{B_1, B_3\}$$

$$N = \{B_1, B_2, B_3, B_4, B_5, B_6, B_7, B_8, B_9, B_{10}\}$$



4.1 Доминаторы

4.1.4 Алгоритм вычисления доминаторов

◇ Первая итерация (окончание)

$$Pred(B_4) = \{B_3, B_7\}$$

$$\begin{aligned} D(B_4) &= \{B_4\} \cup (D(B_3) \cap D(B_7)) = \{B_4\} \cup (\{B_1, B_3\} \cap N) = \\ &= \{B_1, B_3, B_4\} \end{aligned}$$

4.1 Доминаторы

4.1.4 Алгоритм вычисления доминаторов

◇ Первая итерация (окончание)

$$Pred(B_4) = \{B_3, B_7\}$$

$$\begin{aligned} D(B_4) &= \{B_4\} \cup (D(B_3) \cap D(B_7)) = \{B_4\} \cup (\{B_1, B_3\} \cap N) = \\ &= \{B_1, B_3, B_4\} \end{aligned}$$

$$Pred(B_5) = Pred(B_6) = \{B_4\}$$

$$D(B_5) = \{B_5\} \cup D(B_4) = \{B_5\} \cup \{B_1, B_3, B_4\} = \{B_1, B_3, B_4, B_5\}$$

$$D(B_6) = \{B_6\} \cup D(B_4) = \{B_6\} \cup \{B_1, B_3, B_4\} = \{B_1, B_3, B_4, B_6\}$$

4.1 Доминаторы

4.1.4 Алгоритм вычисления доминаторов

◇ Первая итерация (окончание)

$$Pred(B_4) = \{B_3, B_7\}$$

$$\begin{aligned} D(B_4) &= \{B_4\} \cup (D(B_3) \cap D(B_7)) = \{B_4\} \cup (\{B_1, B_3\} \cap N) = \\ &= \{B_1, B_3, B_4\} \end{aligned}$$

$$Pred(B_5) = Pred(B_6) = \{B_4\}$$

$$D(B_5) = \{B_5\} \cup D(B_4) = \{B_5\} \cup \{B_1, B_3, B_4\} = \{B_1, B_3, B_4, B_5\}$$

$$D(B_6) = \{B_6\} \cup D(B_4) = \{B_6\} \cup \{B_1, B_3, B_4\} = \{B_1, B_3, B_4, B_6\}$$

$$Pred(B_7) = \{B_5, B_6, B_{10}\}$$

$$\begin{aligned} D(B_7) &= \{B_7\} \cup (D(B_5) \cap D(B_6)) \cap D(B_{10}) = \{B_7\} \cup (\{B_1, B_3, B_4, B_5\} \cap \\ &= \{B_1, B_3, B_4, B_6\} \cap N) = \{B_1, B_3, B_4, B_7\} \end{aligned}$$

$$Pred(B_8) = \{B_7\}$$

4.1 Доминаторы

4.1.4 Алгоритм вычисления доминаторов

◇ Первая итерация (окончание)

$$Pred(B_4) = \{B_3, B_7\}$$

$$D(B_4) = \{B_4\} \cup (D(B_3) \cap D(B_7)) = \{B_4\} \cup (\{B_1, B_3\} \cap N) = \\ = \{B_1, B_3, B_4\}$$

$$Pred(B_5) = Pred(B_6) = \{B_4\}$$

$$D(B_5) = \{B_5\} \cup D(B_4) = \{B_5\} \cup \{B_1, B_3, B_4\} = \{B_1, B_3, B_4, B_5\}$$

$$D(B_6) = \{B_6\} \cup D(B_4) = \{B_6\} \cup \{B_1, B_3, B_4\} = \{B_1, B_3, B_4, B_6\}$$

$$Pred(B_7) = \{B_5, B_6, B_{10}\}$$

$$D(B_7) = \{B_7\} \cup (D(B_5) \cap D(B_6)) \cap D(B_{10}) = \{B_7\} \cup (\{B_1, B_3, B_4, B_5\} \cap \\ = \{B_1, B_3, B_4, B_6\} \cap N) = \{B_1, B_3, B_4, B_7\}$$

$$Pred(B_8) = \{B_7\}$$

$$D(B_8) = \{B_8\} \cup D(B_7) = \{B_8\} \cup \{B_1, B_3, B_4, B_7\} = \{B_1, B_3, B_4, B_7, B_8\}$$

$$Pred(B_9) = Pred(B_{10}) = \{B_8\}$$

$$D(B_9) = \{B_9\} \cup D(B_8) = \{B_9\} \cup \{B_1, B_3, B_4, B_7, B_8\} = \{B_1, B_3, B_4, B_7, B_8, B_9\}$$

$$D(B_{10}) = \{B_{10}\} \cup D(B_8) = \{B_{10}\} \cup \{B_1, B_3, B_4, B_7, B_8\} = \\ = \{B_1, B_3, B_4, B_7, B_8, B_{10}\}$$

4.1 Доминаторы

4.1.4 Алгоритм вычисления доминаторов

◇ Первая итерация (окончание)

$$Pred(B_4) = \{B_3, B_7\}$$

$$D(B_4) = \{B_4\} \cup (D(B_3) \cap D(B_7)) = \{B_4\} \cup (\{B_1, B_3\} \cap N) = \\ = \{B_1, B_3, B_4\}$$

$$Pred(B_5) = Pred(B_6) = \{B_4\}$$

$$D(B_5) = \{B_5\} \cup D(B_4) = \{B_5\} \cup \{B_1, B_3, B_4\} = \{B_1, B_3, B_4, B_5\}$$

$$D(B_6) = \{B_6\} \cup D(B_4) = \{B_6\} \cup \{B_1, B_3, B_4\} = \{B_1, B_3, B_4, B_6\}$$

$$Pred(B_7) = \{B_5, B_6\}$$

$$D(B_7) = \{B_7\} \cup (D(B_5) \cap D(B_6)) = \{B_7\} \cup (\{B_1, B_3, B_4, B_5\} \cap \\ = \{B_1, B_3, B_4, B_6\} \cap N) = \{B_1, B_3, B_4, B_7\}$$

$$Pred(B_8) = \{B_7\}$$

$$D(B_8) = \{B_8\} \cup D(B_7) = \{B_8\} \cup \{B_1, B_3, B_4, B_7\} = \{B_1, B_3, B_4, B_7, B_8\}$$

$$Pred(B_9) = Pred(B_{10}) = \{B_8\}$$

$$D(B_9) = \{B_9\} \cup D(B_8) = \{B_9\} \cup \{B_1, B_3, B_4, B_7, B_8\} = \{B_1, B_3, B_4, B_7, B_8, B_9\}$$

$$D(B_{10}) = \{B_{10}\} \cup D(B_8) = \{B_{10}\} \cup \{B_1, B_3, B_4, B_7, B_8\} = \\ = \{B_1, B_3, B_4, B_7, B_8, B_{10}\}$$

**Полученные значения
 $D(B_1) - D(B_{10})$ на второй
итерации не изменяются**

4.1 Доминаторы

4.1.5 Непосредственные доминаторы и дерево доминаторов

n	$D(n)$	$IDom(n)$
B_1	B_1	—
B_2	B_1, B_2	B_1
B_3	B_1, B_3	B_1
B_4	B_1, B_3, B_4	B_3
B_5	B_1, B_3, B_4, B_5	B_4
B_6	B_1, B_3, B_4, B_6	B_4
B_7	B_1, B_3, B_4, B_7	B_4
B_8	B_1, B_3, B_4, B_7, B_8	B_7
B_9	$B_1, B_3, B_4, B_7, B_8, B_9$	B_8
B_{10}	$B_1, B_3, B_4, B_7, B_8, B_{10}$	B_8

В таблице приведены списки доминаторов каждой вершины ГПУ из рассмотренного примера.

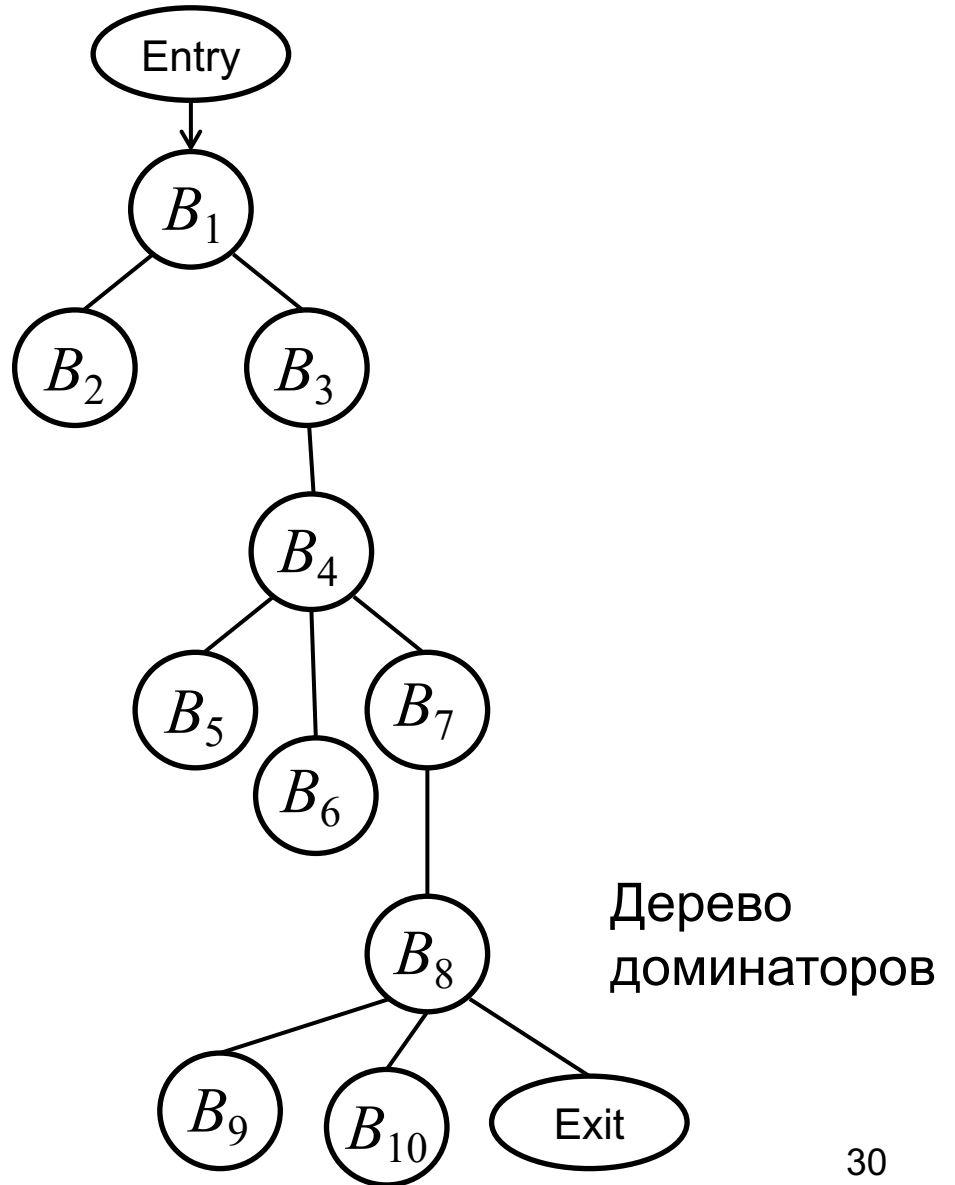
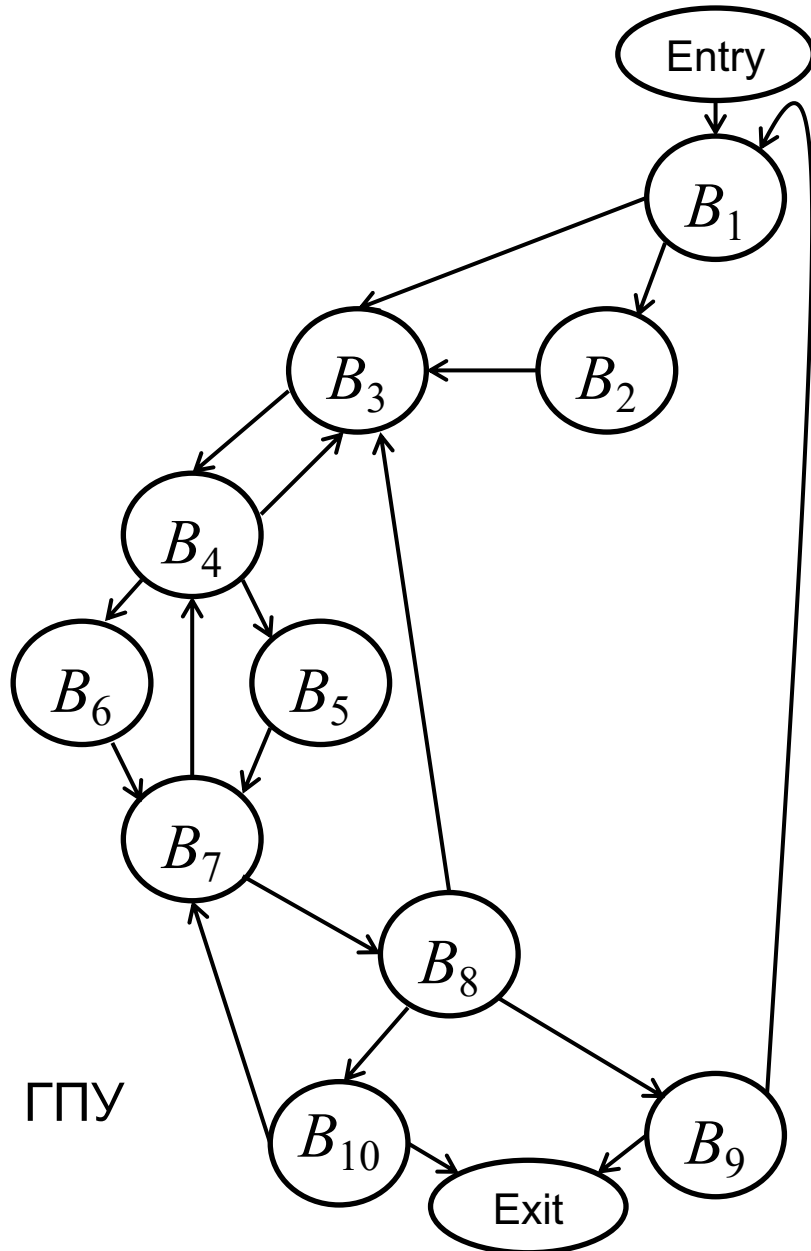
Непосредственный доминатор в каждом списке предпоследний. Соединив дугами для каждого $n \in N$

$IDom(n)$ с n , получим дерево доминаторов. Оно изображено на следующем слайде

+ в начале каждой строки неявно подразумевается *Entry*, который доминирует все блоки ГПУ.

4.1 Доминаторы

4.1.5 Непосредственные доминаторы и дерево доминаторов



4.2 Граница доминирования

4.2.1. Определение границы доминирования (*Dominance Frontier*)

◇ **Определение.** *Границей доминирования* узла n называется множество узлов m , удовлетворяющих условиям:

(1) n является доминатором предшественника m :

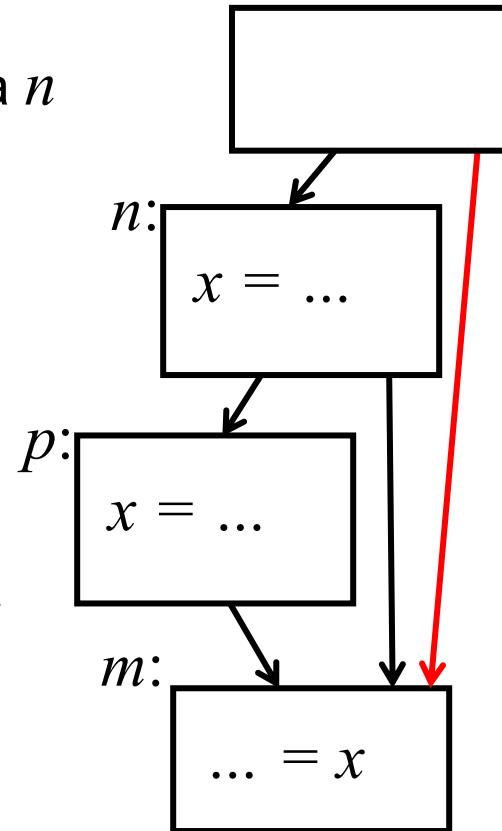
$$\exists p \in \text{Pred}(m) \ \& \ n \in \text{Dom}(p)$$

(2) n не является строгим доминатором m :
 $n \notin (\text{Dom}(m) - \{m\})$.

(иными словами, n не является доминатором m , либо n совпадает с m)

Граница доминирования обозначается, как $DF(n)$.

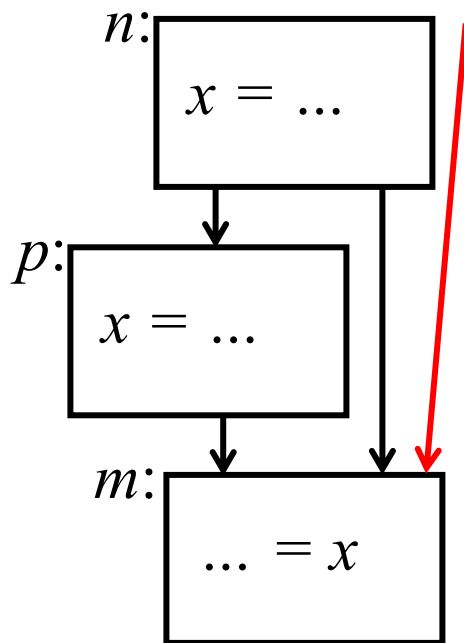
◇ **Неформально:** $DF(n)$ содержит все **первые** узлы, которые достижимы из n , на любом пути графа потока, проходящем через n , но над которыми n не доминирует.



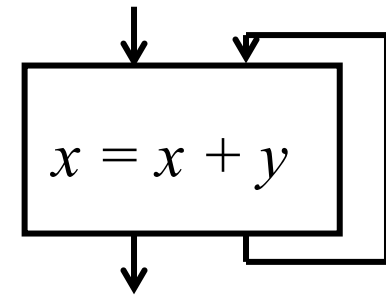
4.2 Граница доминирования

4.2.1. Определение границы доминирования (*Dominance Frontier*)

◇ **Замечание.** Условие про строгость доминатора в (2) Определении 4.2.1 «(2) n не является **строгим** доминатором m » необходимо для определения границы доминирования в случае цикла, тело которого состоит из единственного базового блока, показанном на нижнем рисунке: в этом случае n , m и p совпадают и n является своей собственной границей доминирования.



$n = m = p$:



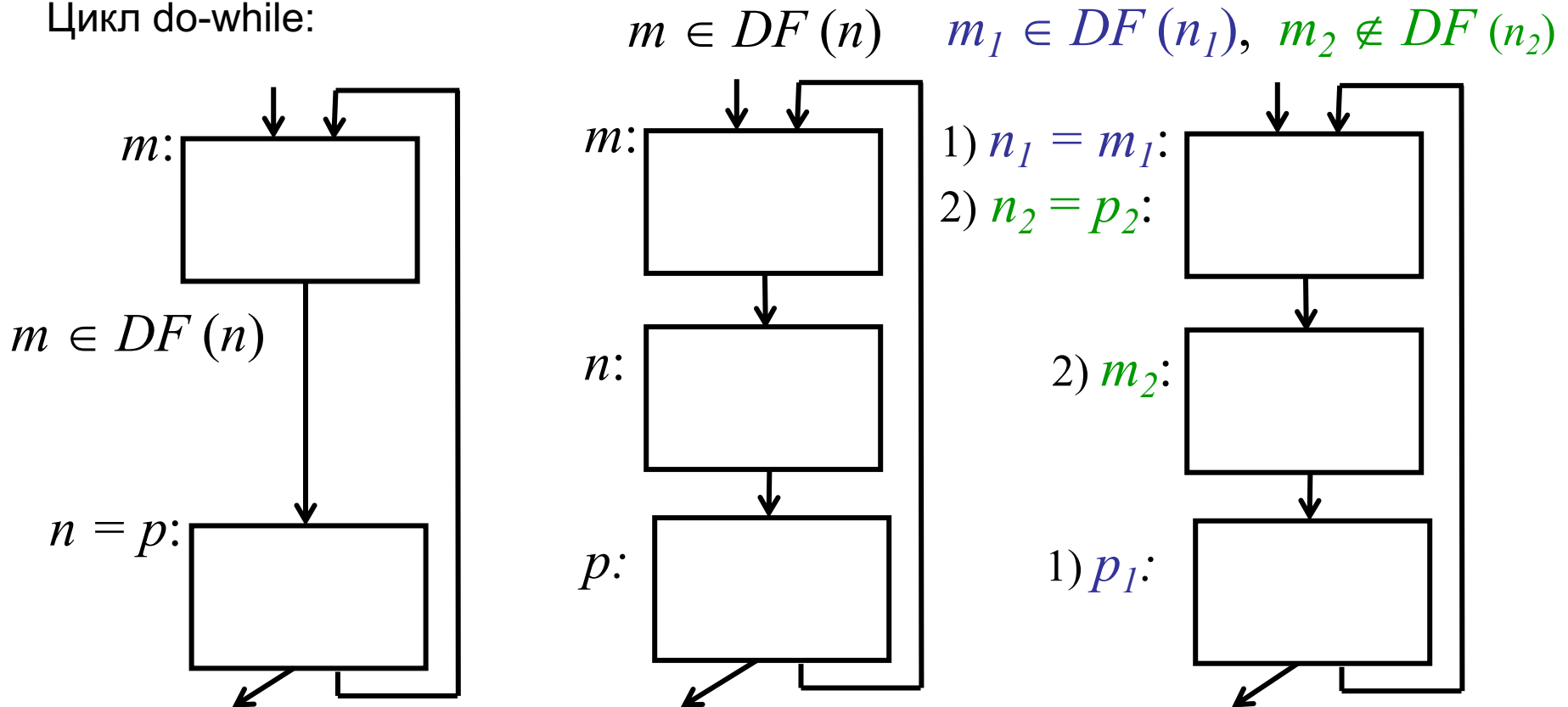
На левом рисунке n не является доминатором m (ни строгим, ни нестрогим).

На правом рисунке n является (нестрогим) доминатором m (т. к. $n = m$), и это допускается определением границы доминирования.

4.2 Граница доминирования

4.2.1. Определение границы доминирования (*Dominance Frontier*)

Цикл do-while:



По определению, $m \in DF(n)$ тогда и только тогда, когда выполняются оба условия:

- n является доминатором предшественника m :

$$\exists p \in Pred(m) \ \& \ n \in Dom(p)$$

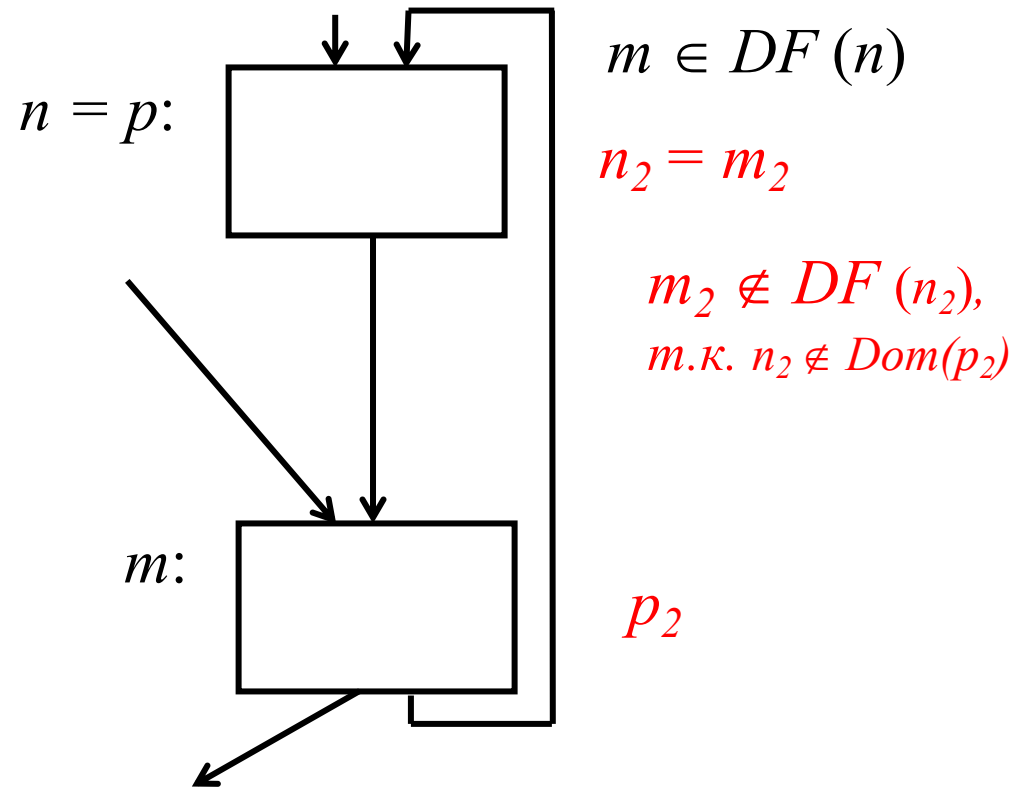
- n не является доминатором m , либо n совпадает с m :

$$n \notin (Dom(m) - \{m\}).$$

4.2 Граница доминирования

4.2.1. Определение границы доминирования (*Dominance Frontier*)

Конструкция, которая похожа на цикл (do-while), но не является циклом:



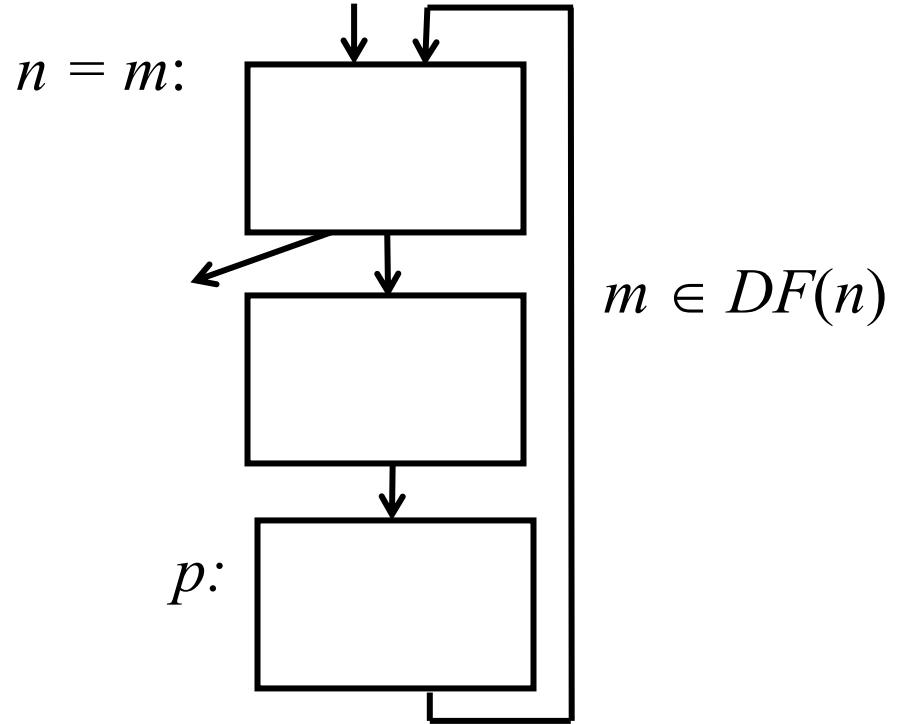
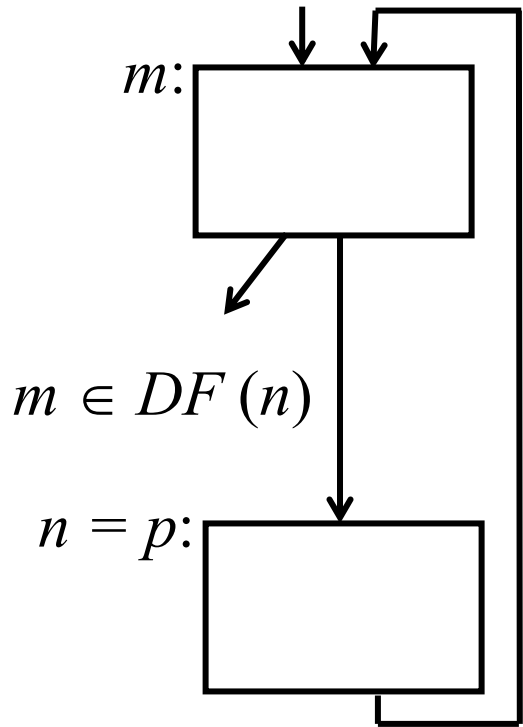
По определению, $m \in DF(n)$ тогда и только тогда, когда выполняются оба условия:

- n является доминатором предшественника m :
$$\exists p \in Pred(m) \ \& \ n \in Dom(p)$$
- n не является доминатором m , либо n совпадает с m :
$$n \notin (Dom(m) - \{m\}).$$

4.2 Граница доминирования

4.2.1. Определение границы доминирования (*Dominance Frontier*)

Цикл while:



По определению, $m \in DF(n)$ тогда и только тогда, когда выполняются оба условия:

- n является доминатором предшественника m :
$$\exists p \in Pred(m) \ \& \ n \in Dom(p)$$
- n не является доминатором m , либо n совпадает с m :
$$n \notin (Dom(m) - \{m\}).$$

4.2 Граница доминирования

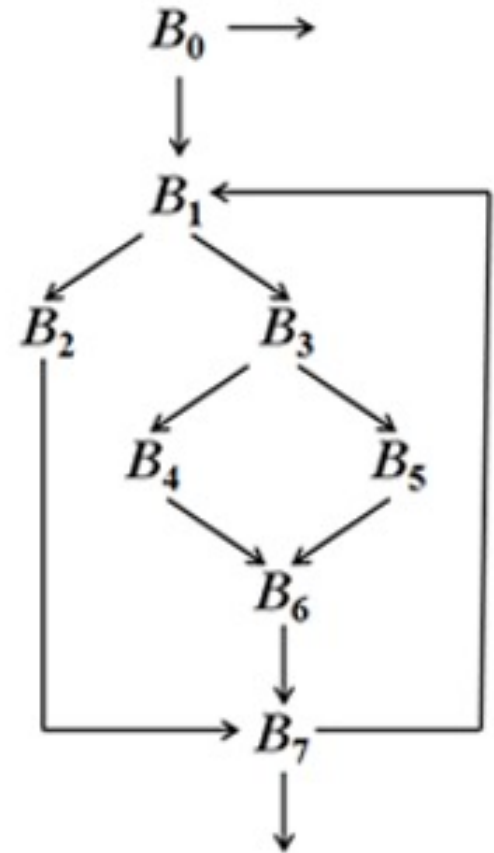
4.2.1. Определение границы доминирования

- ◇ **Пример.** На рисунке справа
 $B_3 \in Dom(B_4)$, $B_3 \in Dom(B_5)$,
 $B_3 \in Dom(B_6)$,
 $B_3 \notin (Dom(B_7) - \{B_7\})$,
т.е. B_3 является доминатором B_4 , B_5 и B_6 ,
но не является доминатором B_7 .

Более того, на любом пути, выходящем из B_3 ,
 B_7 – первая вершина, для которой
 B_3 не является доминатором

Следовательно, $B_7 \in DF(B_3)$

а так как узел B_3 не является доминатором
узлов B_0 , B_1 и B_2 , то $DF(B_3) = \{B_7\}$



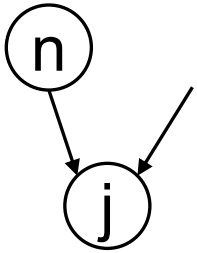
4.2 Граница доминирования

4.2.2. Построение границы доминирования

◇ Свойства узлов, входящих в границу доминирования

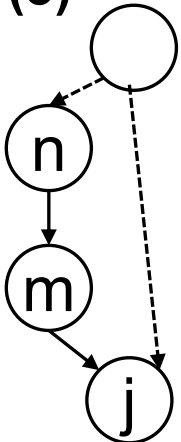
(1) узел, принадлежащий границе доминирования, должен быть точкой сбора графа потока.

(2) если j – точка сбора: $n \in Pred(j) \ \& \ n \notin Dom(j)$,
то $j \in DF(n)$



т.е. точка сбора j входит в границу доминирования любого своего предшественника n , не являющегося доминатором j .

(3) если j – точка сбора:
 $m \in Pred(j) \ \& \ n \in Dom(m) \ \& \ n \notin Dom(j)$, то $j \in DF(n)$



т.е. доминаторы предшественников точки сбора j должны иметь j в своих множествах границ доминирования, если только они не доминируют над j .

4.2 Граница доминирования

4.2.2. Построение границы доминирования

- ◇ Свойства узлов границы доминирования позволяют составить простой алгоритм ее построения

- Шаг 1. Найти все точки сбора j графа потока, т.е. все узлы j , у которых $|Pred(j)| > 1$.
- Шаг 2. Исследовать каждый узел $p \in Pred(j)$ и продвинуться по дереву доминаторов, начиная с p и вплоть до непосредственного доминатора j : при этом j входит в состав границы доминирования каждого из пройденных узлов, за исключением непосредственного доминатора j .

4.2 Граница доминирования

4.2.3. Алгоритм построения границ доминирования

- ◇ **Вход:** граф потока управления с добавленными блоками *Entry* и *Exit*
- ◇ **Выход:** множество границ доминирования для узлов графа потока
- ◇ **Метод:** выполнить следующую программу:

```
for all  $n \in N$  do  $DF(n) = \emptyset$ ;  
for all  $n \in N$  do {  
  if  $|Pred(n)| > 1$  then {  
    for each  $p \in Pred(n)$  do {  
       $r = p$ ;  
      while  $r \neq IDom(n)$  do {  
         $DF(r) = DF(r) \cup \{ n \}$ ;  
         $r = IDom(r)$ ;  
      }  
    }  
  }  
}
```

Предок в ГПУ

распространенная
ошибка на контрольных:
в этом присваивании r и n
часто путают, меняя
их местами

Непосредственный
доминатор (шаг вверх по дереву
доминаторов)

4.2 Граница доминирования

4.2.3. Алгоритм построения границ доминирования

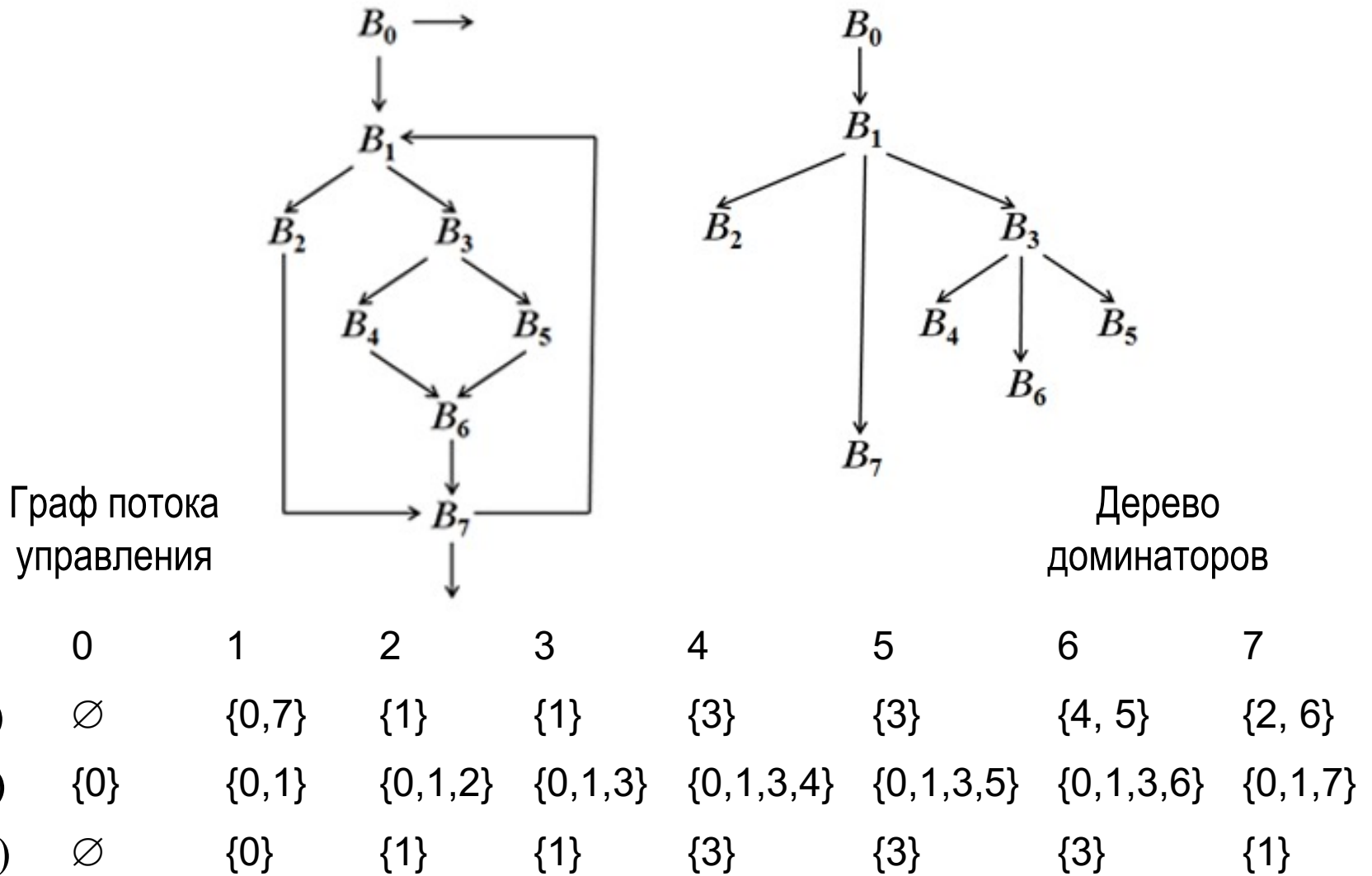
- ◇ **Вход:** граф потока управления с добавленными блоками *Entry* и *Exit*
- ◇ **Выход:** множество границ доминирования для узлов графа потока
- ◇ **Метод:** выполнить следующую программу:

```
for all  $n \in N$  do  $DF(n) = \emptyset$ ;  
for all  $n \in N$  do {  
    if  $|Pred(n)| > 1$  then {  
        for each  $p \in Pred(n)$  do {  
             $r = p$ ;  
            while  $r \neq IDom(n)$  do {
```

У исходного ГПУ должны быть блоки *Entry* и *Exit*, иначе если в первый же блок входит обратная дуга, условие $|Pred(n)| > 1$ не выполнится, и алгоритм не работает корректно. Даже если определить понятие «точки сбора» каким-либо другим способом, у первого узла должен быть предок для его сравнения с *IDom* точки сбора.

4.2 Граница доминирования

4.2.4. Пример применения алгоритма построения границ доминирования



4.2 Граница доминирования

4.2.4. Пример применения алгоритма построения границ доминирования

◇ У графа три точки сбора – входы в узлы B_1 , B_6 и B_7 .

Узел B_6 : $Pred(B_6) = \{B_4, B_5\}$, $Idom(B_6) = \{B_3\}$,

проходим от B_5 до B_3 , добавляем B_6 к $DF(B_5)$,

проходим от B_4 до B_3 , добавляем B_6 к $DF(B_4)$.

Узел B_7 : $Pred(B_7) = \{B_2, B_6\}$, $Idom(B_7) = \{B_1\}$,

проходим от B_2 до B_1 , добавляем B_7 к $DF(B_2)$,

проходим от B_6 до B_3 , добавляем B_7 к $DF(B_6)$

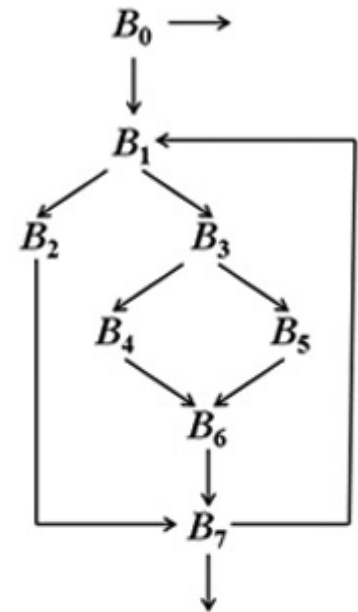
проходим от B_3 до B_1 , добавляем B_7 к $DF(B_3)$

◇ Таблица **текущих** результатов:

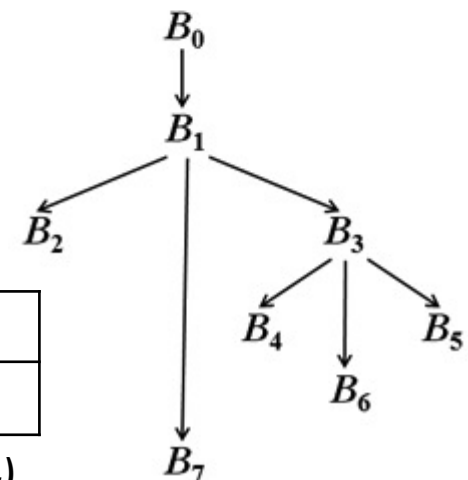
n	0	1	2	3	4	5	6	7
$DF(B_n)$	\emptyset	\emptyset	$\{B_7\}$	$\{B_7\}$	$\{B_6\}$	$\{B_6\}$	$\{B_7\}$	\emptyset

(промежуточные результаты после рассмотрения B_6 и B_7)

Граф потока управления



Дерево доминаторов



4.2 Граница доминирования

4.2.4. Пример применения алгоритма построения границ доминирования

◇ Узел B_1 : $Pred(B_1) = \{B_0, B_7\}$, $Idom(B_1) = \{B_0\}$,

Предок B_0 совпадает с непосредственным доминатором точки сбора B_1 :

$Idom(B_1) = B_0$, значит, точка сбора B_1 не будет добавлена к $DF(B_0)$.

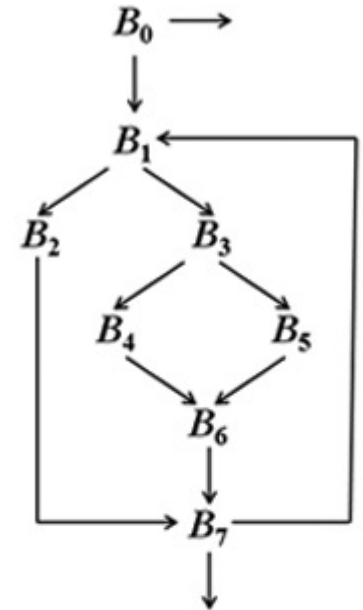
Предок B_7 : 1) проходим от B_7 до B_1 (по обратному ребру), добавляем B_1 к $DF(B_7)$.

2) далее проходим от B_1 до B_0 , добавляем B_1 к $DF(B_1)$.

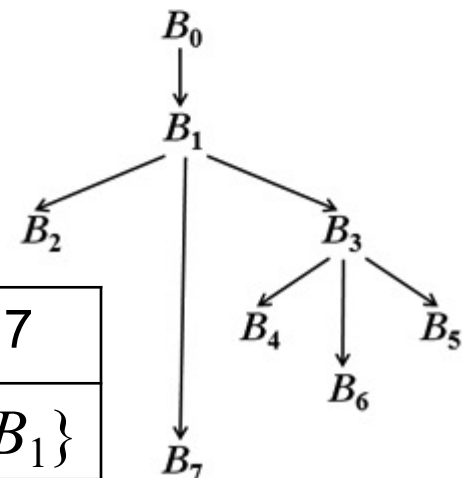
◇ Окончательная таблица результатов:

n	0	1	2	3	4	5	6	7
$DF(B_n)$	\emptyset	$\{B_1\}$	$\{B_7\}$	$\{B_7\}$	$\{B_6\}$	$\{B_6\}$	$\{B_7\}$	$\{B_1\}$

Граф потока управления



Дерево доминаторов



4.3 Постдоминаторы

4.3.1 Определение

- ◇ В ГПУ вершина p является *постдоминатором* вершины n (этот факт записывается как $p \text{ postdom } n$ или $p = \text{Postdom}(n)$), если любой путь от вершины n до вершины $Exit$ проходит через вершину p .
- ◇ **Замечание.** Из определения 4.2.1 следует, что каждая вершина n является постдоминатором самой себя: путь от n до $Exit$ проходит через n .

4.3 Постдоминаторы

4.3.2 Определения

- ◇ *Обратным графом* ориентированного графа $G = \langle N, E \rangle$ называется ориентированный граф $G^R = \langle N, E^R \rangle$, у которого направления всех ребер противоположны.
- ◇ **Постдоминаторы ГПУ – это доминаторы его *обратного графа*.**(*)
- ◇ *Обратная граница доминирования* ($RDF(n)$) вершины $n \in G$ – это обычная граница доминирования в обратном графе G^R .
- ◇ Необходимо отметить, что, несмотря на сказанное выше (*), дерево постдоминаторов для исходного графа G нельзя получить с помощью какого-либо «обращения» или «переподвешивания вверх ногами за Exit» из его дерева доминаторов. В этом легко убедиться на примере (на след. слайде).

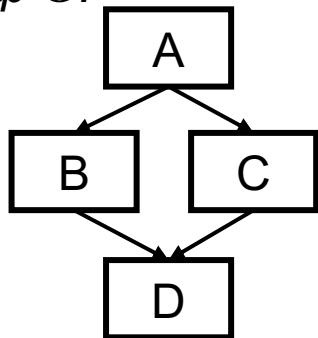
4.3 Постдоминаторы

4.3.2.1 Примеры

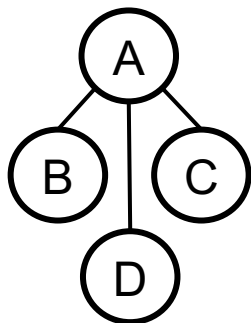
- ◇ Дерево постдоминаторов для исходного графа G нельзя получить с помощью «обращения» его дерева доминаторов.
- ◇ Если бы это было верно, то из $A \text{ dom } B$ следовало бы $B \text{ pdom } A$. Это, конечно же, не так – отношение доминирования рассматривает только пути от Entry, а постдоминирования – только до Exit, и взаимное расположение на путях от Entry не дает никаких гарантий относительно путей к Exit.

В примере ниже видно, что в (2) и (4) даже разные пары вершин соединены ребрами (напр. в (2) **A-B**, а в (4) **D-B**). Т.е. видно, что $(DomTree(G))^R \neq PostDomTree(G)$. Но при этом $DomTree(G^R) = PostDomTree(G)$.

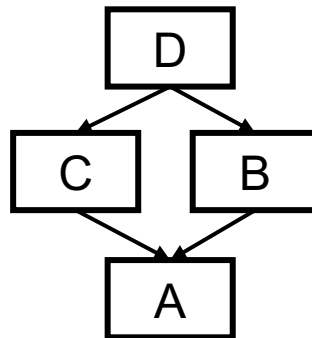
(1) Исходный граф G :



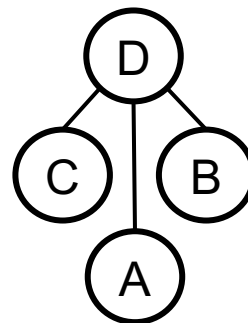
(2) Дерево доминаторов для G :



(3) Обратный граф G^R :



(4) Дерево доминаторов для G^R (оно же постдоминаторов для исходного G):



4.3 Постдоминаторы

4.3.3 Применение постдоминаторов. Зависимость по управлению.

- ◇ По определению, вершина m ГПУ *зависит по управлению* от вершины n тогда, и только тогда, когда:
 - ◇ существует непустой путь T от n до m , такой что $\forall k \in T - \{n\}: m = \text{Postdom}(k)$, т.е. если выполнение программы пошло по пути T , то, чтобы достичь *exit*, оно обязательно пройдет через m .
 - ◇ m не является строгим постдоминатором n .
(У n может быть несколько выходов, так что помимо T возможны и другие пути, проходящие через n , но потом ведущие не в m , а в другие вершины).
- ◇ Другими словами: несколько ветвей исходят из n . Какие-то из них ведут в m , какие-то нет. Решение, принимаемое в ветвлении n определяет, будет ли исполняться m .
- ◇ Обратная граница доминирования позволяет определять границы зависимостей по управлению.

4.3 Постдоминаторы

4.3.4 Эквивалентность по управлению

- ◇ **Определение.** Два базовых блока B_i и B_j эквивалентны по управлению, если B_i выполняется тогда, и только тогда, когда выполняется B_j .
- ◇ **Утверждение.** Если выполняются соотношения:
$$B_i = \text{Dom}(B_j) \text{ и } B_j = \text{Postdom}(B_i)$$
то базовые блоки B_i и B_j эквивалентны по управлению

4.3 Постдоминаторы

4.3.3 Зависимость и эквивалентность по управлению. Примеры.

- ◇ **V3** (а также **V6**) зависит по управлению от **V1**, т. к. в **V1** принимается решение о выборе пути дальнейшего выполнения (и попадания в **V3** и **V6**)
- ◇ С точки зрения определения зависимости по управлению:

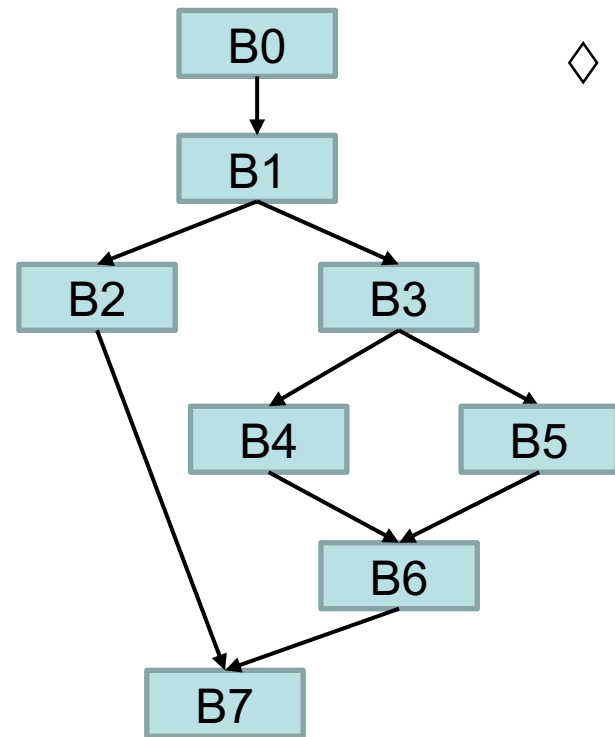
- ◇ существует путь $(V1, V3]$, в котором **V3** является постдоминатором для всех узлов в пути, кроме первого (**V1**), и **V3** не является постдоминатором **V1** =>

V3 зависит по управлению от V1

- ◇ аналогично,

V5 зависит по управлению от V3

- ◇ нет пути, исходящего из **V1**, в котором **V4** или **V5** являются постдоминаторами для узлов этого пути => **V4** и **V5** не зависят по управлению от **V1**, хотя и зависят от **V3**, а **V3** зависит от **V1** => зависимость по управлению не транзитивна



4.3 Постдоминаторы

4.3.3 Зависимость и эквивалентность по управлению. Примеры.

◇ В3 (а также **В6**) зависит по управлению от **В1**, т. к. в В1 принимается решение о выборе пути дальнейшего выполнения (и попадания в В6)

◇ С точки зрения определения зависимости по управлению:

◇ существуют пути (вообще говоря, все пути, начинающихся в В1, проходящие через В3, и заканчивающихся в В6):

(В1, В3, ..., b_i , ..., В6] – в них

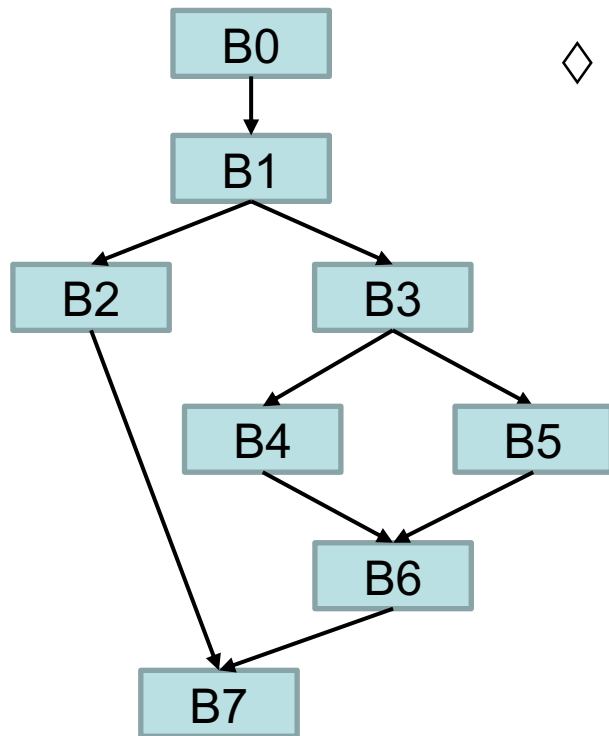
В6 будет постдоминатором для всех

промежуточных узлов b_i (не считая В1)

◇ В6 не является постдоминатором В1

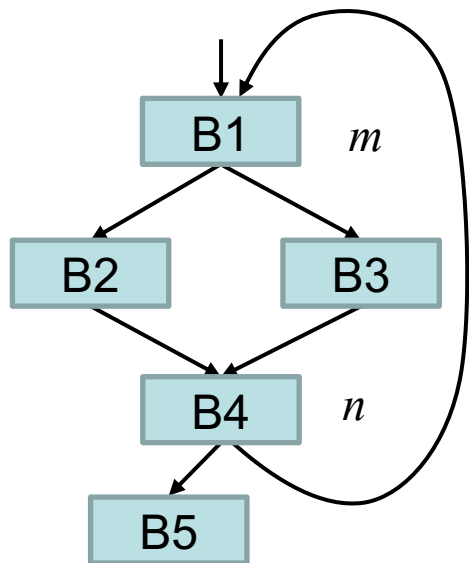
◇ **В1 входит в обратную границу доминирования В6 (и В3)**

◇ В6 не зависит по управлению от В3 – эти блоки **эквивалентны по управлению**: если выполнен код в В3, то обязательно выполнится и В6, и наоборот: если выполнен В6, то прежде был выполнен и В3.



4.3 Постдоминаторы

4.3.3 Зависимость и эквивалентность по управлению. Примеры.



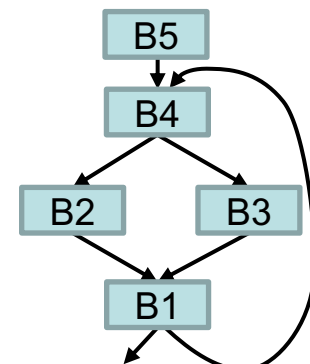
ГПУ $G = \langle N, E \rangle$

m	1	2	3	4	5
RDF(m)	{ 4 }	{ 1 }	{ 1 }	{ 4 }	\emptyset

- ◇ **B1** зависит по управлению от **B4** и по RDF, и по определению: существует путь $(B4, B1]$, в котором B1 постдоминирует все узлы, кроме первого B4, при этом B1 не является постдоминатором B4.
- ◇ B4 зависит по управлению от B4: \exists путь $(B4, B1, B2, B4]$, в котором B4 постдоминирует все узлы, и B4 не является *строгим* постдоминатором B4.
- ◇ **B2** и **B3** зависят по управлению от **B1**.

Определение. Вершина m ГПУ *зависит по управлению* от вершины n тогда, и только тогда, когда:

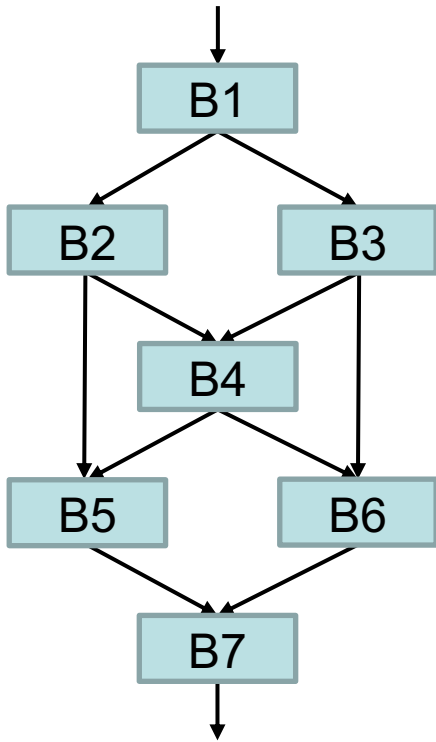
- 1) существует непустой путь T от n до m , такой что $\forall k \in T - \{n\}: m = Postdom(k)$, т.е. если выполнение программы пошло по пути T , то чтобы достичь *Exit*, оно обязательно пройдет через m .
- 2) m не является строгим постдоминатором n .



Обратный граф $G^R = \langle N, E^R \rangle$ для построения RDF для исходного G (DF для G^R)

4.3 Постдоминаторы

4.3.3 Зависимость и эквивалентность по управлению. Примеры.



- ◇ Базовый блок может зависеть по управлению сразу от нескольких блоков.
- ◇ Например, **B5** зависит по управлению от **B2** и **B4**:
 - ◇ $RDF(B5) = \{ B2, B4 \}$
 - ◇ в каждом из этих блоков может быть принято решение о выполнении B5