Wrapup!

CSE 401 Section 10

Michael, Eunia, Kris, Anand, Gavin

Announcements

- HW 4 due today, 12/10/20
  - However everyone has 2 free late days! (Absolute last day is Saturday, 12/12/20)
- CSE M 501 final project due Friday, 12/11/2020, report due 1 day later
  - No late days on these
- Email course staff if you need any additional resources or support!

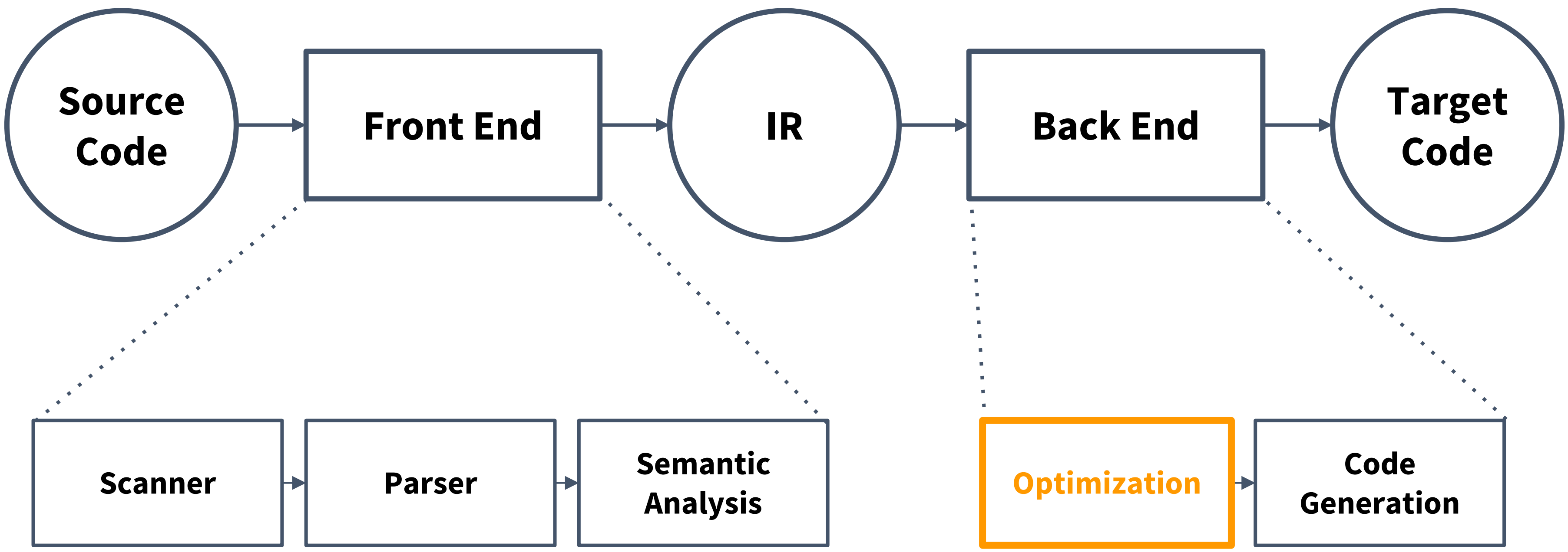
THANK YOU

For an amazing quarter! We hope you learned a lot and enjoyed being your TAs along the way ☺

Agenda for Section

- The choice is yours!
- Optimizations Review
- Practice Problems (Optimizations, Very Busy Expressions, Week 9's Problem 3)
- HW Questions/General Office Hours

Review of Optimizations



Review of Optimizations

Peephole

Local

Intraprocedural / Global

Interprocedural

Review of Optimizations

Peephole	A few Instructions
Local	
Intraprocedural / Global	
Interprocedural	

Review of Optimizations

Peephole	A few Instructions
Local	A Basic Block
Intraprocedural / Global	
Interprocedural	

Review of Optimizations

Peephole	A few Instructions
Local	A Basic Block
Intraprocedural / Global	A Function/Method
Interprocedural	

Review of Optimizations

Peephole	A few Instructions
Local	A Basic Block
Intraprocedural / Global	A Function/Method
Interprocedural	A Program

Problems 1_a and 1_b

```
L0:  a = 0
L1:  b = a + 1
L2:  c = c + b
L3:  a = b * 2
L4:  if a < N goto L1
L5:  return c
```

Block	GEN	KILL	IN (1)	OUT (1)	IN (2)	OUT (2)
L0	L0					
L1	L1					
L2	L2					
L3	L3					
L4						
L5						

```
L0:  a = 0
L1:  b = a + 1
L2:  c = c + b
L3:  a = b * 2
L4:  if a < N goto L1
L5:  return c
```

Block	GEN	KILL	IN (1)	OUT (1)	IN (2)	OUT (2)
L0	L0	L3				
L1	L1					
L2	L2					
L3	L3	L0				
L4						
L5						

```
L0:  a = 0
L1:  b = a + 1
L2:  c = c + b
L3:  a = b * 2
L4:  if a < N goto L1
L5:  return c
```

Block	GEN	KILL	IN (1)	OUT (1)	IN (2)	OUT (2)
L0	L0	L3				
L1	L1		L0			
L2	L2		L0, L1			
L3	L3	L0	L0, L1, L2			
L4			L1, L2, L3			
L5			L1, L2, L3			

```
L0:  a = 0
L1:  b = a + 1
L2:  c = c + b
L3:  a = b * 2
L4:  if a < N goto L1
L5:  return c
```

Block	GEN	KILL	IN (1)	OUT (1)	IN (2)	OUT (2)
L0	L0	L3		L0		
L1	L1		L0	L0, L1		
L2	L2		L0, L1	L0, L1, L2		
L3	L3	L0	L0, L1, L2	L1, L2, L3		
L4			L1, L2, L3	L1, L2, L3		
L5			L1, L2, L3	L1, L2, L3		

```
L0:  a = 0
L1:  b = a + 1
L2:  c = c + b
L3:  a = b * 2
L4:  if a < N goto L1
L5:  return c
```

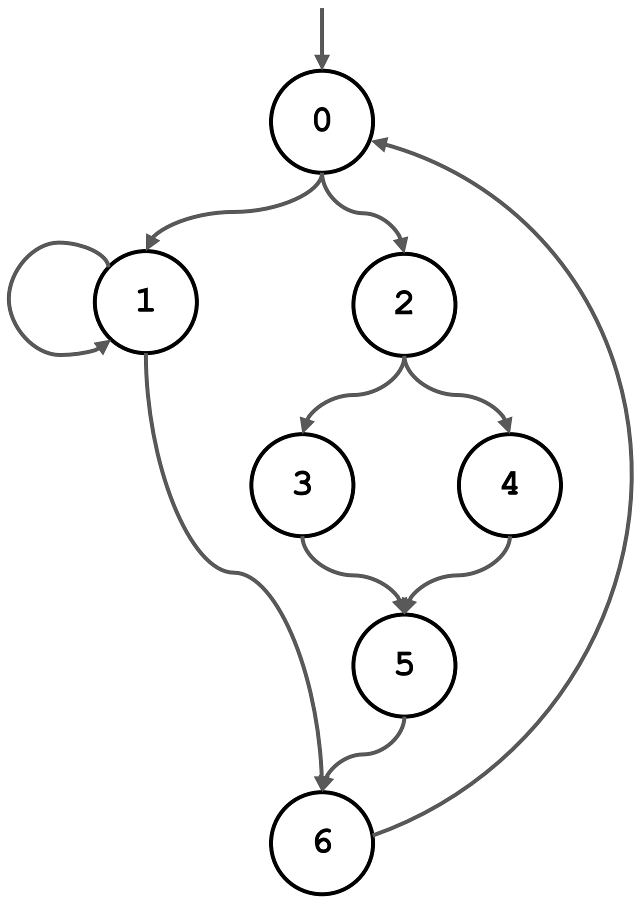
Block	GEN	KILL	IN (1)	OUT (1)	IN (2)	OUT (2)
L0	L0	L3		L0		L0
L1	L1		L0	L0, L1	L0, L1, L2, L3	L0, L1, L2, L3
L2	L2		L0, L1	L0, L1, L2	L0, L1, L2, L3	L0, L1, L2, L3
L3	L3	L0	L0, L1, L2	L1, L2, L3	L0, L1, L2, L3	L1, L2, L3
L4			L1, L2, L3	L1, L2, L3	L1, L2, L3	L1, L2, L3
L5			L1, L2, L3	L1, L2, L3	L1, L2, L3	L1, L2, L3

```
L0:  a = 0
L1:  b = a + 1
L2:  c = c + b
L3:  a = b * 2
L4:  if a < N goto L1
L5:  return c
```

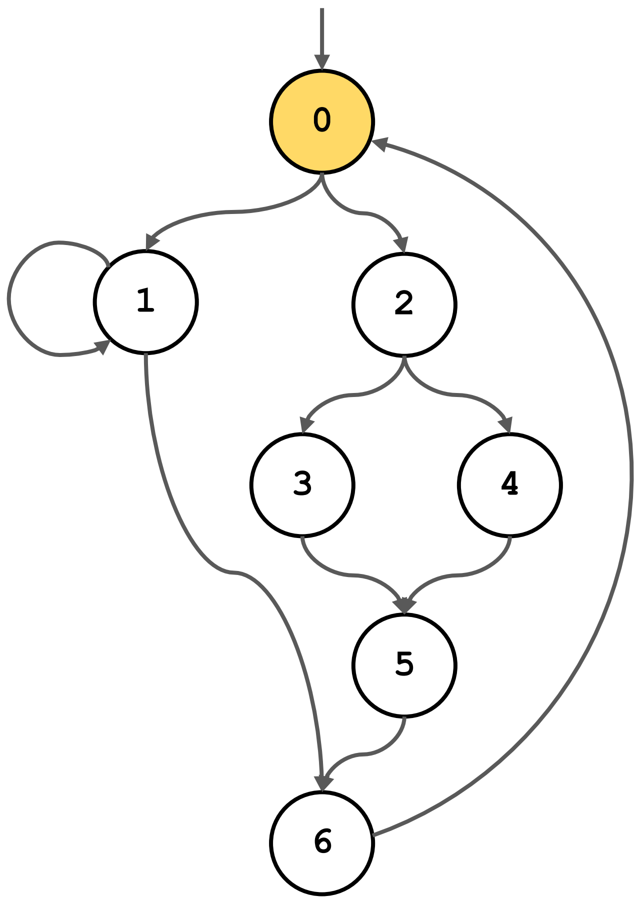
Convergence!

Block	GEN	KILL	IN (1)	OUT (1)	IN (2)	OUT (2)
L0	L0	L3		L0		L0
L1	L1		L0	L0, L1	L0, L1, L2, L3	L0, L1, L2, L3
L2	L2		L0, L1	L0, L1, L2	L0, L1, L2, L3	L0, L1, L2, L3
L3	L3	L0	L0, L1, L2	L1, L2, L3	L0, L1, L2, L3	L1, L2, L3
L4			L1, L2, L3	L1, L2, L3	L1, L2, L3	L1, L2, L3
L5			L1, L2, L3	L1, L2, L3	L1, L2, L3	L1, L2, L3

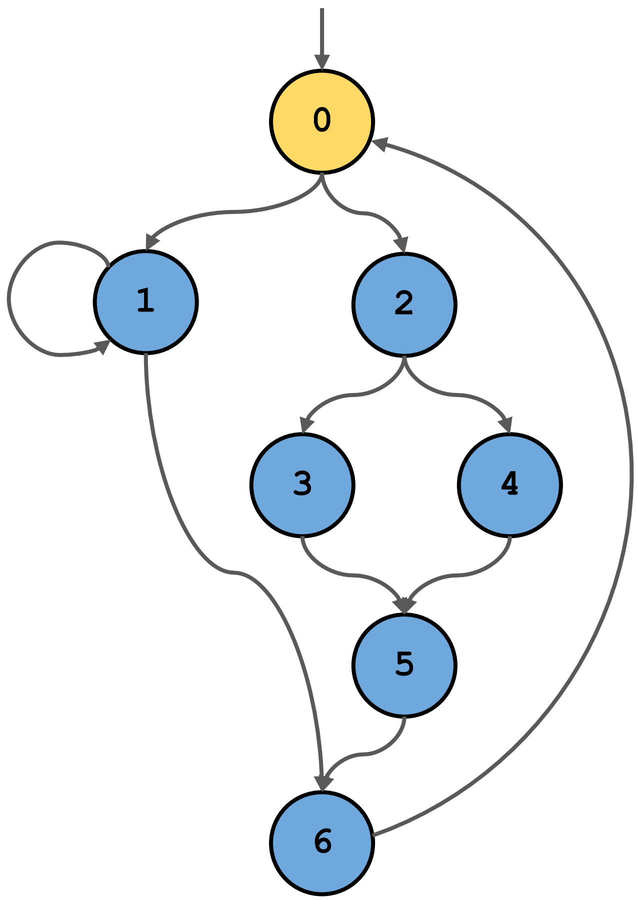
Problem 2(a)



NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0		
1		
2		
3		
4		
5		
6		



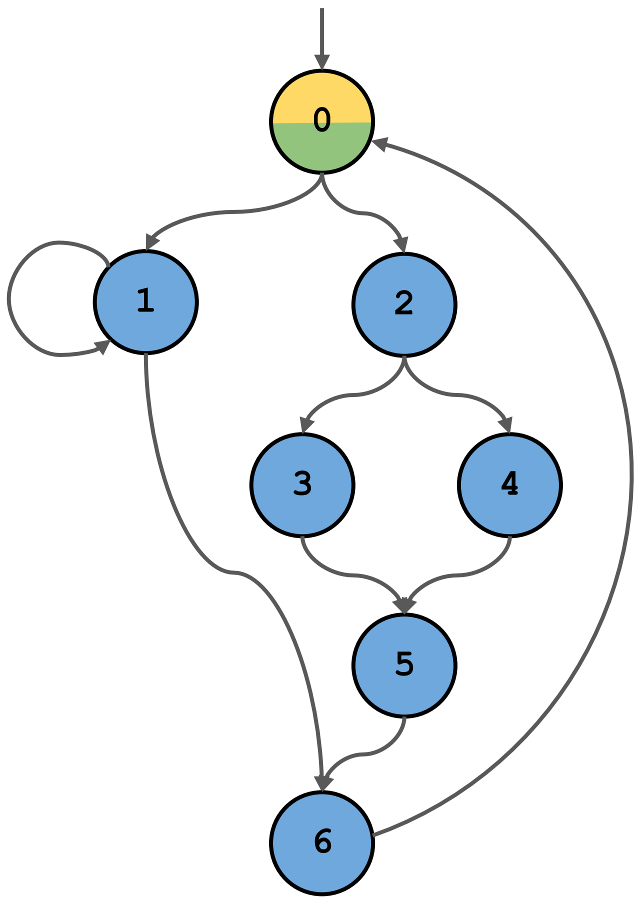
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0		
1		
2		
3		
4		
5		
6		



NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5, 6	
1		
2		
3		
4		
5		
6		

A node **X** ***dominates*** a node **Y** iff every path from the entry point of the control flow graph to **Y** includes **X** .
A node **X** ***strictly dominates*** a node **Y** iff **X** dominates **Y** and **X** ≠ **Y**

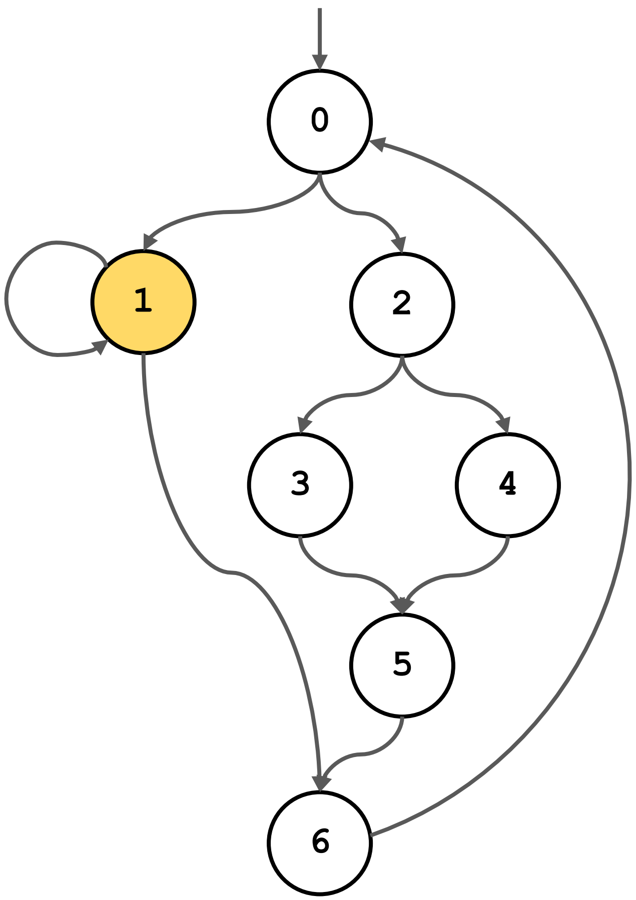
Need to go through 0 to get through 1, 2, 3, 4, 5, 6 and 0 cannot strictly dominate itself



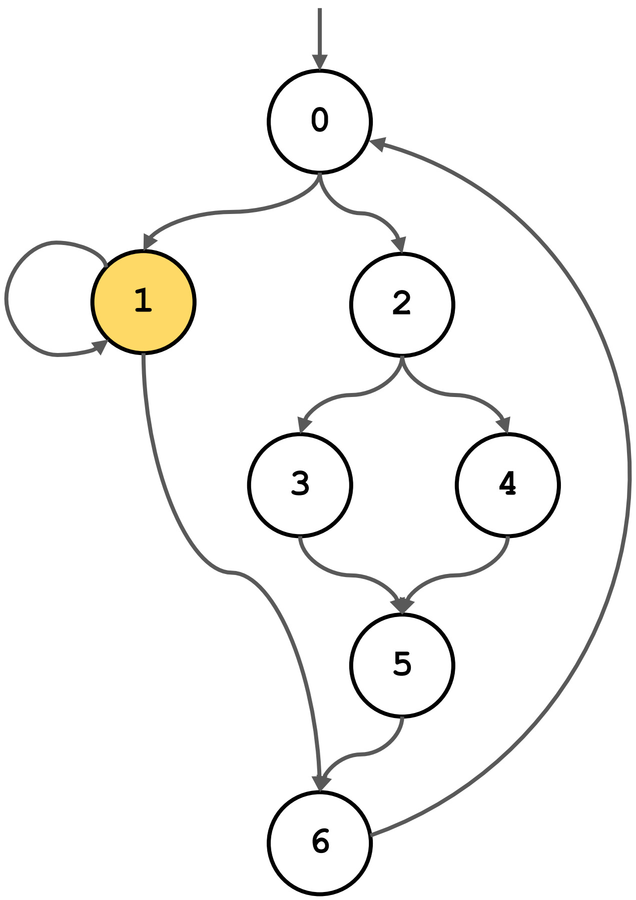
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5, 6	0
1		
2		
3		
4		
5		
6		

A node **Y** is in the ***dominance frontier*** of node **X** iff **X** dominates an immediate predecessor of **Y** but **X** does not strictly dominate **Y** .
A node **0** is in the ***dominance frontier*** of node **0** iff **0** dominates an immediate predecessor **(6)** of **0** but **0** does not strictly dominate 0

0 dominates 6, 6 is an immediate predecessor of 0, 0 does not strictly dominate 0



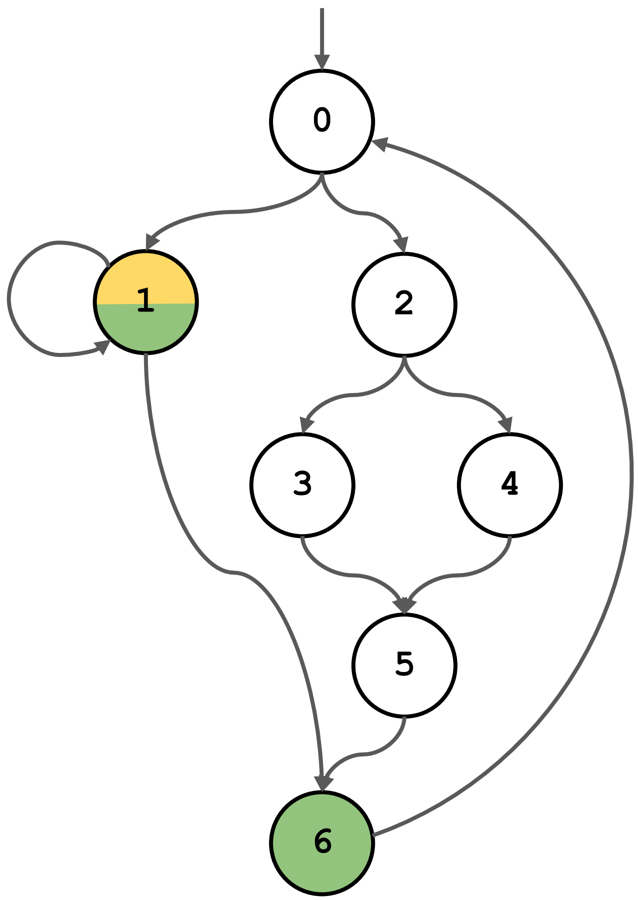
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5, 6	0
1		
2		
3		
4		
5		
6		



NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5, 6	0
1	∅	
2		
3		
4		
5		
6		

A node **X** ***dominates*** a node **Y** iff every path from the entry point of the control flow graph to **Y** includes **X** .
A node **X** ***strictly dominates*** a node **Y** iff **X** dominates **Y** and $X \neq Y$

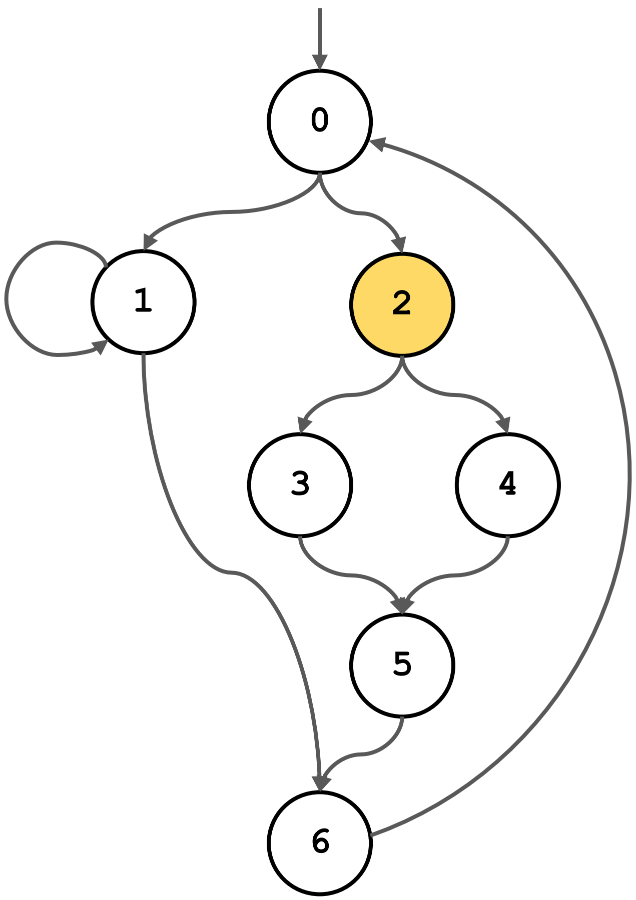
1 does not dominate 6 because there is a path from 5 that doesn't include 1. 1 does not strictly dominate itself



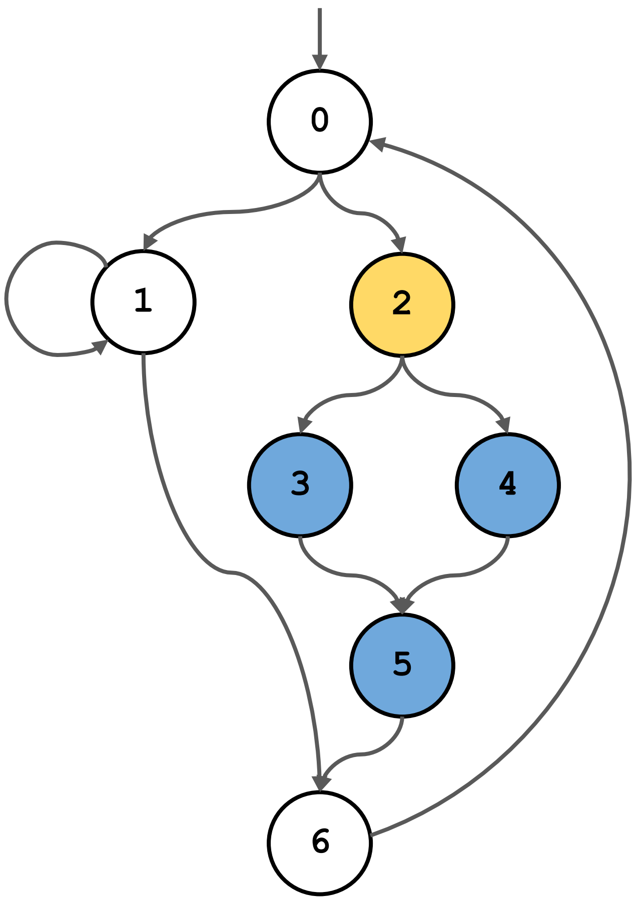
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5, 6	0
1	Ø	1, 6
2		
3		
4		
5		
6		

A node **Y** is in the ***dominance frontier*** of node **X** iff **X** dominates an immediate predecessor of **Y** but **X** does not strictly dominate **Y** .

X = 1, Y = 6, 1 dominates 1, 1 is an immediate predecessor of 6, 1 does not strictly dominate 6
X = 1, Y = 1, 1 dominates 1, 1 is an immediate predecessor of 1, 1 does not strictly dominate 1



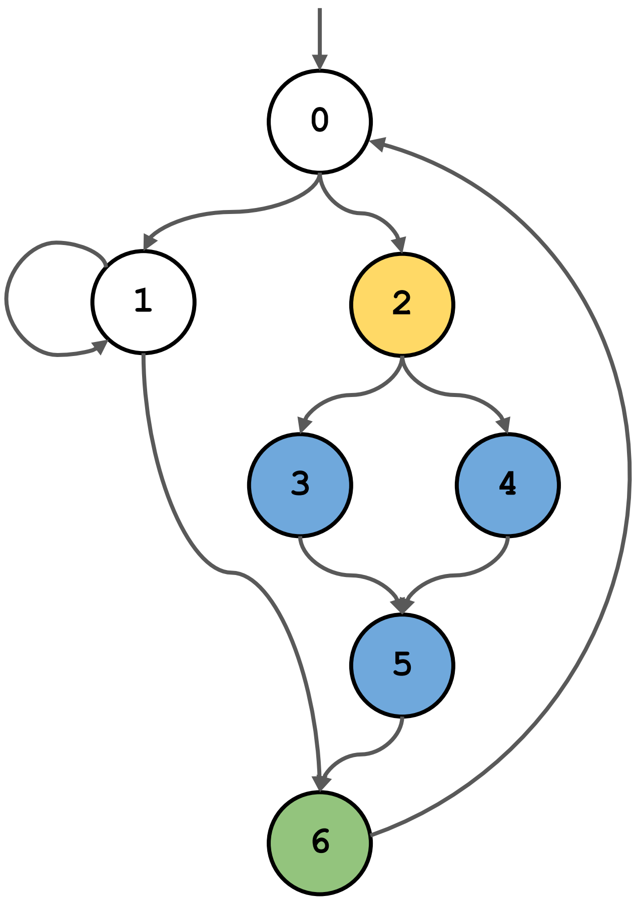
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5, 6	0
1	∅	1, 6
2		
3		
4		
5		
6		



NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5, 6	0
1	Ø	1, 6
2	3, 4, 5	
3		
4		
5		
6		

A node **X** ***dominates*** a node **Y** iff every path from the entry point of the control flow graph to **Y** includes **X** .
A node **X** ***strictly dominates*** a node **Y** iff **X** dominates **Y** and **X** ≠ **Y**

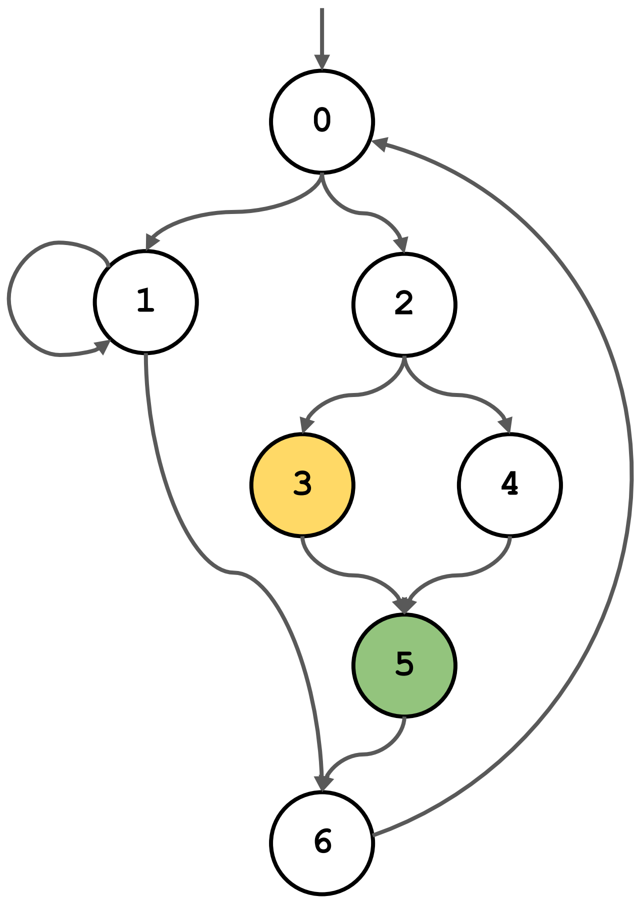
Need to go through 2 to get through 3, 4, 5 and 2 cannot strictly dominate itself



NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5, 6	0
1	Ø	1, 6
2	3, 4, 5	6
3		
4		
5		
6		

A node **Y** is in the ***dominance frontier*** of node **X** iff **X** dominates an immediate predecessor of **Y** but **X** does not strictly dominate **Y** .

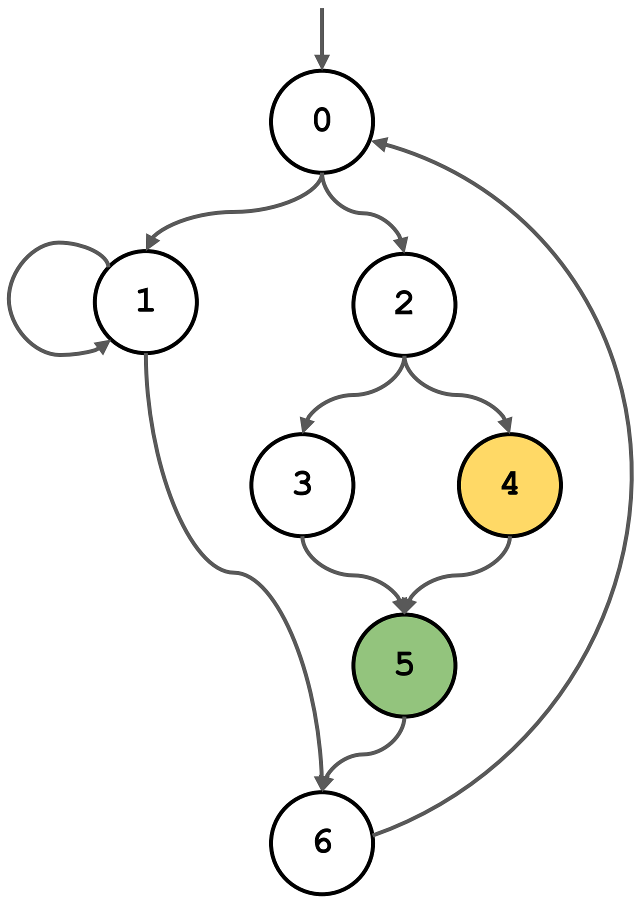
X = 2, Y = 6, 2 dominates 5, 5 is an immediate predecessor of 6, 2 does not strictly dominate 6



NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5, 6	0
1	∅	1, 6
2	3, 4, 5	6
3	∅	5
4		
5		
6		

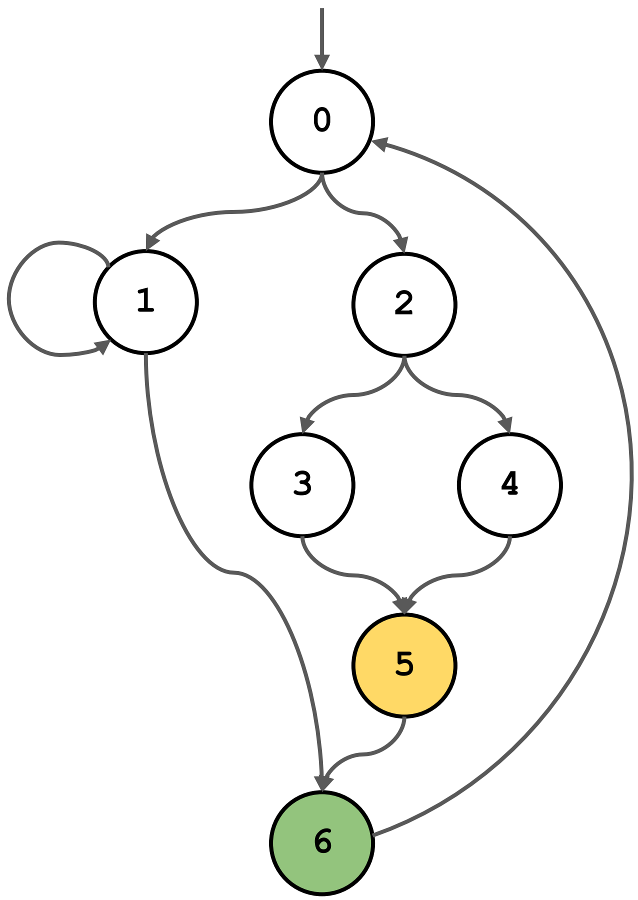
3 does not strictly dominate 5 (path through 4) and therefore does not strictly dominate anything else

3 dominates 3, 3 is an immediate predecessor of 5, 3 does not strictly dominate 5



NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5, 6	0
1	Ø	1, 6
2	3, 4, 5	6
3	Ø	5
4	Ø	5
5		
6		

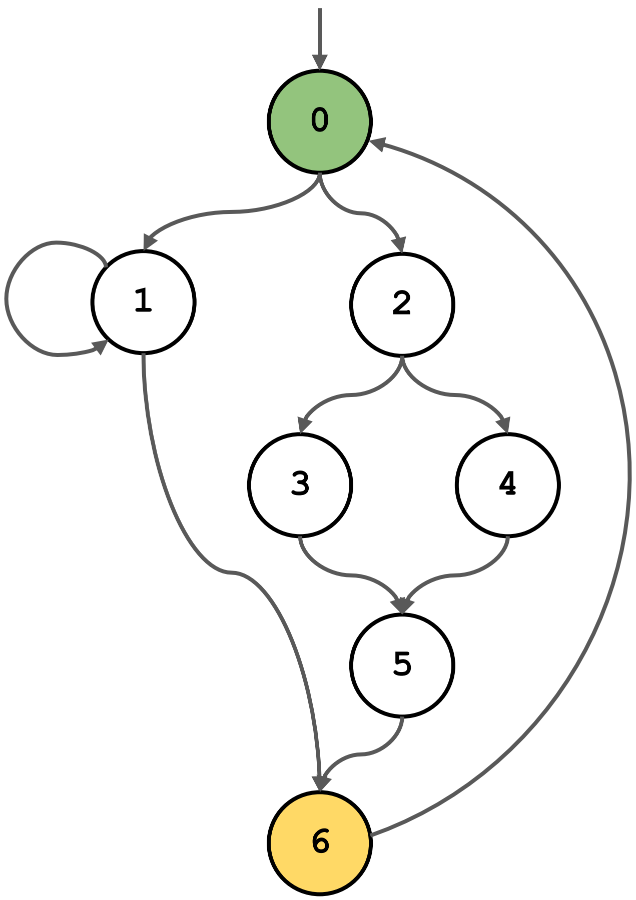
Same as previous slide but with 4 instead of 3



NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5, 6	0
1	Ø	1, 6
2	3, 4, 5	6
3	Ø	5
4	Ø	5
5	Ø	6
6		

5 does not strictly dominate 6 (path through 1) and therefore does not strictly dominate anything else

5 dominates 5, 5 is an immediate predecessor of 6, 5 does not strictly dominate 6



NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5, 6	0
1	Ø	1, 6
2	3, 4, 5	6
3	Ø	5
4	Ø	5
5	Ø	6
6	Ø	0

6 does not strictly dominate 0 (path through 0) and therefore does not strictly dominate anything else

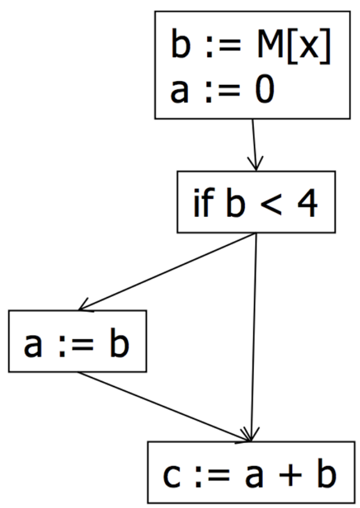
6 dominates 6, 6 is an immediate predecessor of 0, 6 does not strictly dominate 0

Problem 2(b)

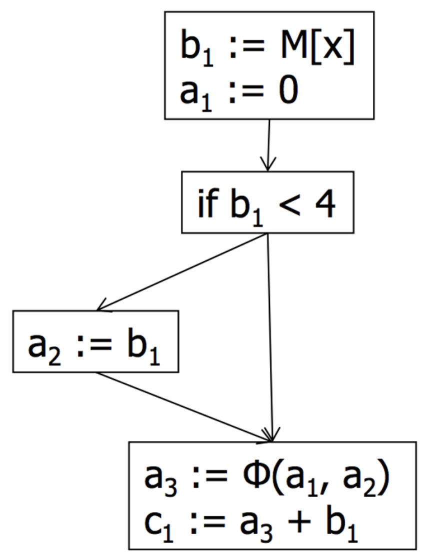
Phi-Functions

- A way to represent multiple possible values for a certain definition
  - Not a "real" instruction – just a form of bookkeeping needed for SSA

Original

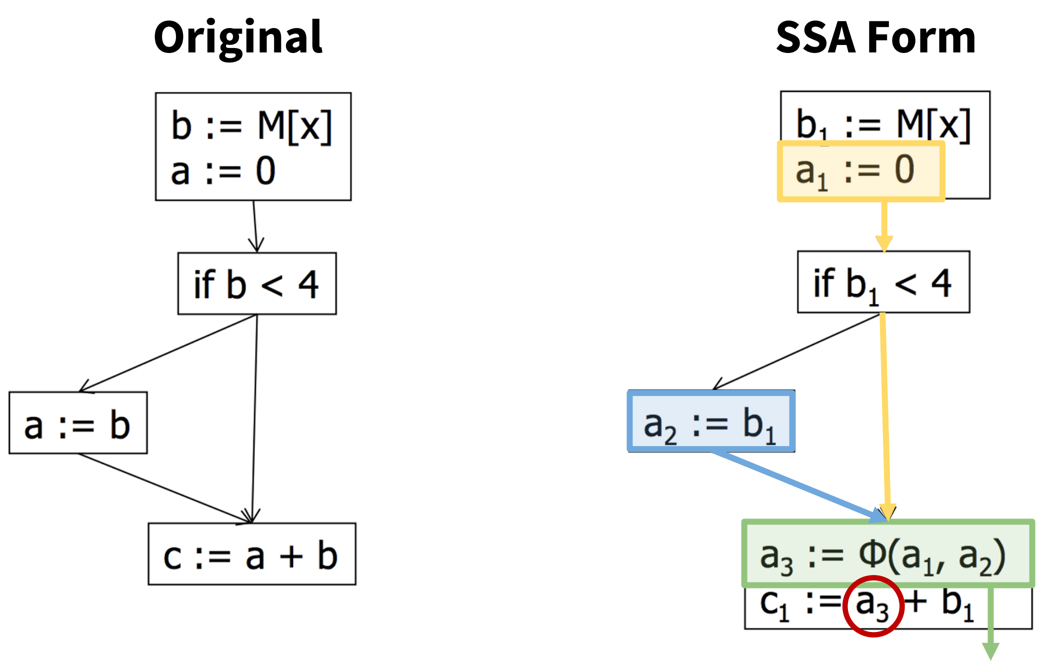


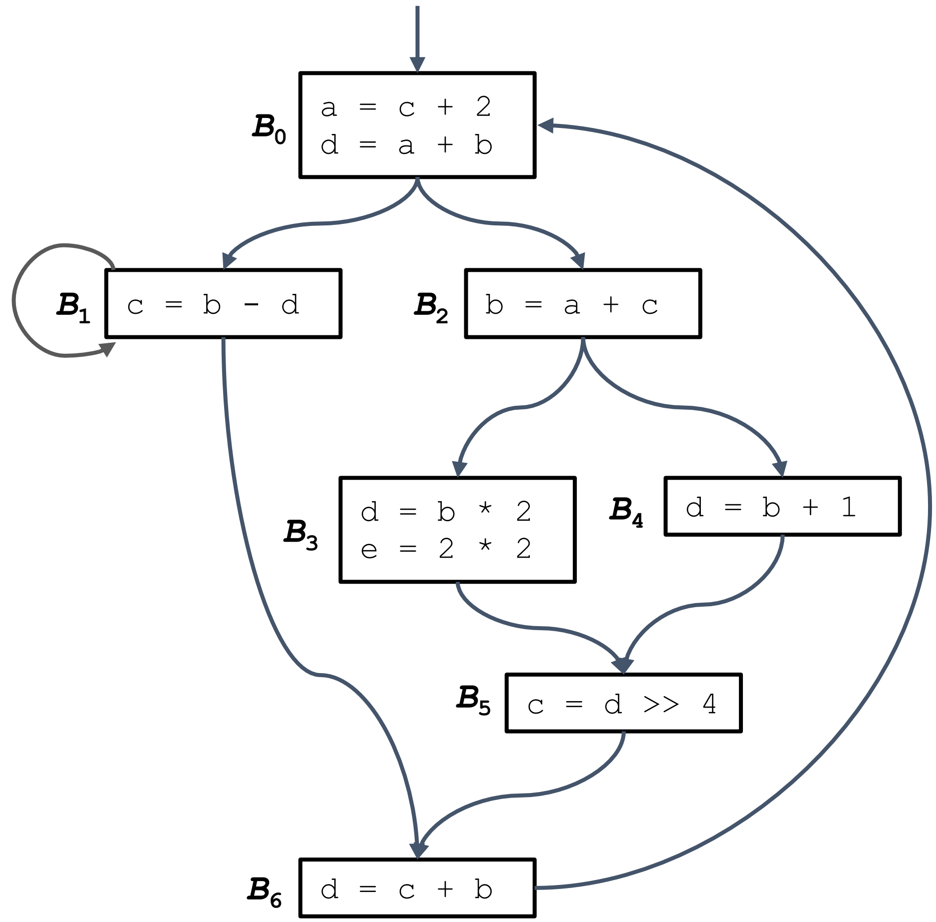
SSA Form



Where to place Phi-Functions?

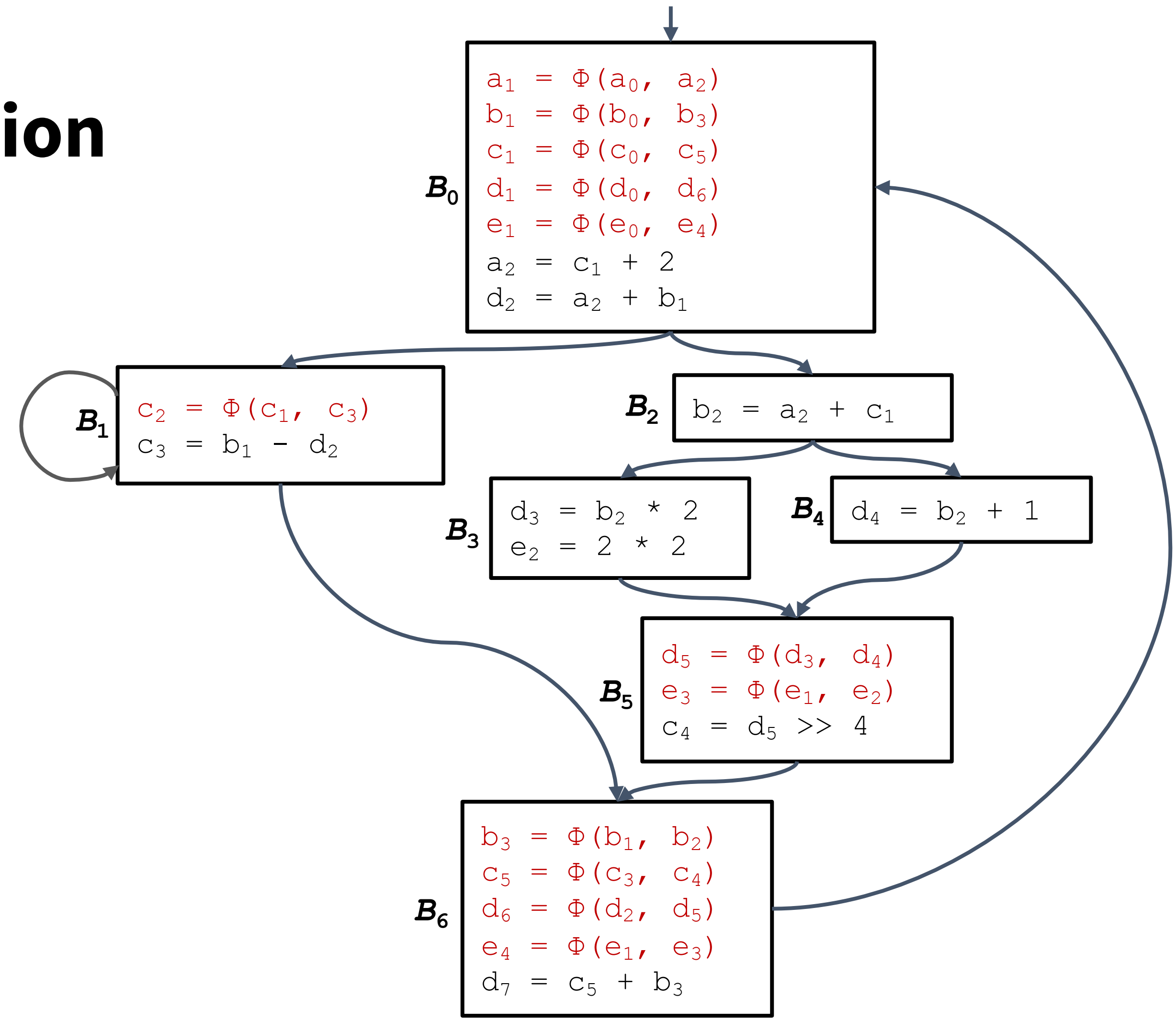
- Wherever a variable has multiple possible definitions entering a block
  - Inefficient (and unnecessary!) to consider all possible phi-functions at the start of each block





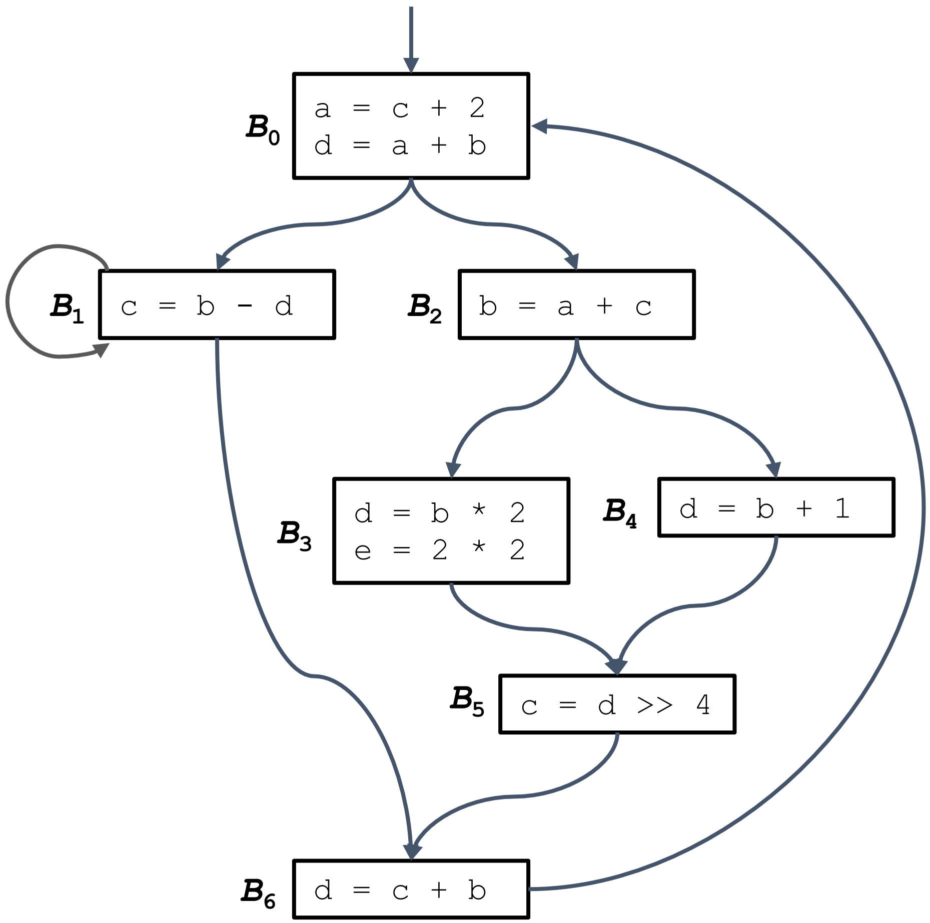
B0
a = c + 2
d = a + b
B1
c = b - d
B2
b = a + c
B3
d = b * 2
e = 2 * 2
B4
d = b + 1
B5
c = d >> 4
B6
d = c + b

Solution


B0
a1 = Φ(a0, a2)
b1 = Φ(b0, b3)
c1 = Φ(c0, c5)
d1 = Φ(d0, d6)
e1 = Φ(e0, e4)
a2 = c1 + 2
d2 = a2 + b1
B1
c2 = Φ(c1, c3)
c3 = b1 - d2
B2
b2 = a2 + c1
B3
d3 = b2 * 2
e2 = 2 * 2
B4
d4 = b2 + 1
B5
d5 = Φ(d3, d4)
e3 = Φ(e1, e2)
c4 = d5 >> 4
B6
b3 = Φ(b1, b2)
c5 = Φ(c3, c4)
d6 = Φ(d2, d5)
e4 = Φ(e1, e3)
d7 = c5 + b3

Converting to SSA

1. **Compute the dominance frontier of each node** ⟵ Already done (in problem 2a)
2. Determine which variables need merging in each node
3. Assign numbers to definitions and add phi functions



Step 1: Dominance Frontiers

NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5, 6	0
1	Ø	1, 6
2	3, 4, 5	6
3	Ø	5
4	Ø	5
5	Ø	6
6	Ø	0

Converting to SSA

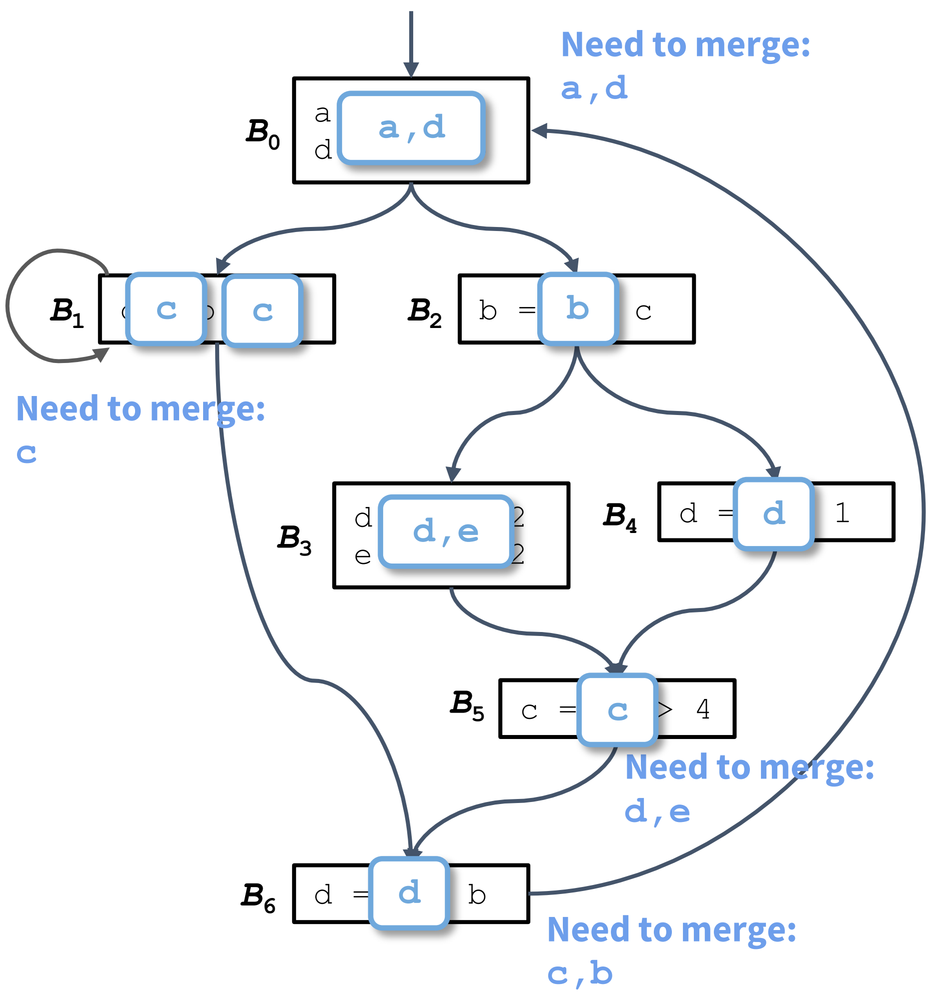
1. Compute the dominance frontier of each node ☑

2. **Determine which variables need merging in each node** ← **We will compute using the dominance frontiers**

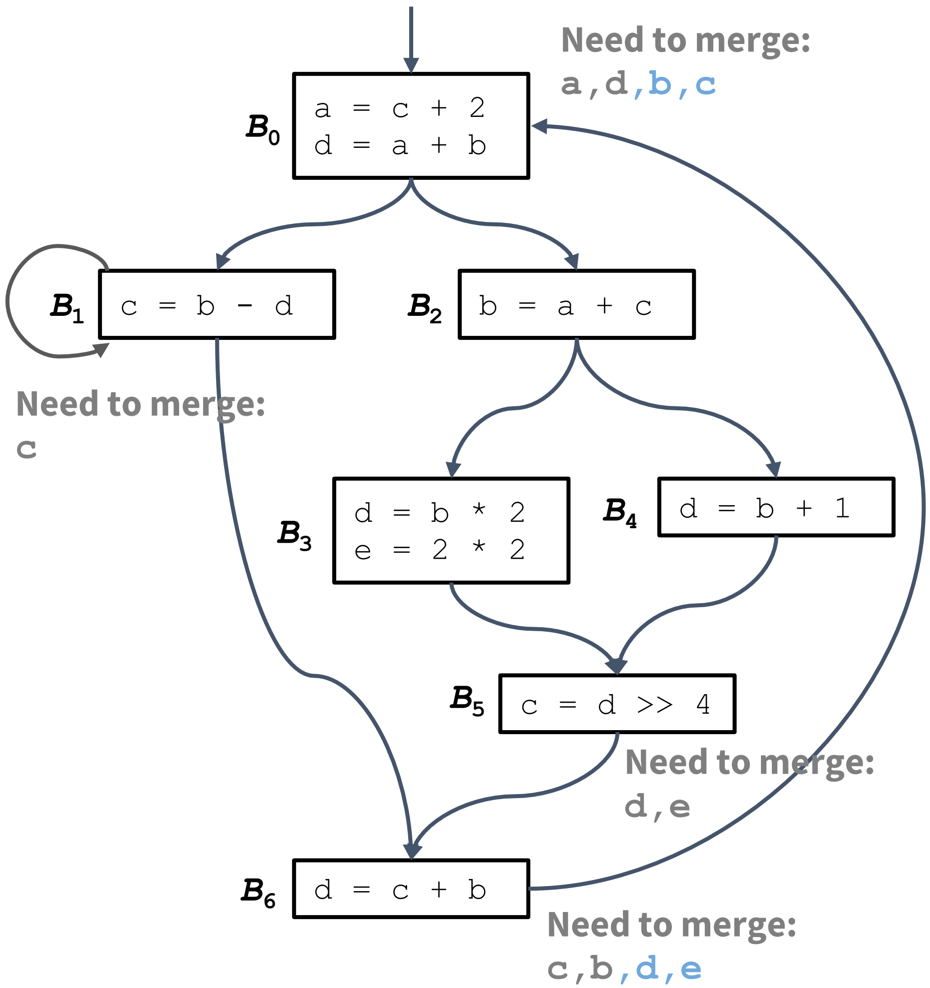
3. Assign numbers to definitions and add phi functions

Step 2: Determine Necessary Merges

ITERATION 1: Each node in the dominance frontier of node X will merge any definitions created in node X.



NODE		DOMINANCE FRONTIER
0	a,d	0
1	c	1, 6
2	b	6
3	d,e	5
4	d	5
5	c	6
6	d	0



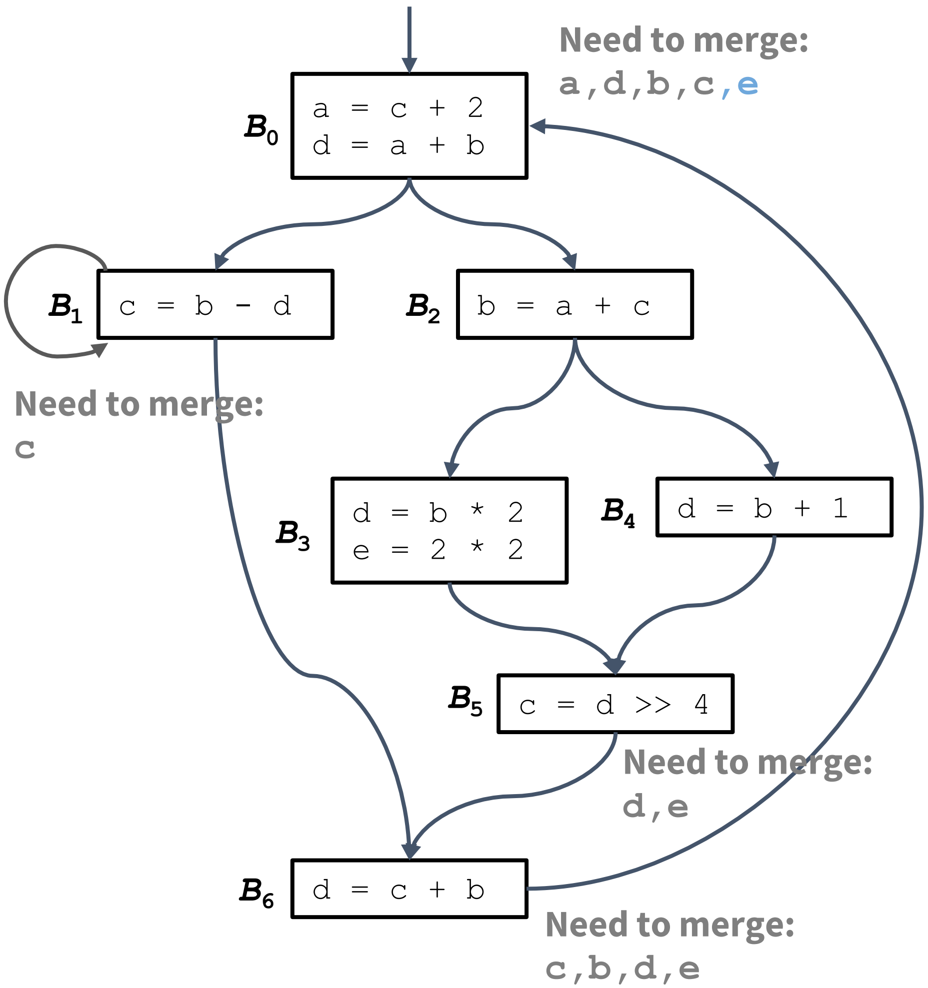
Step 2: Determine Necessary Merges

ITERATION 2: Each merge will create a new definition, which may need merging again.

NODE		DOMINANCE FRONTIER
0		0
1		1, 6
2		6
3		5
4		5
5	d,e	6
6	b,c	0

Step 2: Determine Necessary Merges

ITERATION 3: Each merge will create a new definition, which may need merging again.



NODE		DOMINANCE FRONTIER
0		0
1		1, 6
2		6
3		5
4		5
5		6
6	d,e	0

Converting to SSA

1. Compute the dominance frontier of each node ✓
2. Determine which variables need merging in each node ✓
3. **Assign numbers to definitions and add phi functions** ← **Place phi functions first, then increment subscripts**

Step 3: Assign Definition Numbers

Merges go first, and each successive definition of a variable should increment its index by 1.

B_0

```
a = c + 2
d = a + b
```

Need to merge:
a,b,c,d,e

⇒

B_0

$$
\begin{aligned}
a_1 &= \Phi(a_0,\ a_2) \\
b_1 &= \Phi(b_0,\ b_3) \\
c_1 &= \Phi(c_0,\ c_5) \\
d_1 &= \Phi(d_0,\ d_7) \\
e_1 &= \Phi(e_0,\ e_4) \\
a_2 &= c_1 + 2 \\
d_2 &= a_2 + b_1
\end{aligned}
$$

Note: these subscripts determined after doing the rest of the CFG!

Step 3: Assign Definition Numbers

Merges go first, and each successive definition of a variable should increment its index by 1.

***B*₁**

$c = b - d$

Need to merge:
c

⇒

***B*₁**

$c_2 = \Phi(c_1, c_3)$
$c_3 = b_1 - d_2$

Note: must merge its own (later) definition because of the back-edge!

Step 3: Assign Definition Numbers

Merges go first, and each successive definition of a variable should increment its index by 1.

B_2

```
b = a + c
```

Nothing to merge

⇒

B_2

$b_2 = a_2 + c_1$

Step 3: Assign Definition Numbers

Merges go first, and each successive definition of a variable should increment its index by 1.

B_3

```
d = b * 2
e = 2 * 2
```

Nothing to merge

⇒

B_3

$d_3 = b_2 * 2$

$e_2 = 2 * 2$

Step 3: Assign Definition Numbers

Merges go first, and each successive definition of a variable should increment its index by 1.

B_4

```
d = b + 1
```

Nothing to merge

⇒

B_4

$d_4 = b_2 + 1$

Step 3: Assign Definition Numbers

Merges go first, and each successive definition of a variable should increment its index by 1.

B$_5$

```
c = d >> 4
```

Need to merge:
d,e

B$_5$

$d_5 = \Phi(d_3,\ d_4)$
$e_3 = \Phi(e_1,\ e_2)$
$c_4 = d_5 >> 4$

Step 3: Assign Definition Numbers

Merges go first, and each successive definition of a variable should increment its index by 1.

B_6

```
d = c + b
```

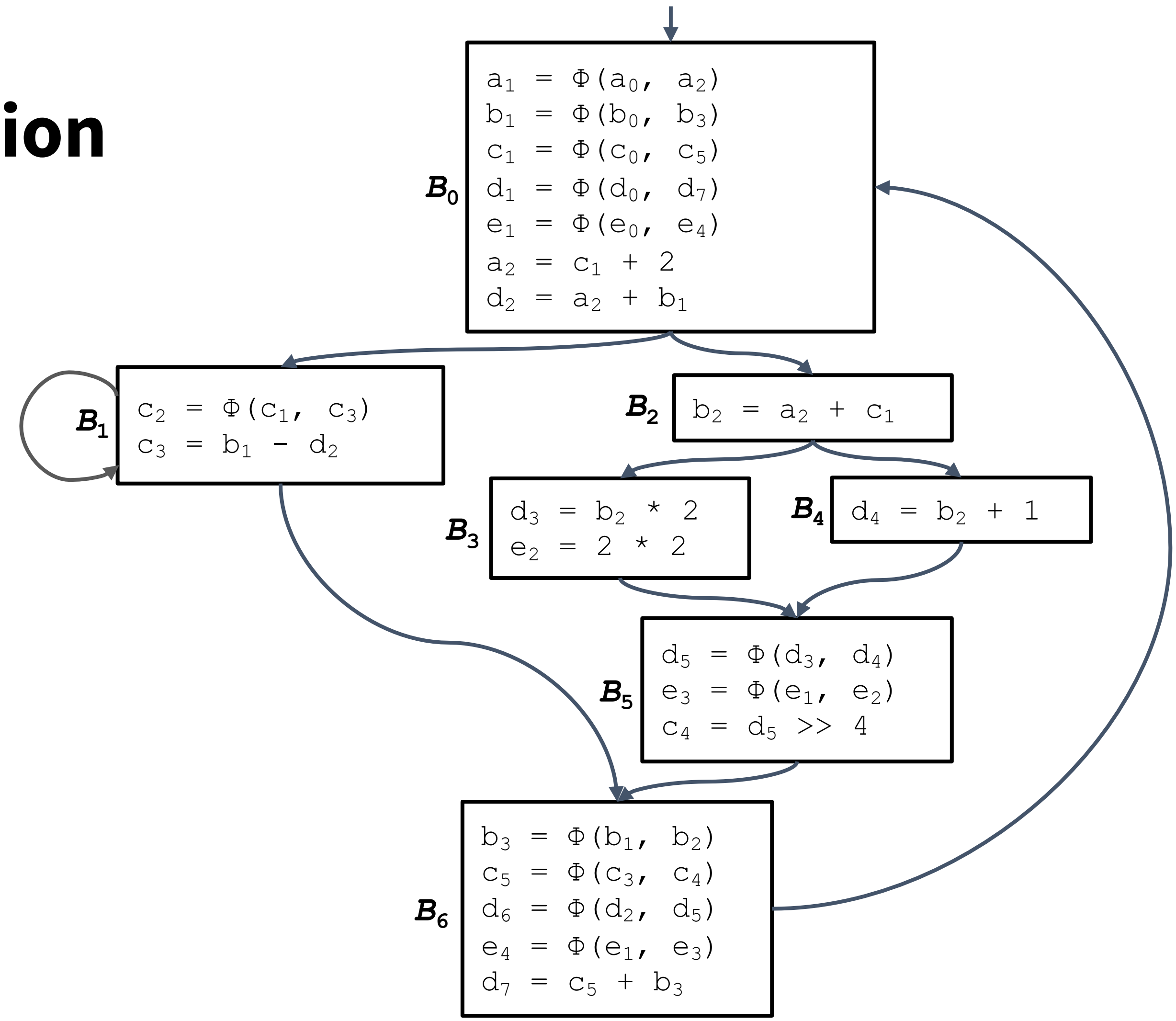
Need to merge:
`b,c,d,e`

⇨

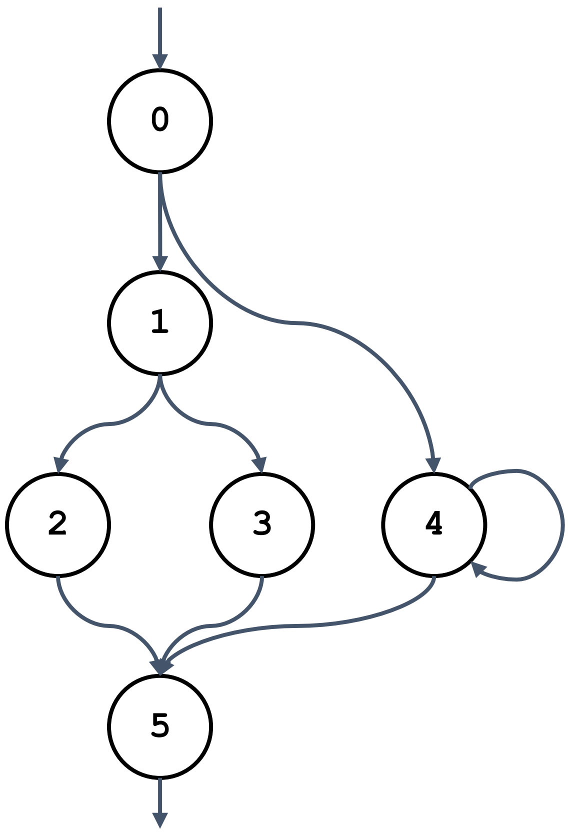
B_6

$$
\begin{aligned}
b_3 &= \Phi(b_1,\ b_2)\\
c_5 &= \Phi(c_3,\ c_4)\\
d_6 &= \Phi(d_2,\ d_5)\\
e_4 &= \Phi(e_1,\ e_3)\\
d_7 &= c_5 + b_3
\end{aligned}
$$

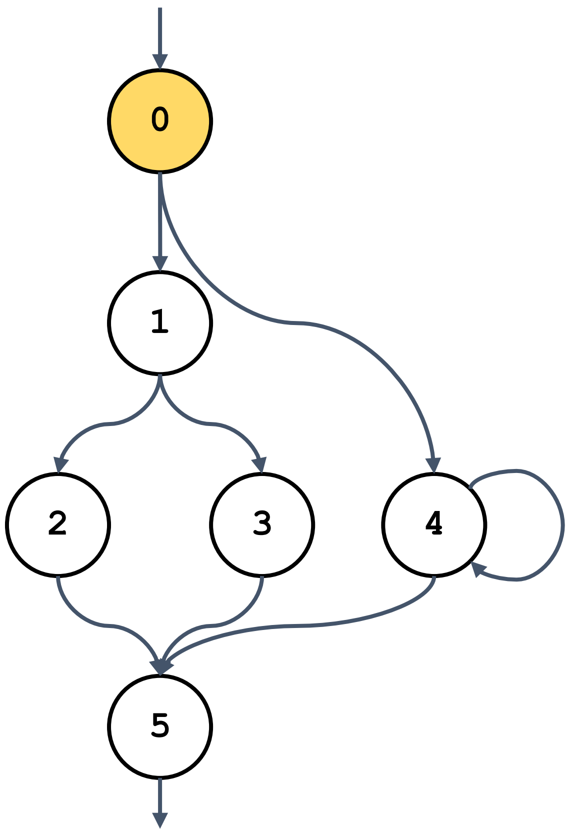
Solution


B0
a1 = Φ(a0, a2)
b1 = Φ(b0, b3)
c1 = Φ(c0, c5)
d1 = Φ(d0, d7)
e1 = Φ(e0, e4)
a2 = c1 + 2
d2 = a2 + b1
B1
c2 = Φ(c1, c3)
c3 = b1 - d2
B2
b2 = a2 + c1
B3
d3 = b2 * 2
e2 = 2 * 2
B4
d4 = b2 + 1
B5
d5 = Φ(d3, d4)
e3 = Φ(e1, e2)
c4 = d5 >> 4
B6
b3 = Φ(b1, b2)
c5 = Φ(c3, c4)
d6 = Φ(d2, d5)
e4 = Φ(e1, e3)
d7 = c5 + b3

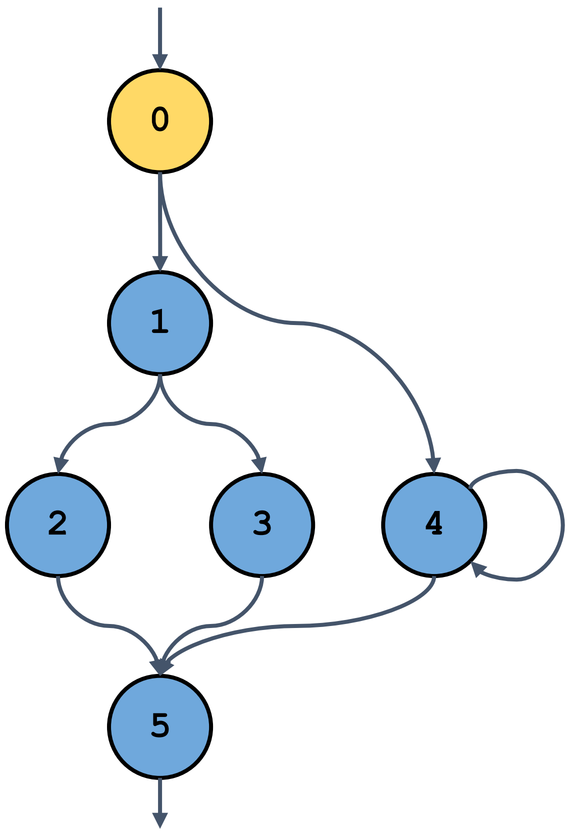
Problem 3



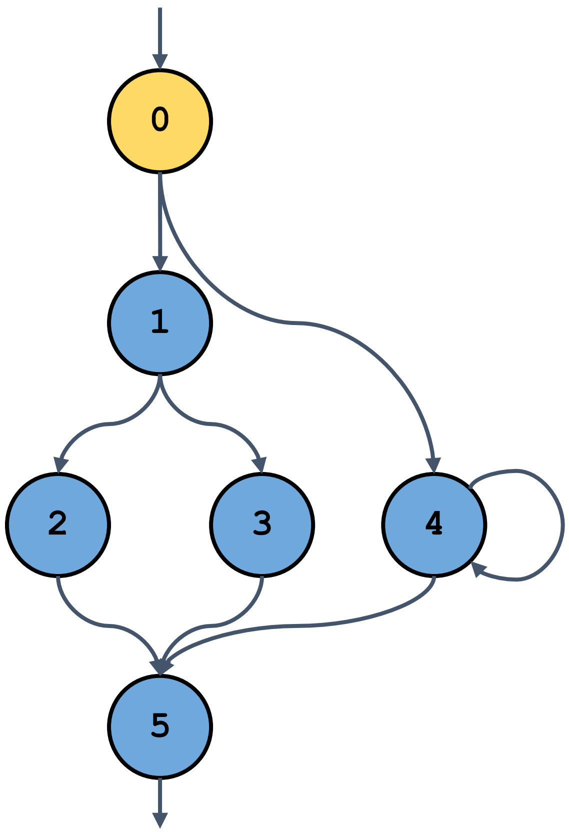
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0		
1		
2		
3		
4		
5		



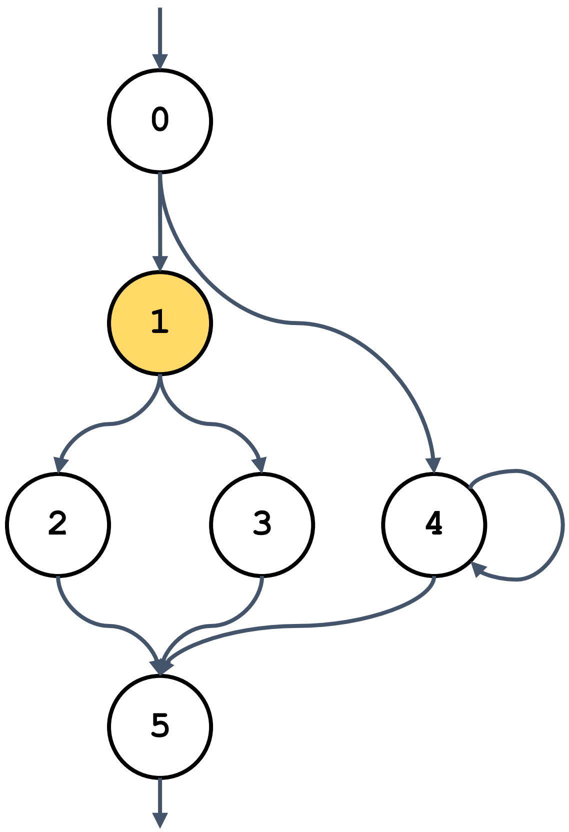
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0		
1		
2		
3		
4		
5		



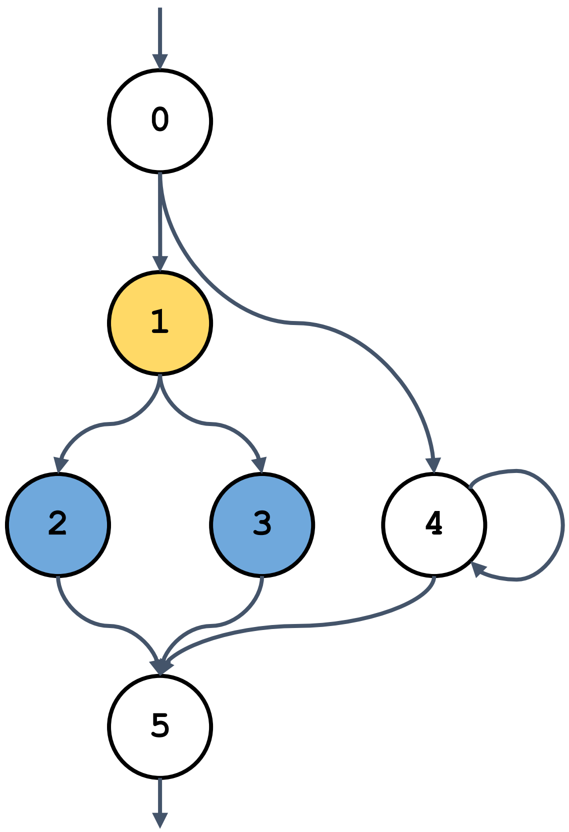
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5	
1		
2		
3		
4		
5		



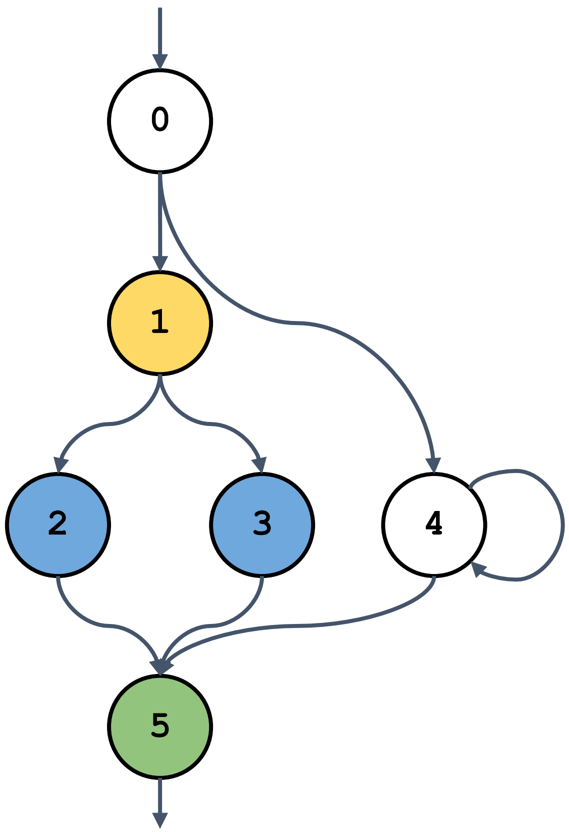
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5	Ø
1		
2		
3		
4		
5		



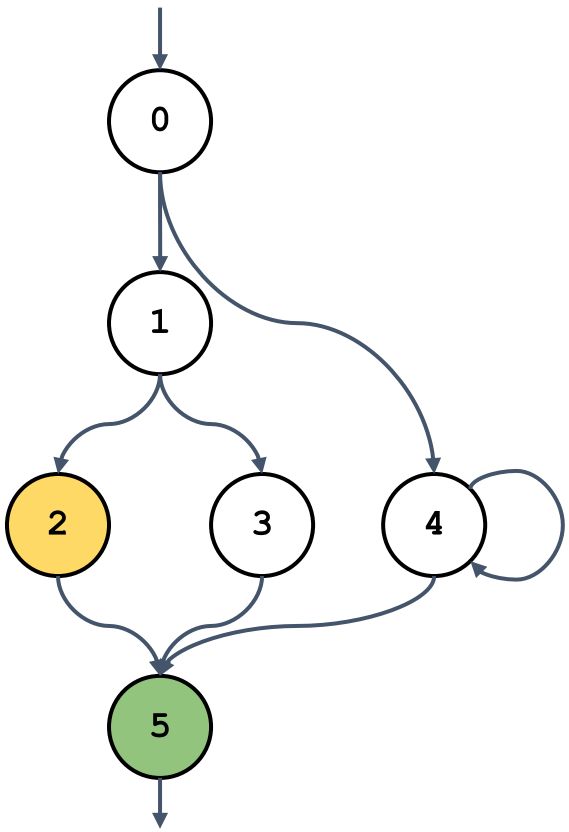
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5	∅
1		
2		
3		
4		
5		



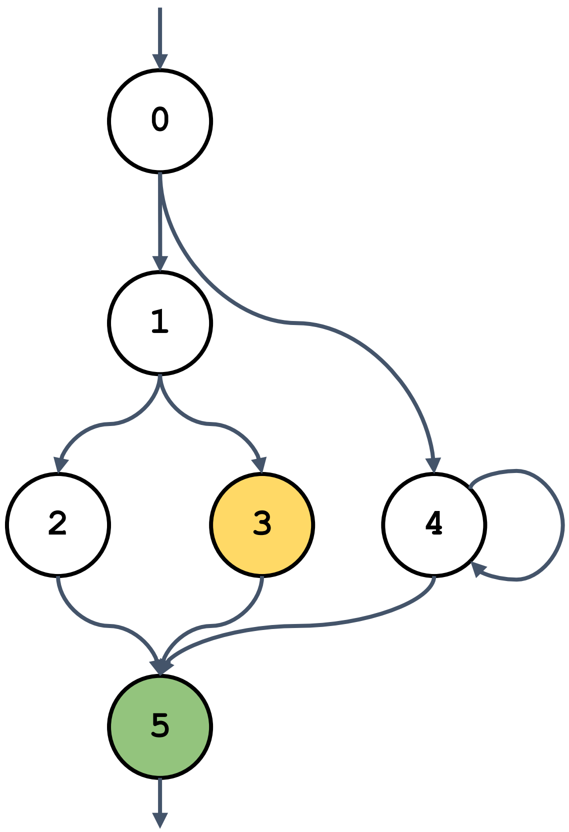
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5	∅
1	2, 3	
2		
3		
4		
5		



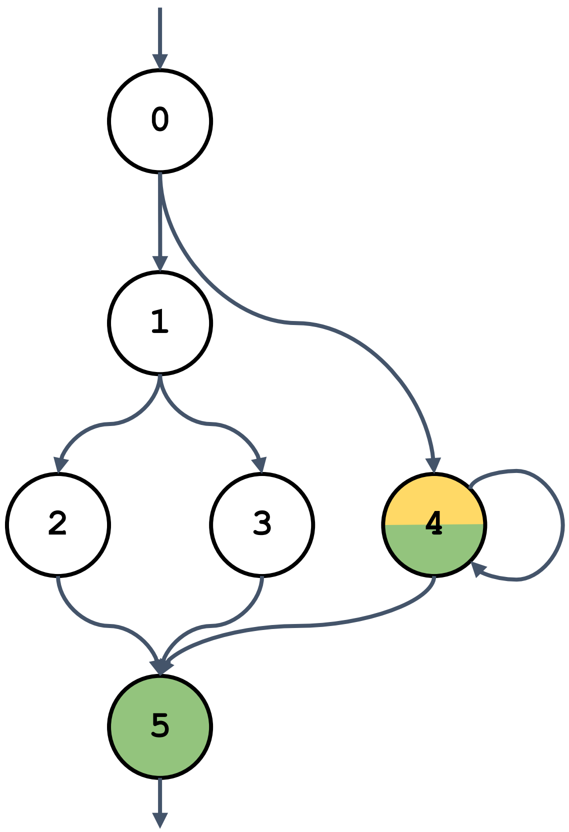
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5	∅
1	2, 3	5
2		
3		
4		
5		



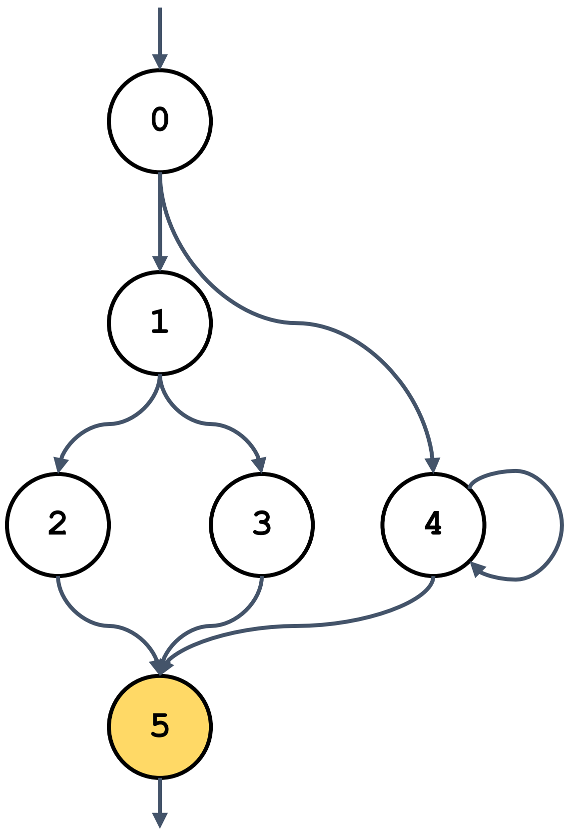
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5	∅
1	2, 3	5
2	∅	5
3		
4		
5		



NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5	∅
1	2, 3	5
2	∅	5
3	∅	5
4		
5		



NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5	∅
1	2, 3	5
2	∅	5
3	∅	5
4	∅	4, 5
5		



NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5	∅
1	2, 3	5
2	∅	5
3	∅	5
4	∅	4, 5
5	∅	∅