Single Static Assignment

CSE 401 Section 10/10

Aaron Johnston, Miya Natsuhara,
Kory Watson, Sam Wolfson

Adapted from Laura Vonesson's Wi17 Slides

The Final Stretch

You are here

SUN	MON	TUE	WED	THU	FRI	SAT
				 401 Compiler Additions		401 Report M501 Additions

M501 Report Evals!!	Review Session (4:30, MOR 230)	**Final Exam** (2:30)	**Eternal Mastery of Compilers**

Problem 1

(review of dataflow)

Single Static Assignment

An intermediate representation where each variable has only one definition:

Original

```
a := x + y
b := a - 1
a := y + b
b := x * 4
a := a + b
```

SSA Form

$$
\begin{aligned}
a_1 &:= x_1 + y_1 \\
b_1 &:= a_1 - 1 \\
a_2 &:= y_1 + b_1 \\
b_2 &:= x_1 * 4 \\
a_3 &:= a_2 + b_2
\end{aligned}
$$

SSA: Why We Love It

- Without SSA, all definitions and uses of a variable get mixed together
    - Computing information about the definitions of a variable is an expensive but necessary part of many dataflow analyses

SSA: Why We Love It

- Without SSA, all definitions and uses of a variable get mixed together
    - Computing information about the definitions of a variable is an expensive but necessary part of many dataflow analyses

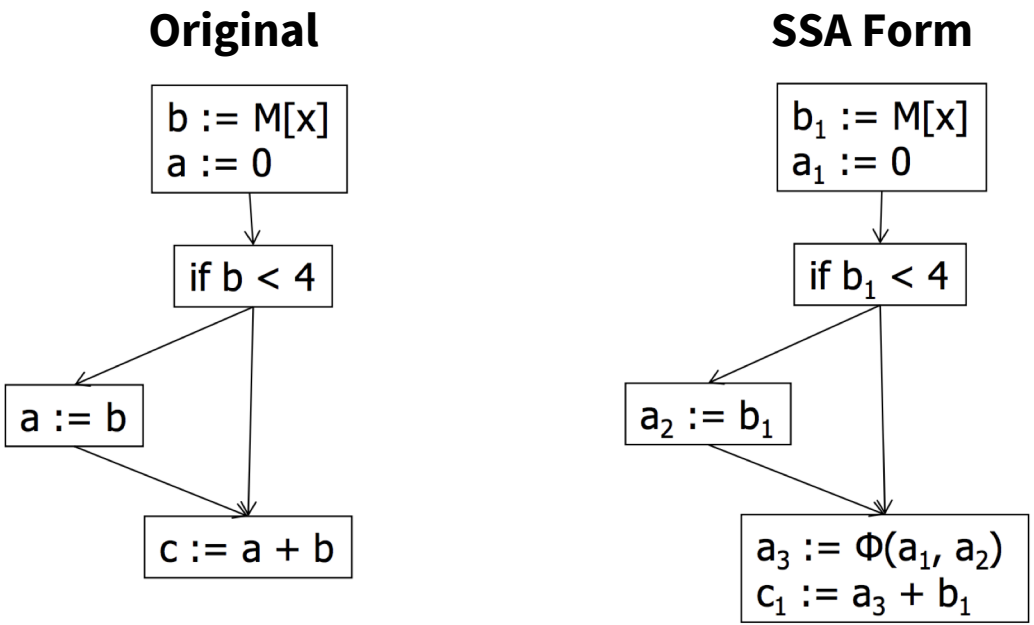
- Doing the work of converting to SSA once makes many analyses + optimizations more efficient
    - SSA can be thought of as an implicit representation of Definition/Use chains

SSA: Why We Love It

- Ex: Dead Store Elimination
  - Without SSA: Compute live variables at every point, which requires working backwards and using the dataflow sets to check for *any path* that does not kill the variable, and eliminate any stores that are not to a live variable.
  - With SSA: Eliminate any store where the variable being assigned has 0 uses.
- Constant Propagation is another optimization made much easier by SSA

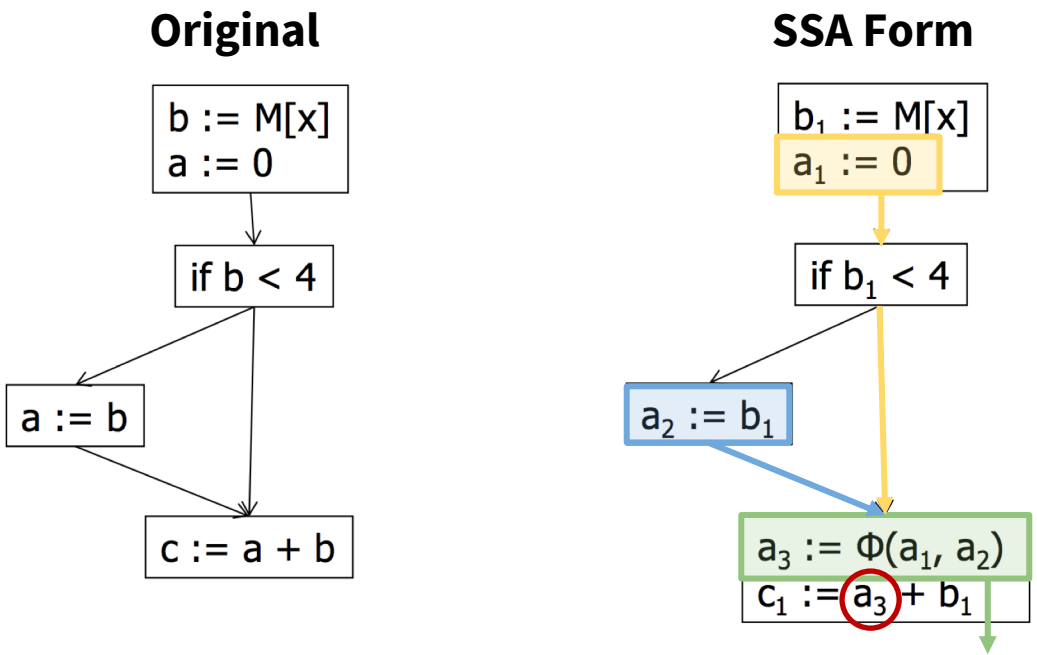
Phi-Functions

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



Where to place Phi-Functions?

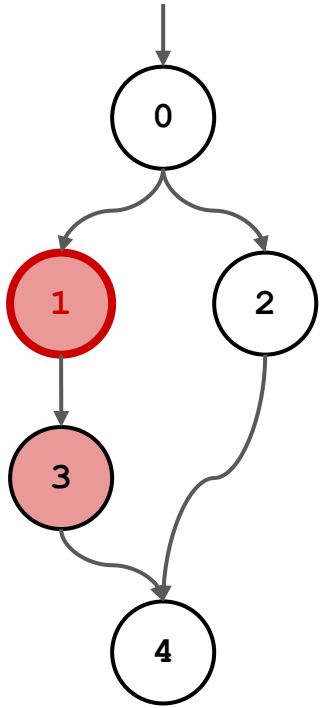
- Wherever a variable has multiple possible definitions entering a block
  - Inefficient (and unnecessary!) to consider all possible phi-functions at the start of each block



Dominators

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

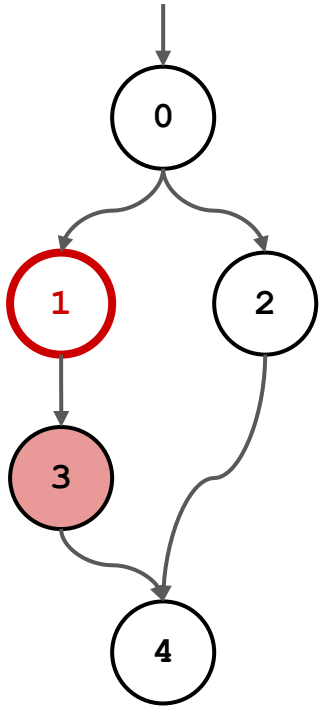
Node 1 dominates nodes 1 and 3. It does not dominate 4 because there is another path that reaches it.



Strict Dominance

- A node **`X`** ***strictly dominates*** a node **`Y`** iff **`X`** dominates **`Y`** and **`X ≠ Y`** .

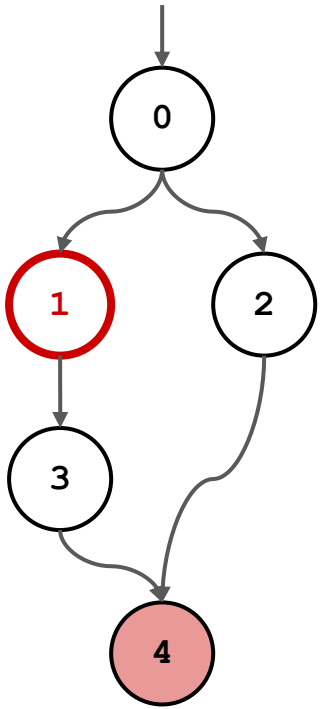
Node 1 only strictly dominates node 3 because it is the only dominated node that is not equal to 1.



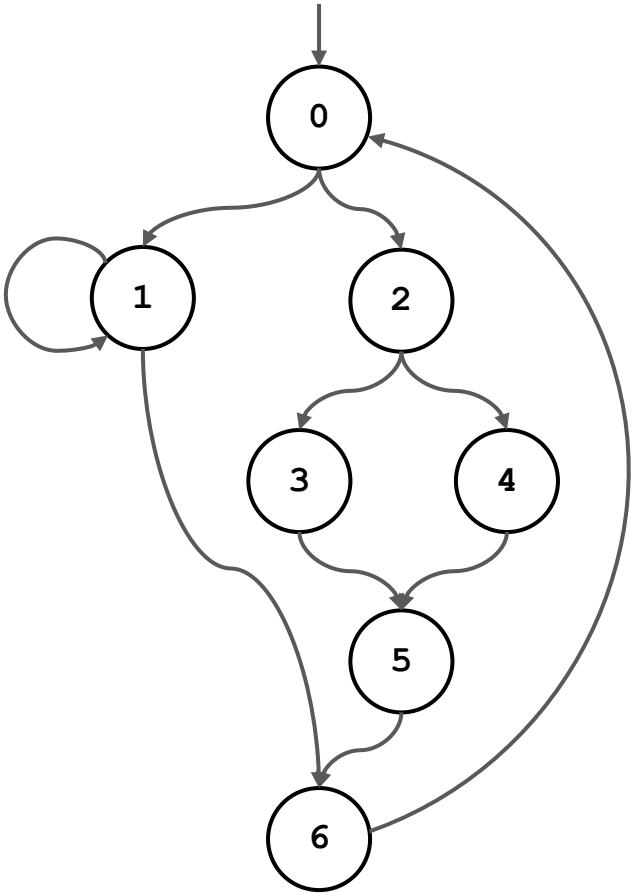
Dominance Frontiers

- 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` .
- Essentially, the border between dominated and non-dominated nodes
    - Note: a node can be in its own dominance frontier!
- This is where phi function merging is necessary

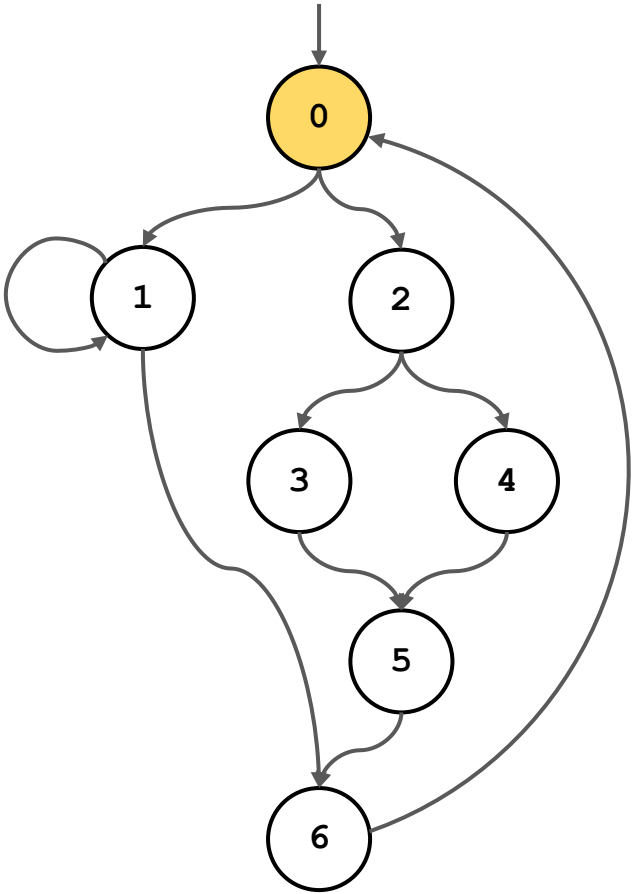
Node 4 is in the dominance frontier of node 1 because an immediate predecessor (node 3) is dominated by 1.



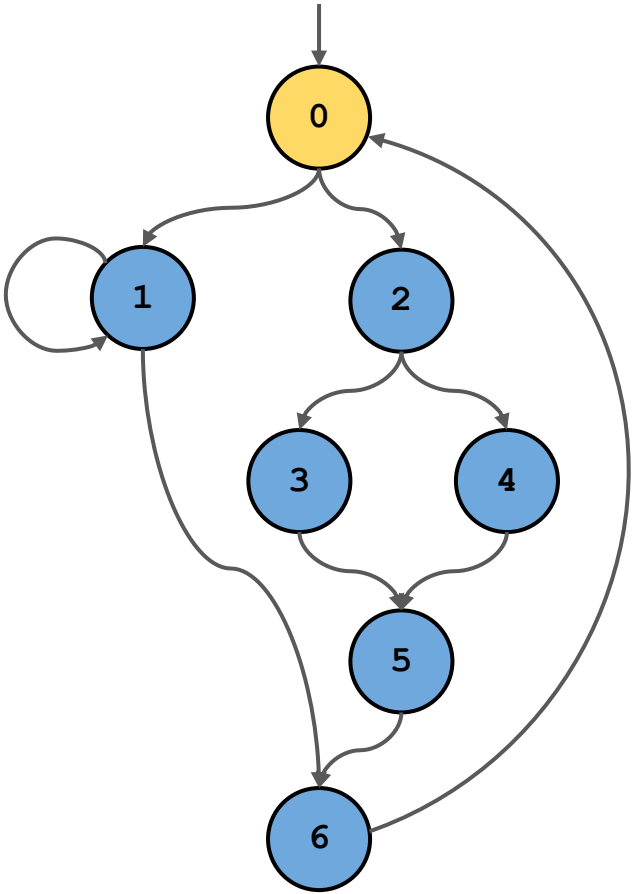
Problem 2(a)



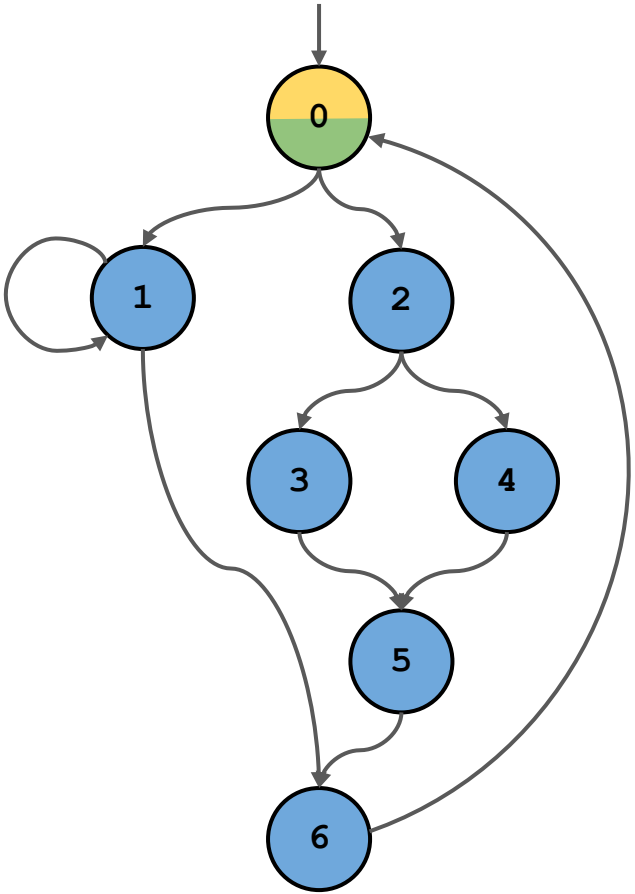
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0		
1		
2		
3		
4		
5		
6		



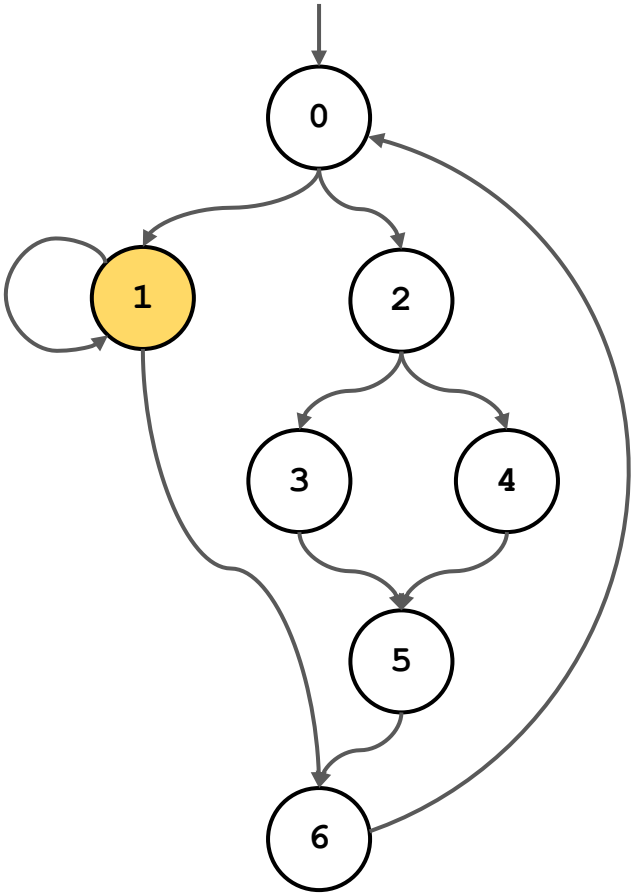
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0		
1		
2		
3		
4		
5		
6		



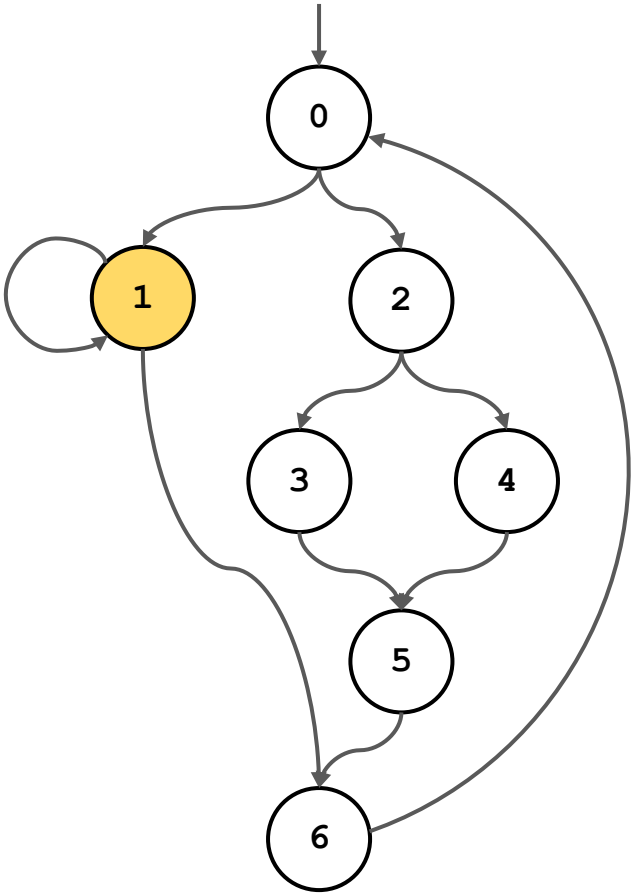
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5, 6	
1		
2		
3		
4		
5		
6		



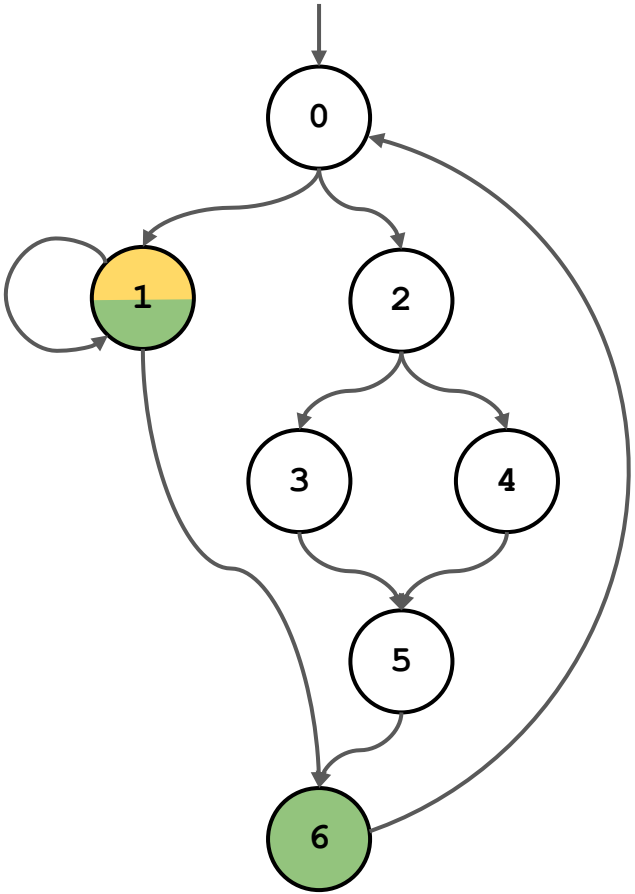
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5, 6	0
1		
2		
3		
4		
5		
6		



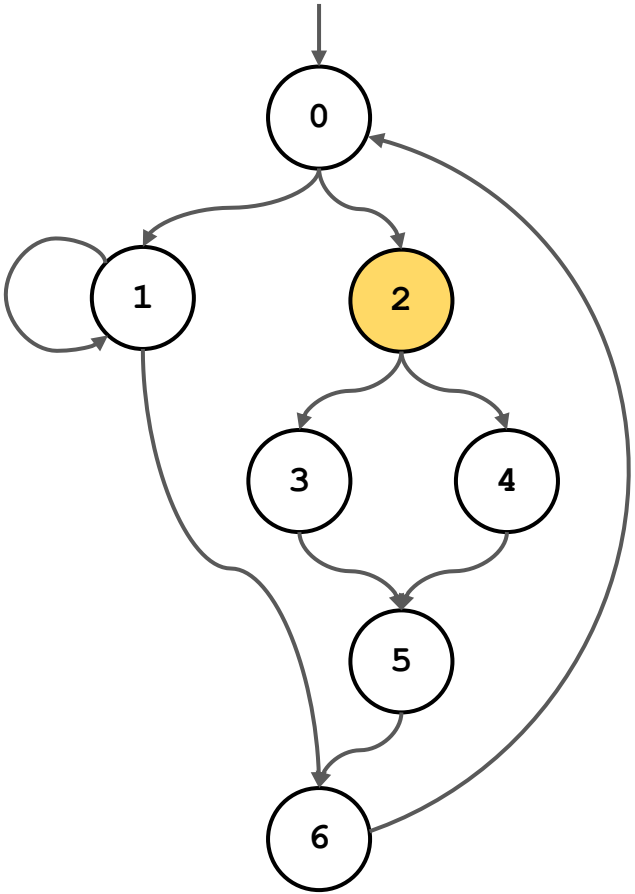
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5, 6	0
1		
2		
3		
4		
5		
6		



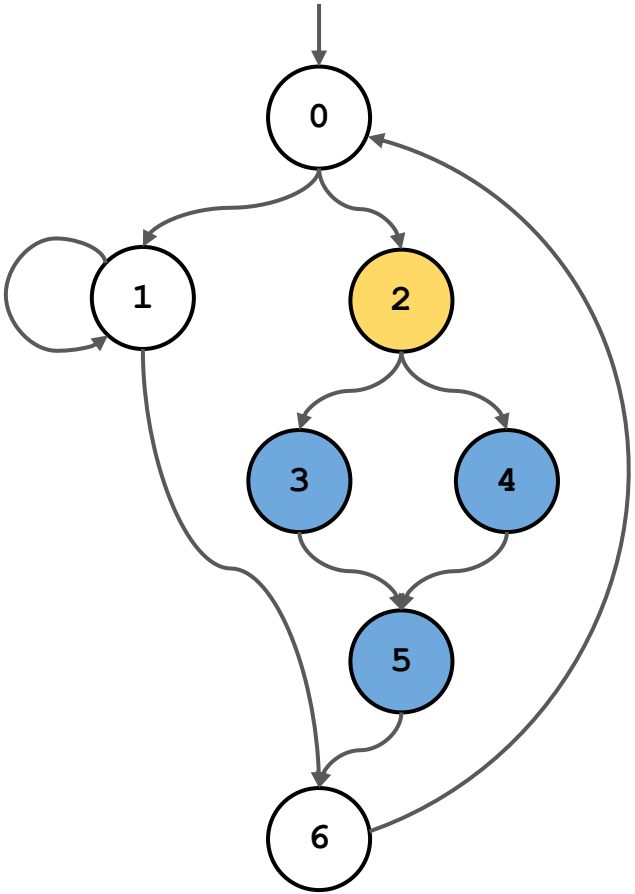
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5, 6	0
1	∅	
2		
3		
4		
5		
6		



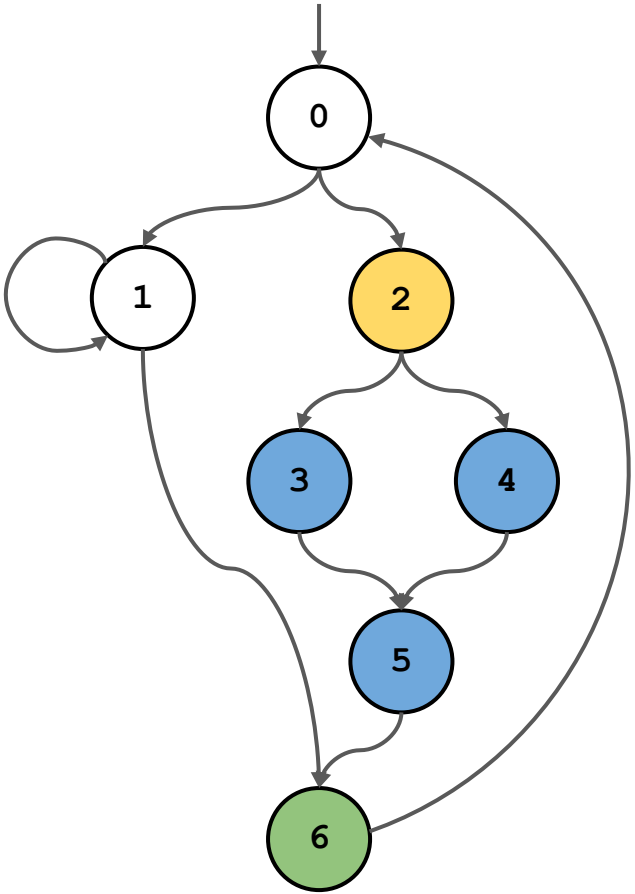
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5, 6	0
1	∅	1, 6
2		
3		
4		
5		
6		



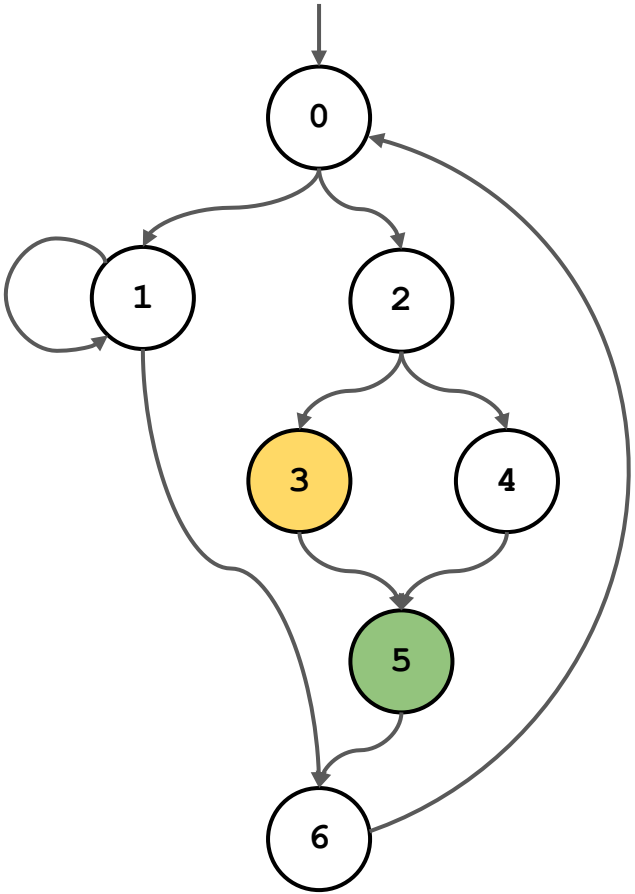
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5, 6	0
1	∅	1, 6
2		
3		
4		
5		
6		



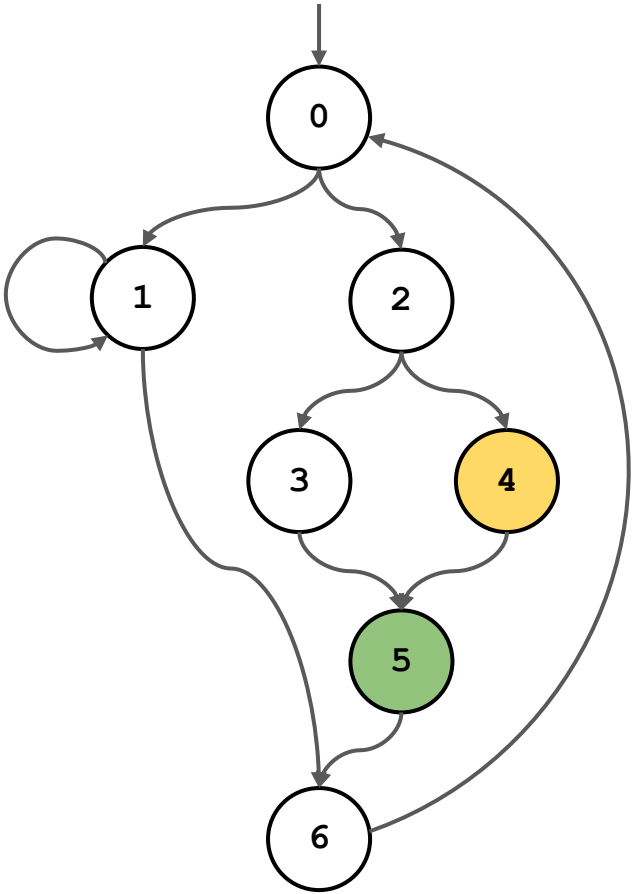
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5, 6	0
1	∅	1, 6
2	**3, 4, 5**	
3		
4		
5		
6		



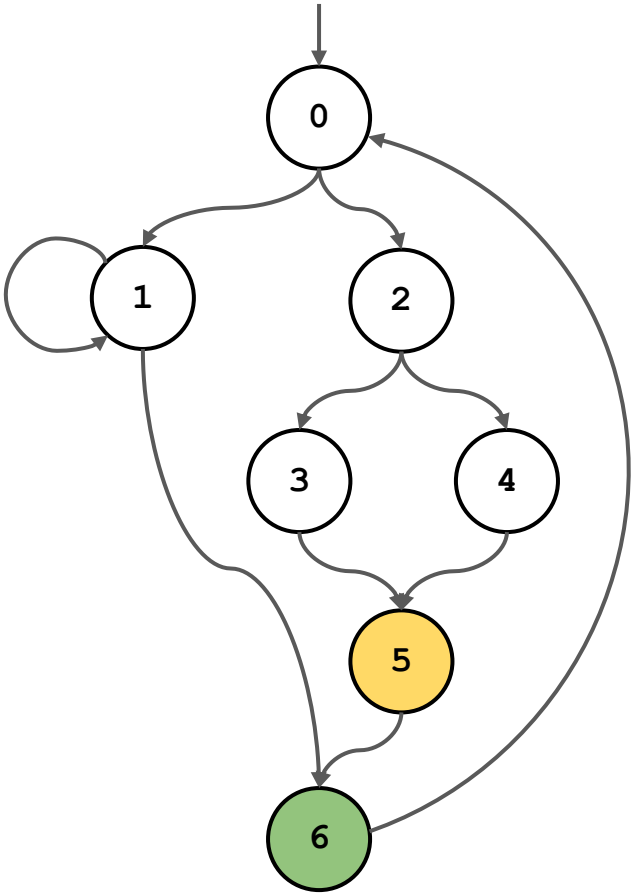
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5, 6	0
1	∅	1, 6
2	**3, 4, 5**	**6**
3		
4		
5		
6		



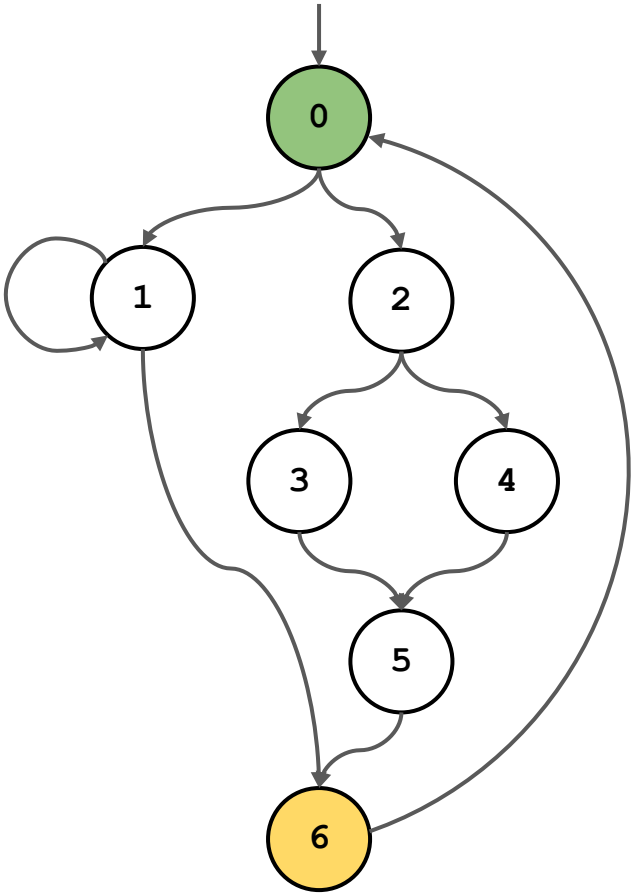
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5, 6	0
1	∅	1, 6
2	3, 4, 5	6
3	∅	5
4		
5		
6		



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**
5		
6		

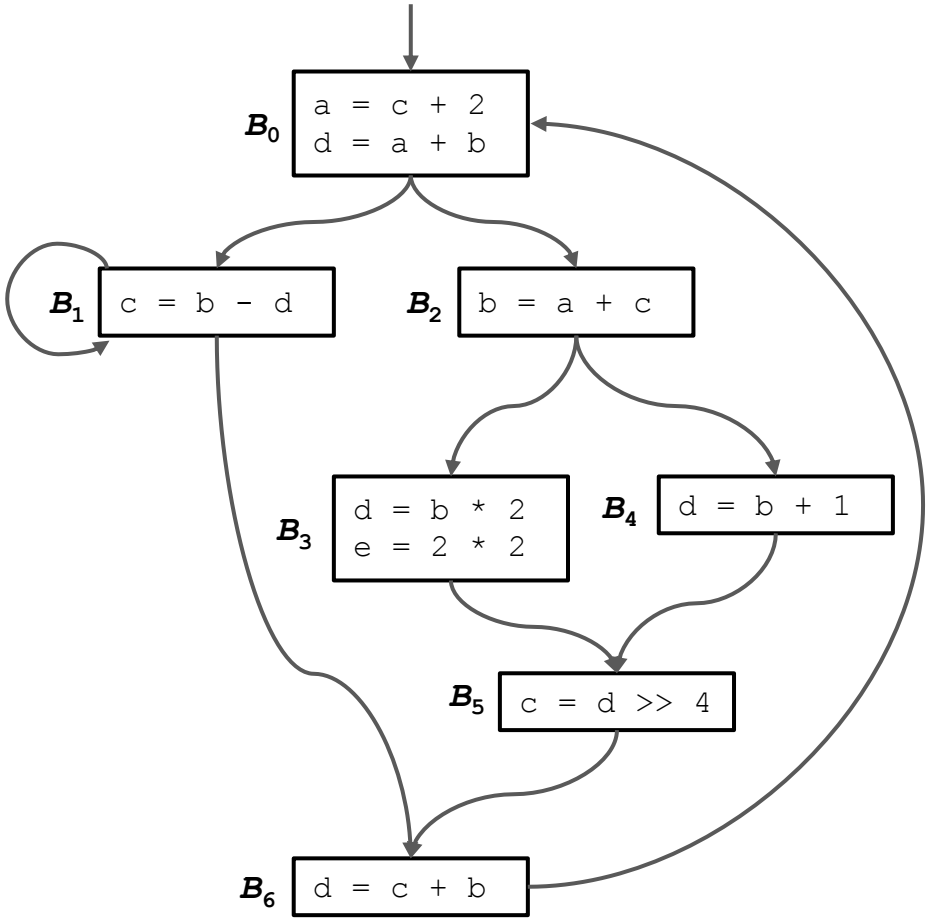


NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5, 6	0
1	∅	1, 6
2	3, 4, 5	6
3	∅	5
4	∅	5
5	∅	6
6		

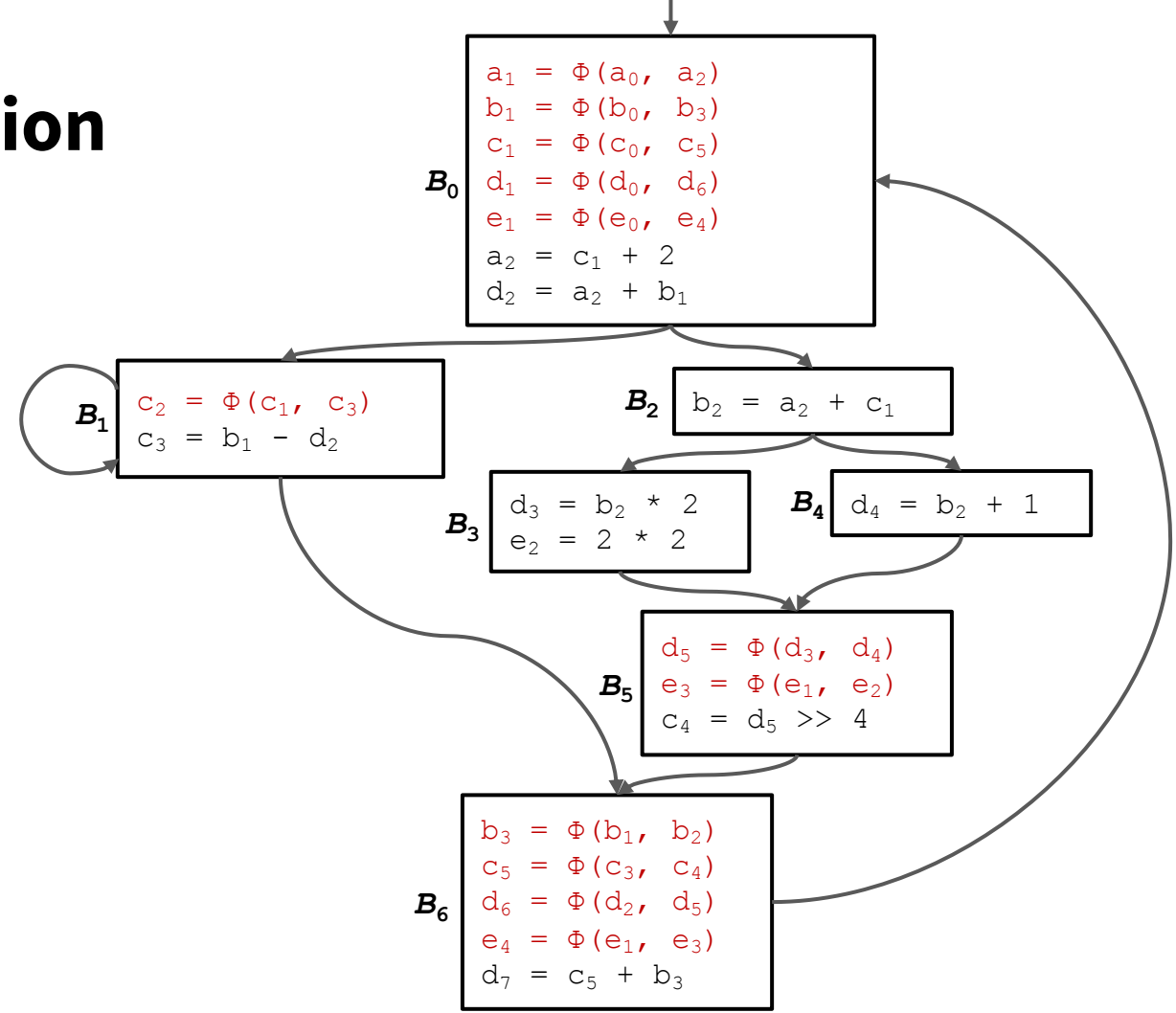


NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5, 6	0
1	∅	1, 6
2	3, 4, 5	6
3	∅	5
4	∅	5
5	∅	6
6	**∅**	**0**

Problem 2(b)

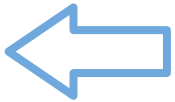

B0
a = c + 2
d = a + b
B1
c = b - d
B2
b = a + c
B3
d = b * 2
e = 2 * 2
B4
d = b + 1
B5
c = d >> 4
B6
d = c + b

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**

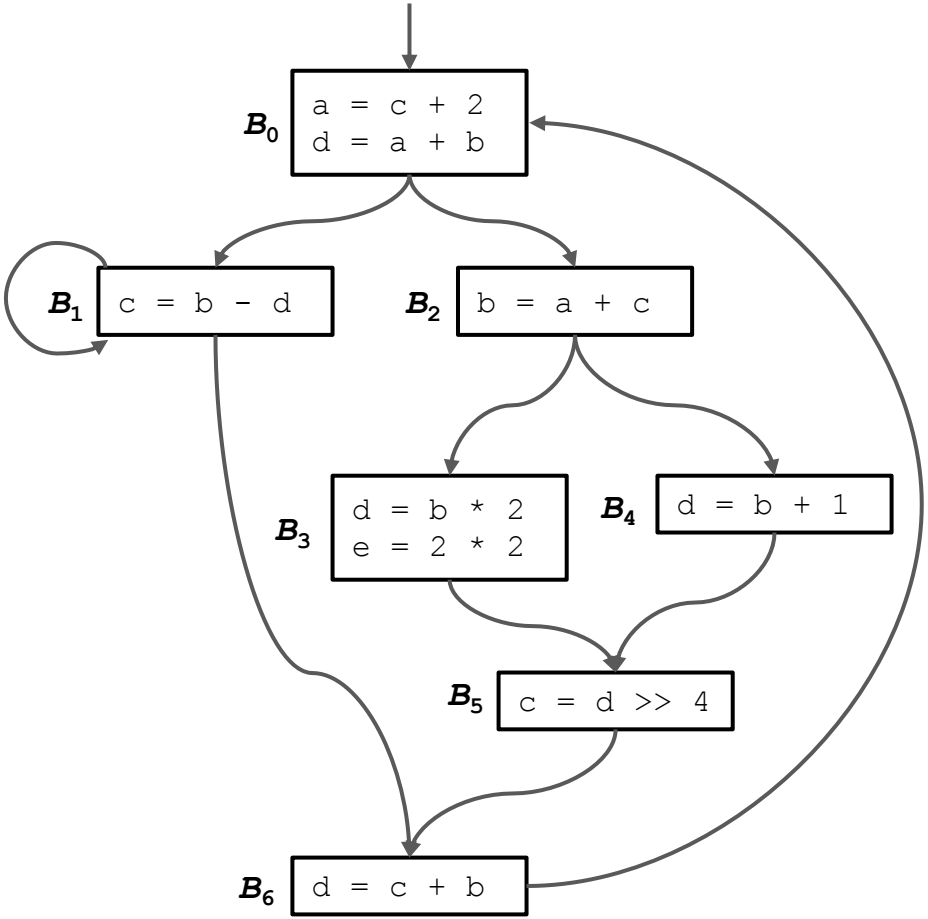


Already done (in problem 2a)

2. Determine which variables need merging in each node

3. Assign numbers to definitions and add phi functions

Step 1: Dominance Frontiers

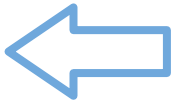


NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5, 6	0
1	Ø	1, 6
2	3, 4, 5	6
3	Ø	5
4	Ø	5
5	Ø	6
6	Ø	0

Converting to SSA

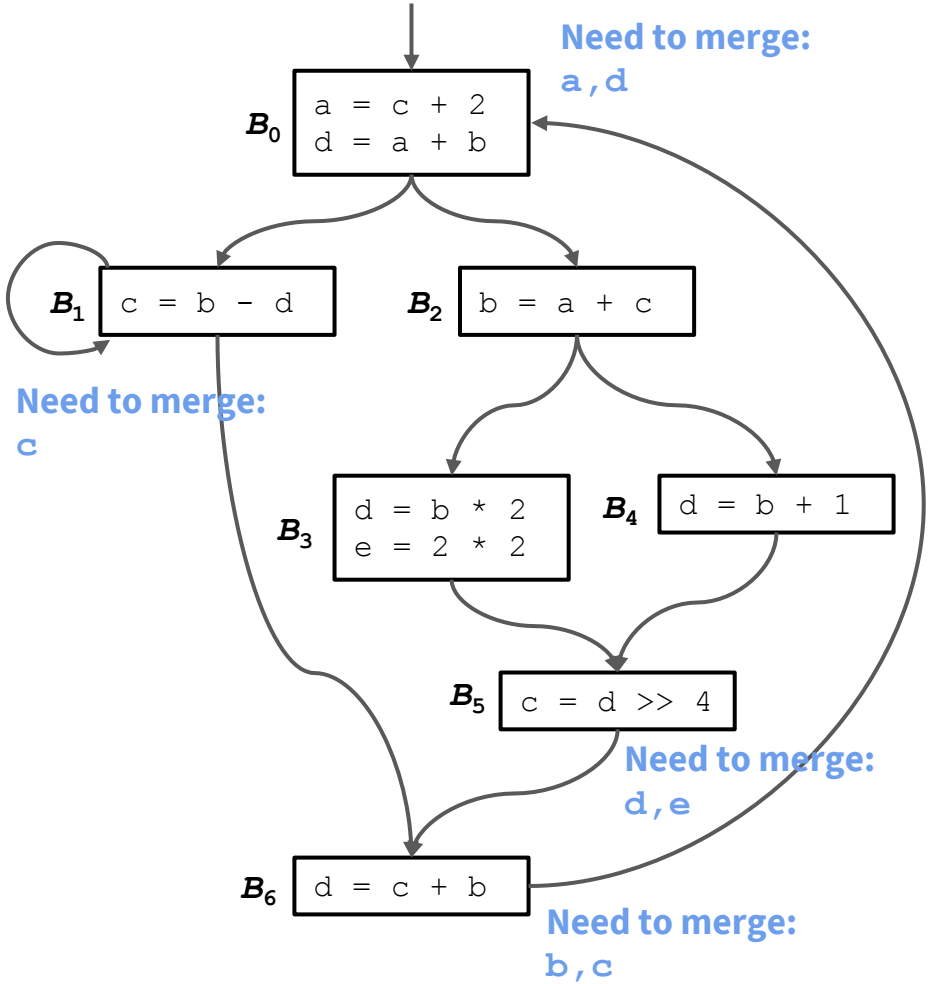
1. Compute the dominance frontier of each node

2. **Determine which variables need merging in each node**



We will compute using the dominance frontiers

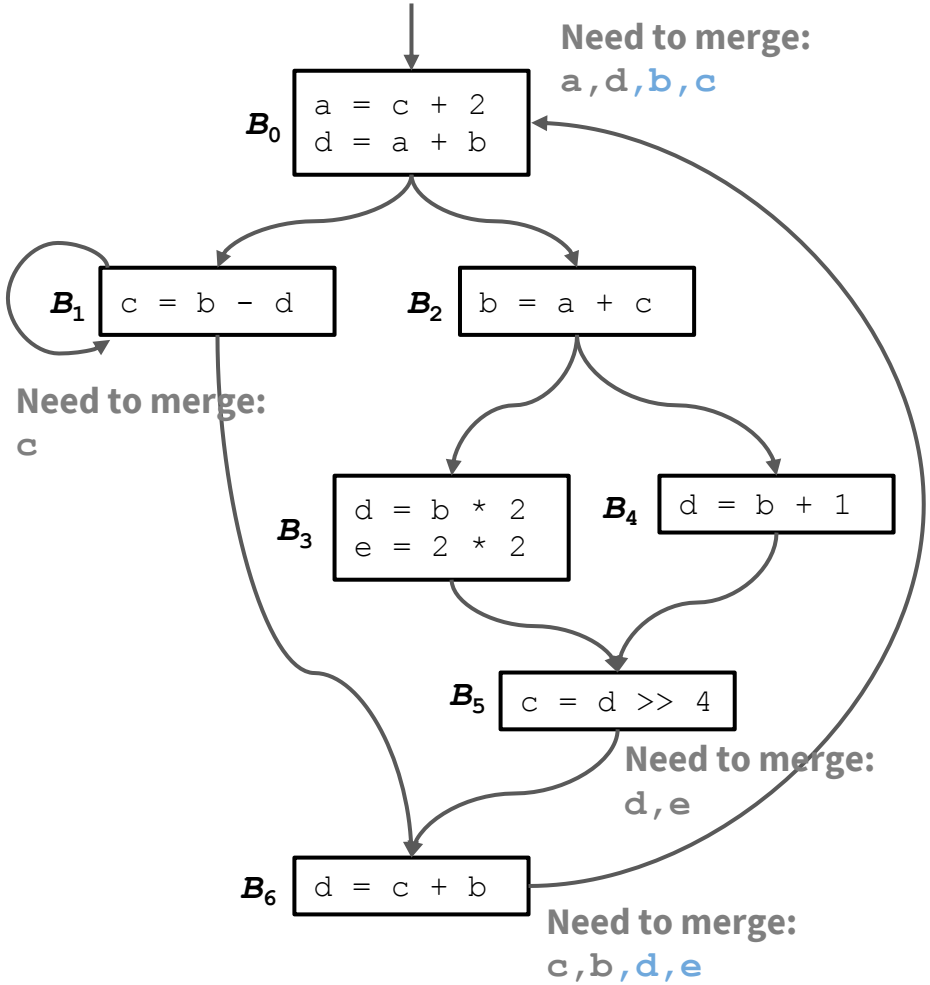
3. Assign numbers to definitions and add phi functions



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



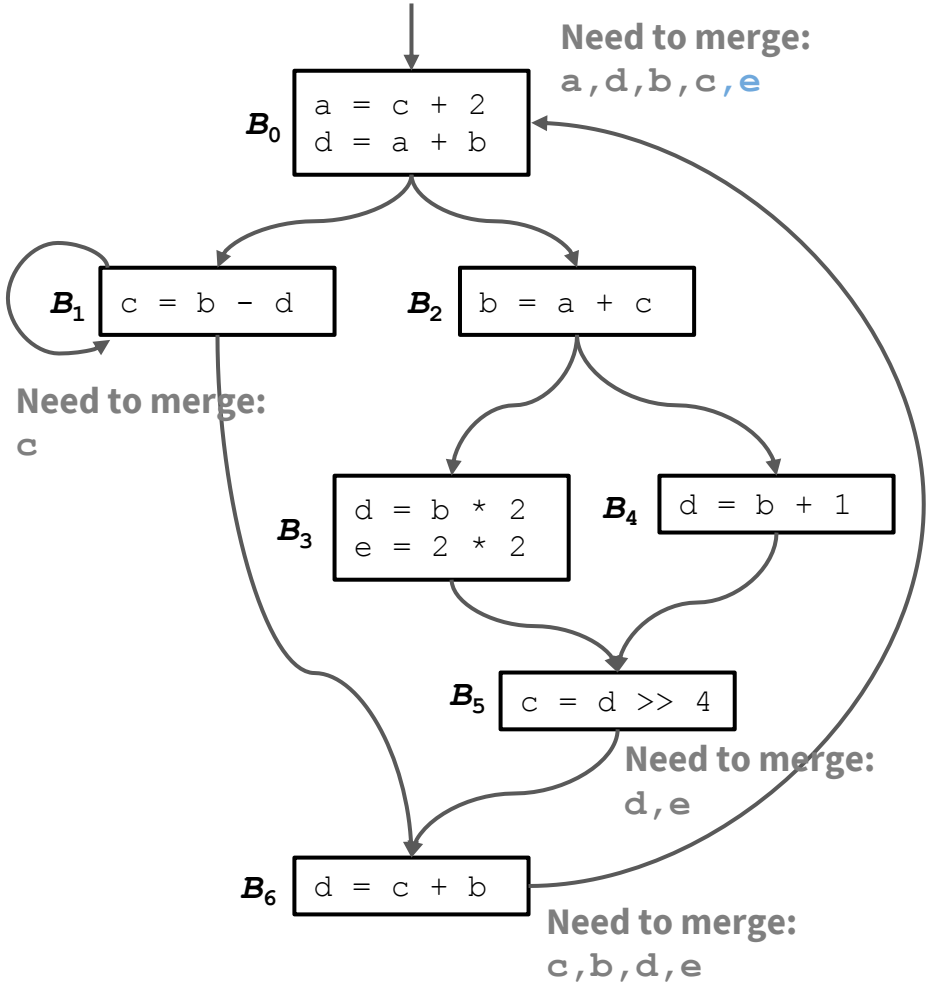
Step 2: Determine Necessary Merges

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	6
6	0

d,e

b,c



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	0

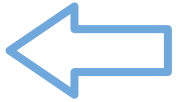
d,e

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

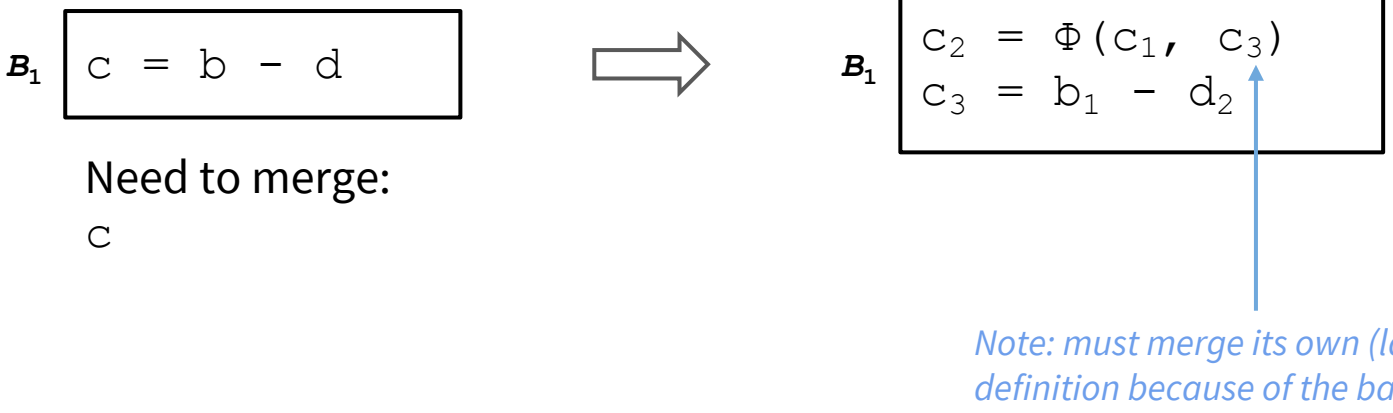
B_0

$a_1 = \Phi(a_0, a_2)$
$b_1 = \Phi(b_0, b_3)$
$c_1 = \Phi(c_0, c_5)$
$d_1 = \Phi(d_0, d_6)$
$e_1 = \Phi(e_0, e_4)$
$a_2 = c_1 + 2$
$d_2 = a_2 + b_1$

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

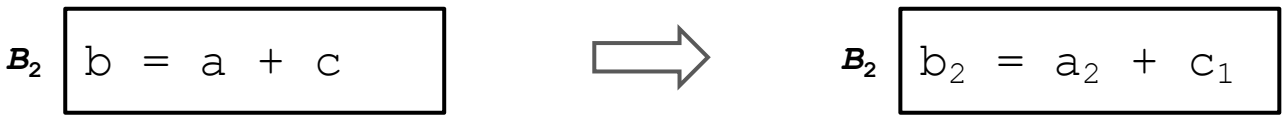
Step 3: Assign Definition Numbers

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



Step 3: Assign Definition Numbers

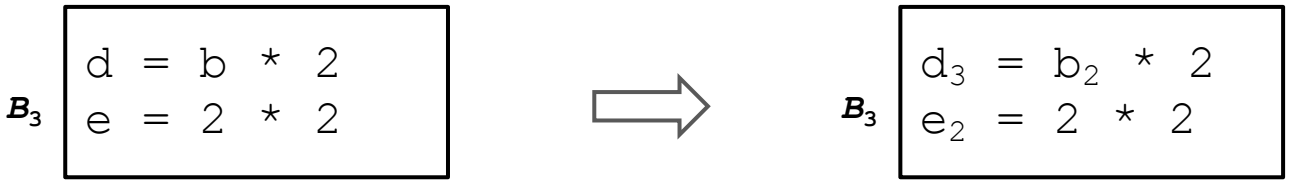
Merges go first, and each successive definition of a variable should increment its index by 1.



Nothing to merge

Step 3: Assign Definition Numbers

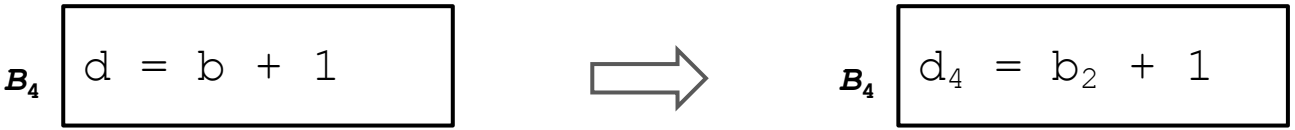
Merges go first, and each successive definition of a variable should increment its index by 1.



Nothing to merge

Step 3: Assign Definition Numbers

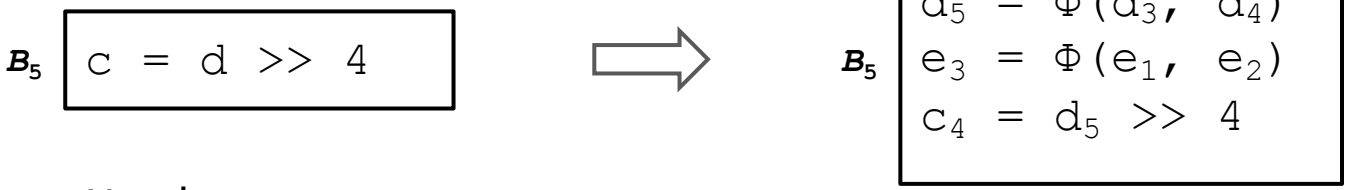
Merges go first, and each successive definition of a variable should increment its index by 1.



Nothing to merge

Step 3: Assign Definition Numbers

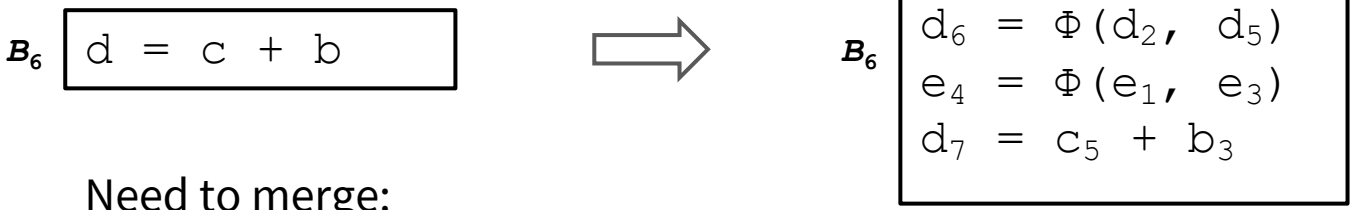
Merges go first, and each successive definition of a variable should increment its index by 1.



Need to merge:
d, e

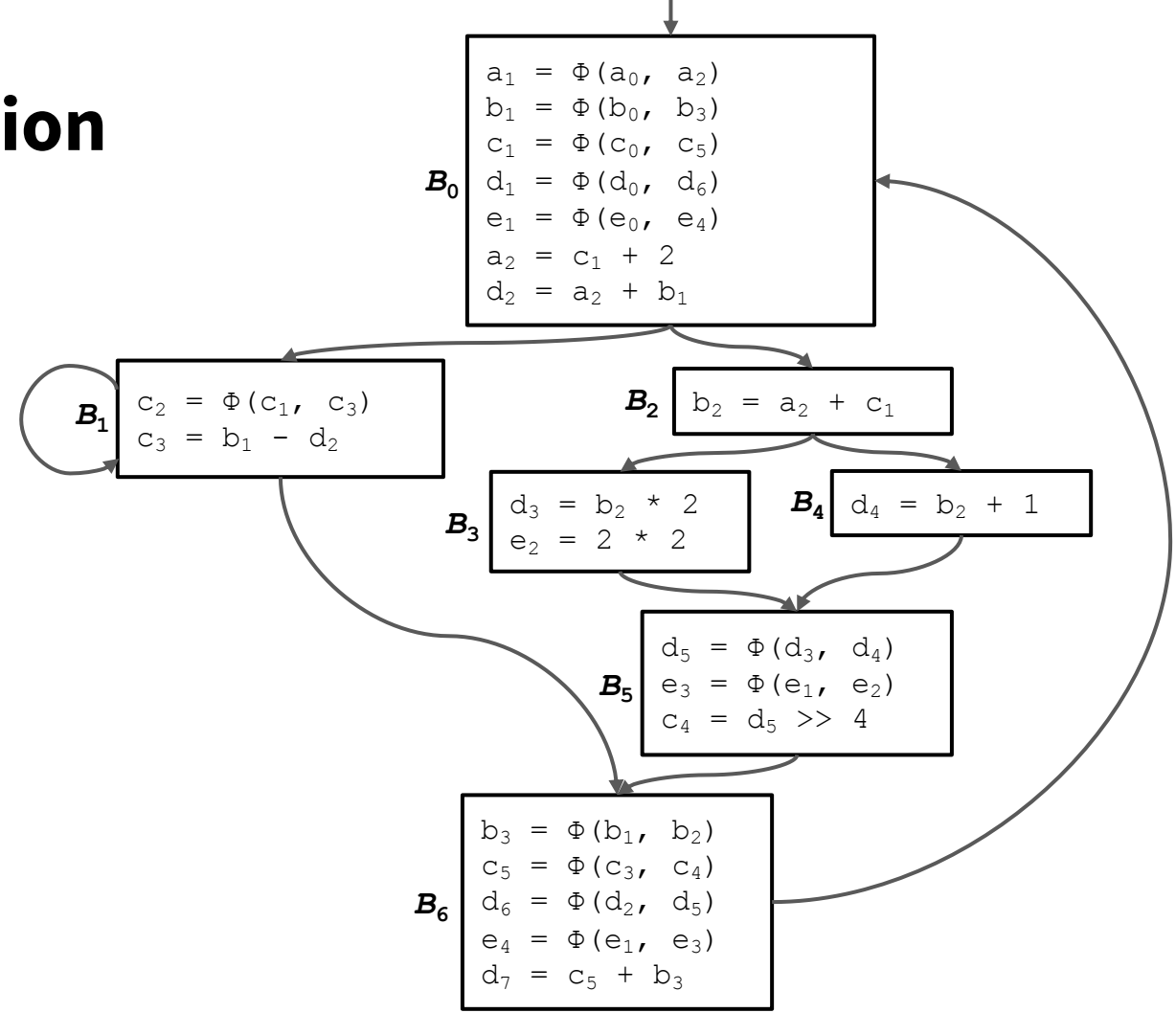
Step 3: Assign Definition Numbers

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



Need to merge:
c,e,b,i,d,g

Solution

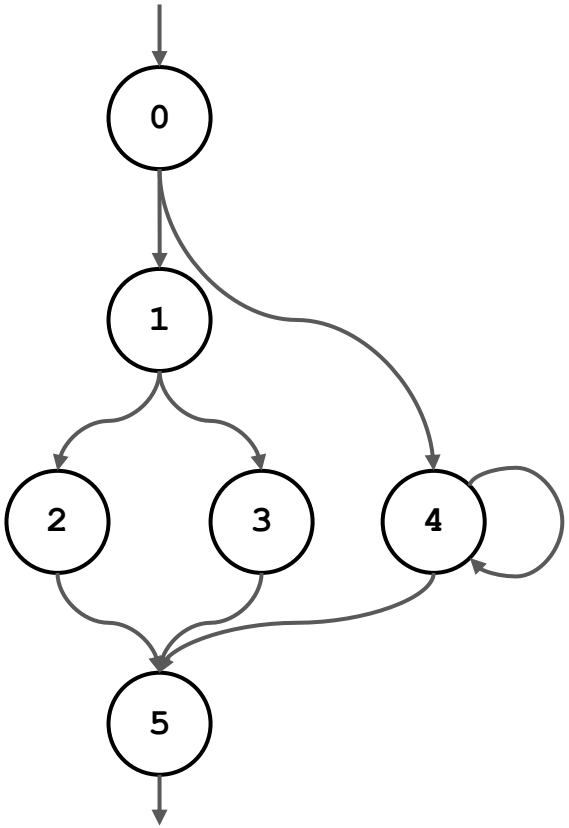

B_0
$a_1 = \Phi(a_0, a_2)$
$b_1 = \Phi(b_0, b_3)$
$c_1 = \Phi(c_0, c_5)$
$d_1 = \Phi(d_0, d_6)$
$e_1 = \Phi(e_0, e_4)$
$a_2 = c_1 + 2$
$d_2 = a_2 + b_1$
B_1
$c_2 = \Phi(c_1, c_3)$
$c_3 = b_1 - d_2$
B_2
$b_2 = a_2 + c_1$
B_3
$d_3 = b_2 * 2$
$e_2 = 2 * 2$
B_4
$d_4 = b_2 + 1$
B_5
$d_5 = \Phi(d_3, d_4)$
$e_3 = \Phi(e_1, e_2)$
$c_4 = d_5 >> 4$
B_6
$b_3 = \Phi(b_1, b_2)$
$c_5 = \Phi(c_3, c_4)$
$d_6 = \Phi(d_2, d_5)$
$e_4 = \Phi(e_1, e_3)$
$d_7 = c_5 + b_3$

Thanks for a Great Quarter!

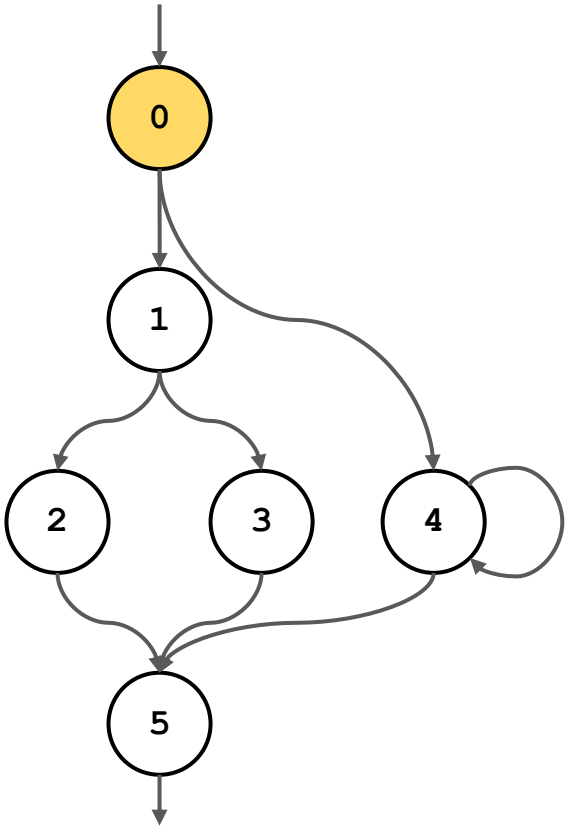
- The 401 19au Staff :)

Extra Practice:

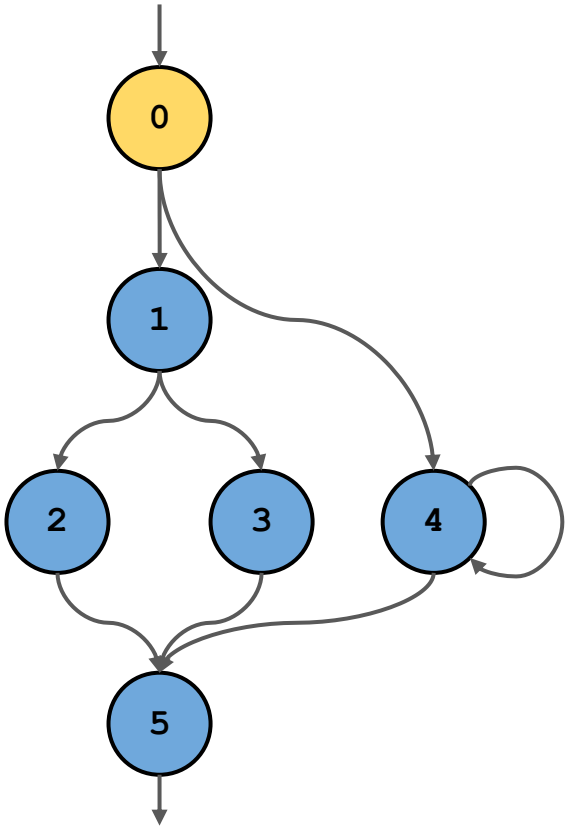
Problem 3



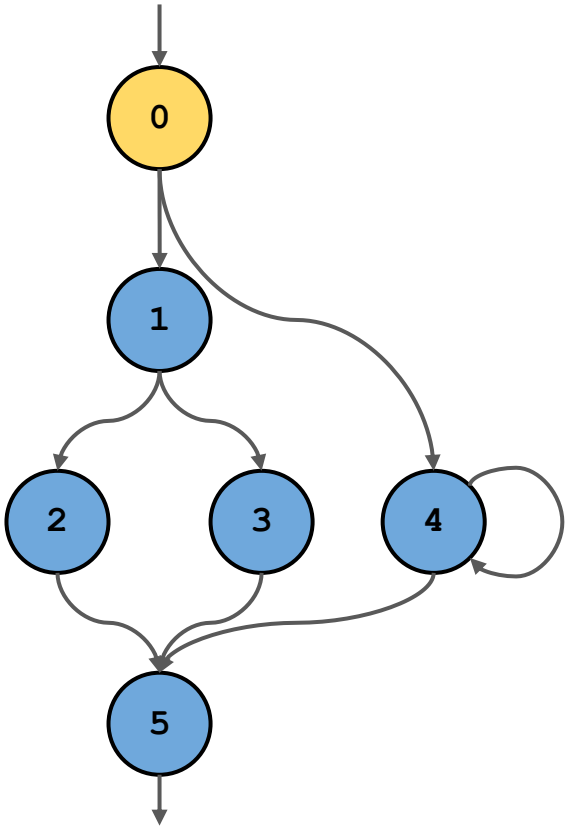
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0		
1		
2		
3		
4		
5		



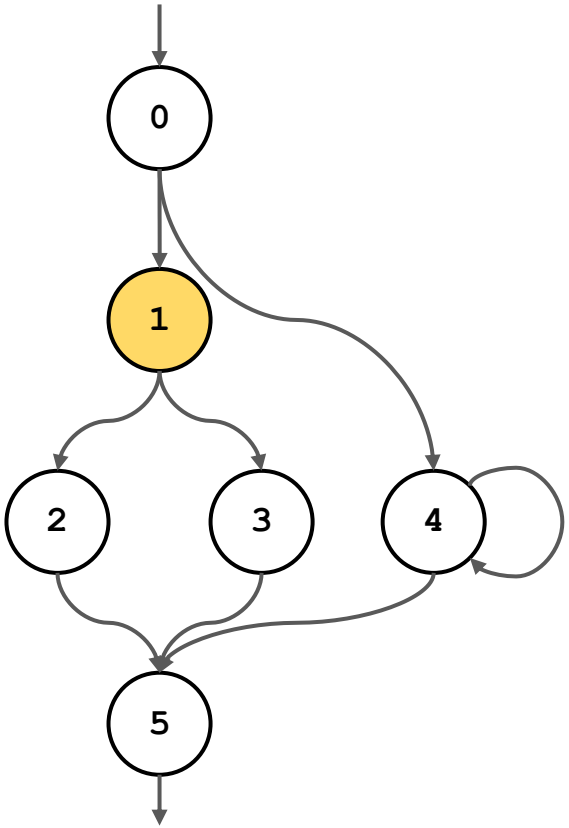
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0		
1		
2		
3		
4		
5		



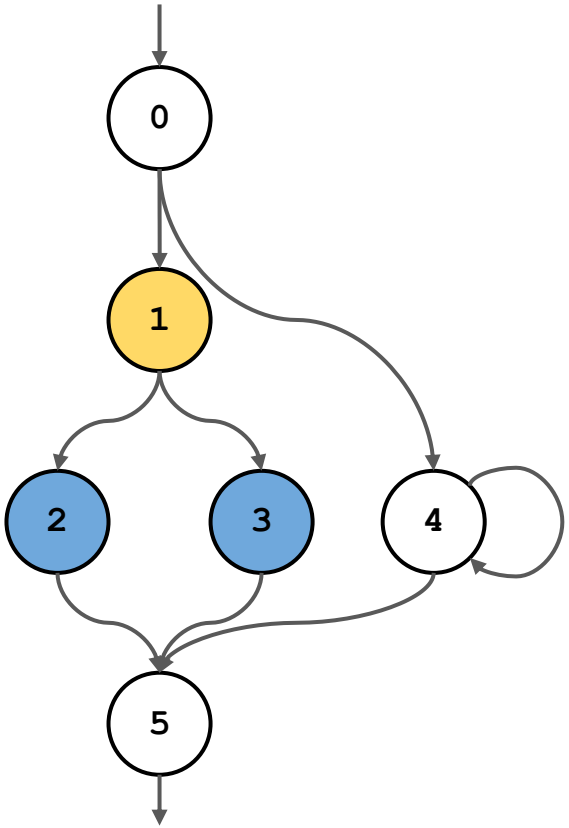
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5	
1		
2		
3		
4		
5		



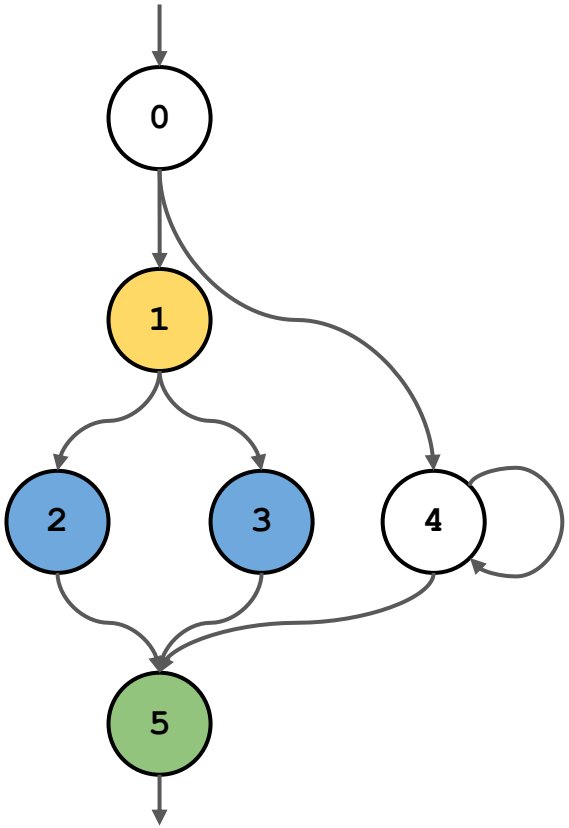
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5	Ø
1		
2		
3		
4		
5		



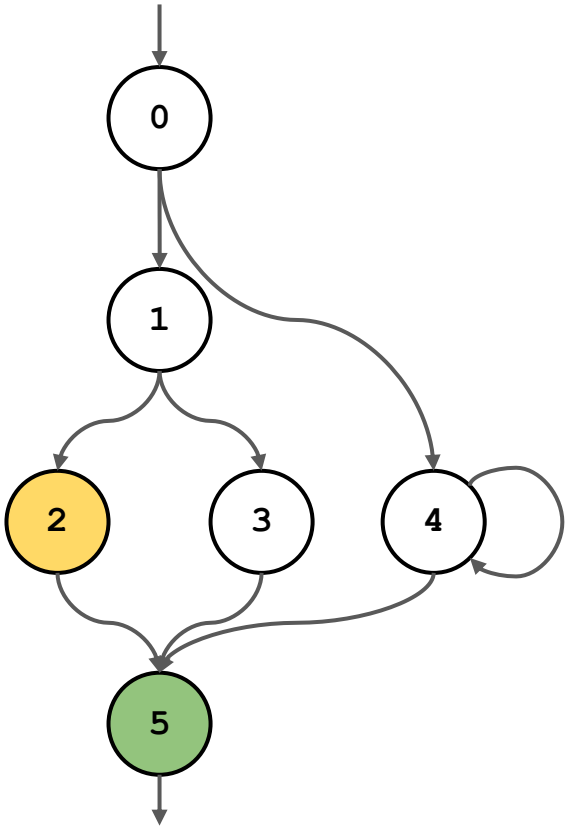
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5	Ø
1		
2		
3		
4		
5		



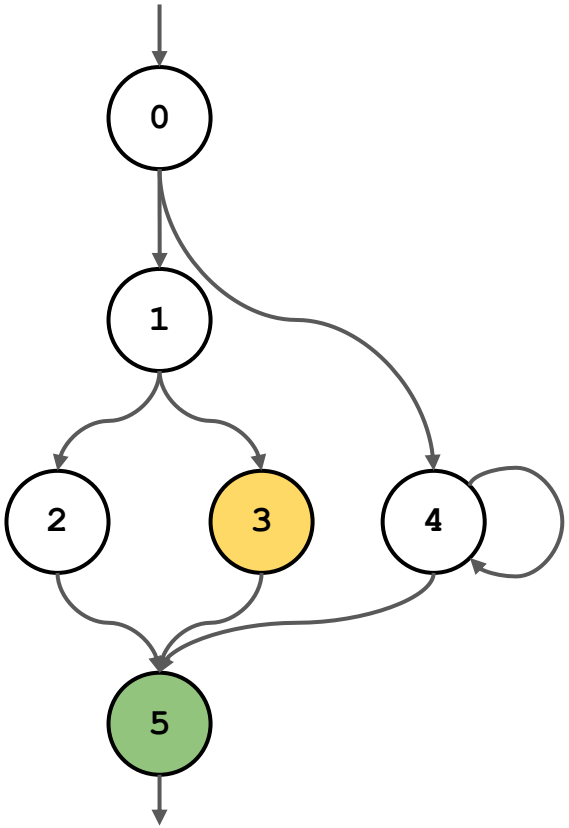
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5	Ø
1	2, 3	
2		
3		
4		
5		



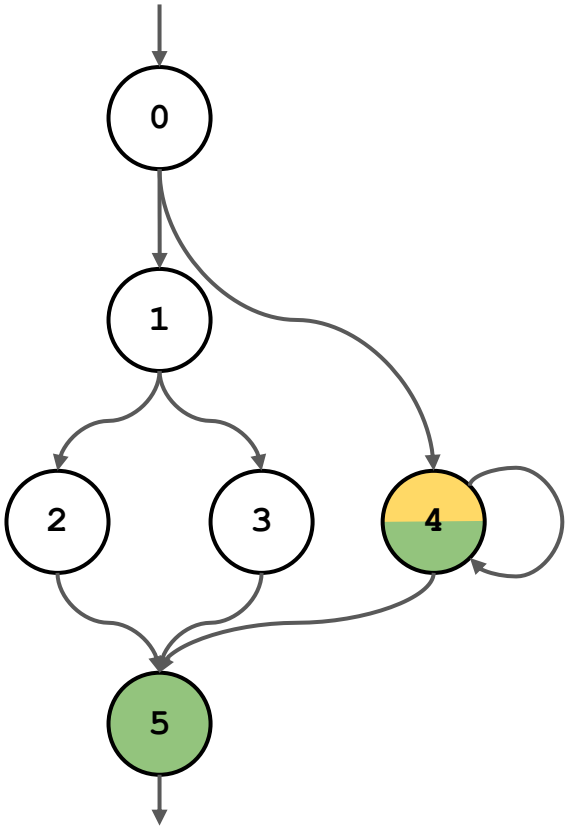
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5	Ø
1	2, 3	5
2		
3		
4		
5		



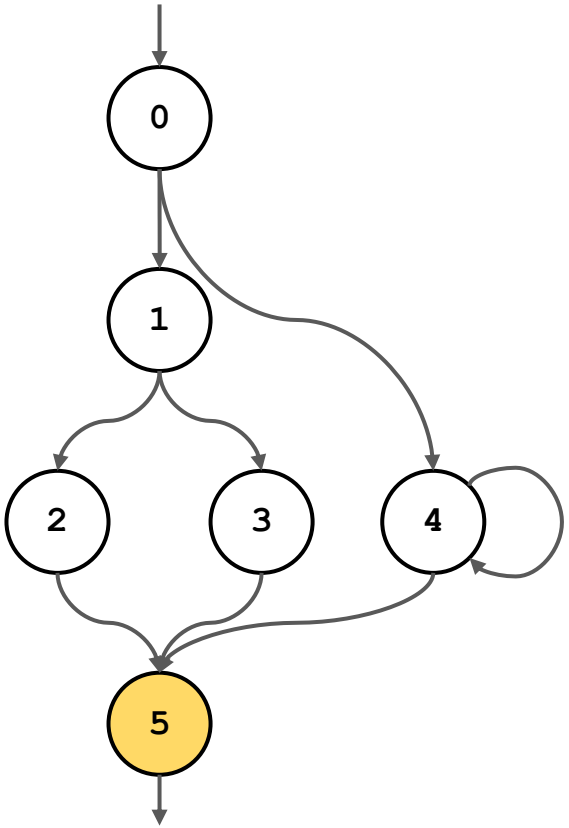
NODE	STRICTLY DOMINATES	DOMINANCE FRONTIER
0	1, 2, 3, 4, 5	Ø
1	2, 3	5
2	Ø	5
3		
4		
5		



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



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



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