CS & IT Engineering

Compiler Design

Intermediate Code & Code Optimization

Deva sir

Lecture: 4

Topics to be covered:

- Live Variable Analysis [Backward]
  - GEN & KILL Sets for each BB
  - IN & OUT Sets for each BB using Backward Analysis
- Reaching Definitions Analysis [Forward Analysis]
- Other topics: Code Generation
  Run Time Environments
- Assignment Questions.

Live Variable Analysis:

GEN (USE) Set

BB_k:

$x = a + b$

$y = x * c$

$z = y - d$

$GEN_k = \{a, b, c, d\}$

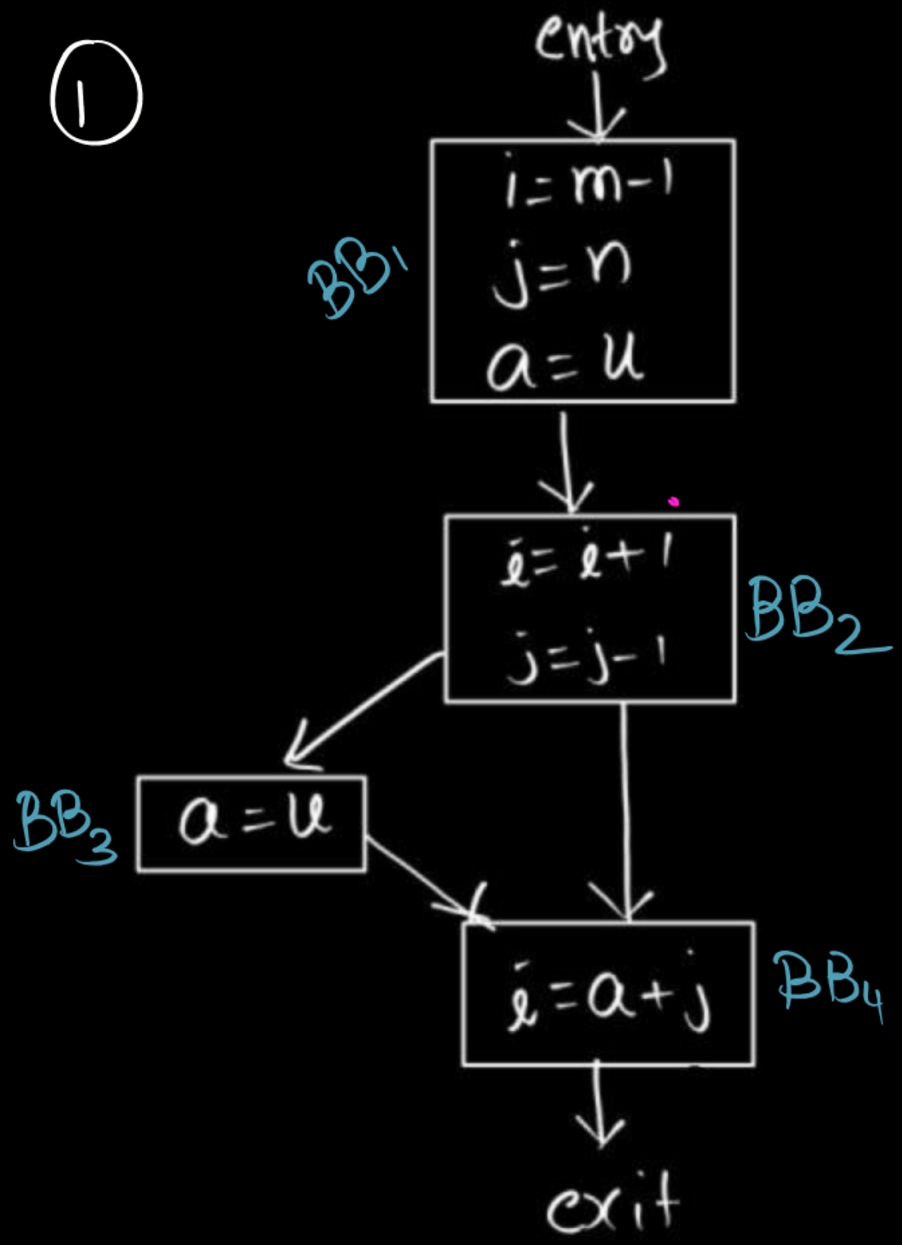
$= \{ l \mid l$ is used (read) at some statement in BB_k and no definition before the statement $\}$

KILL (DEF) Set

$KILL_k = \{x, y, z\}$

$= \{ v \mid v$ has definition in $BB_k \}$

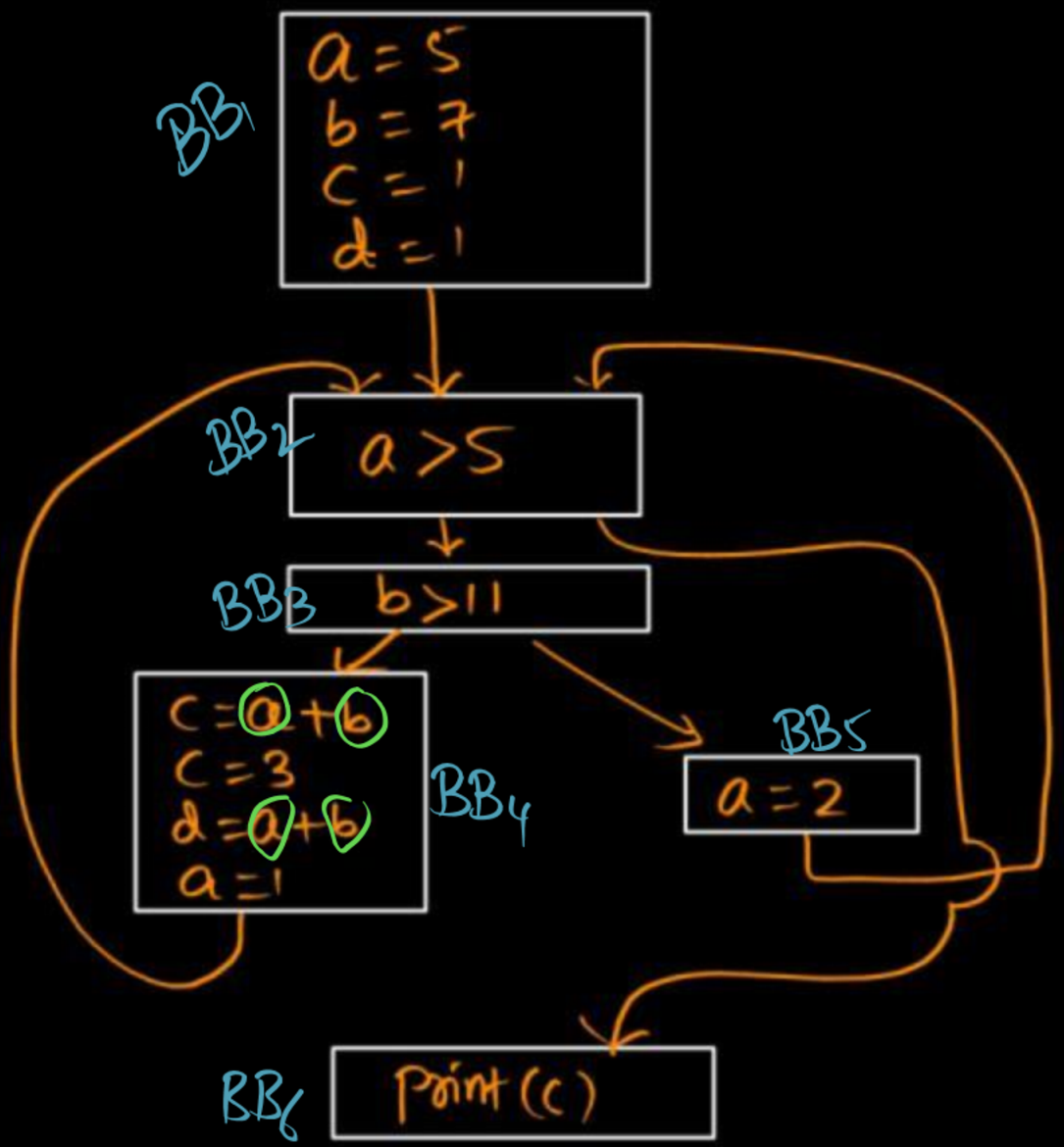
(1)



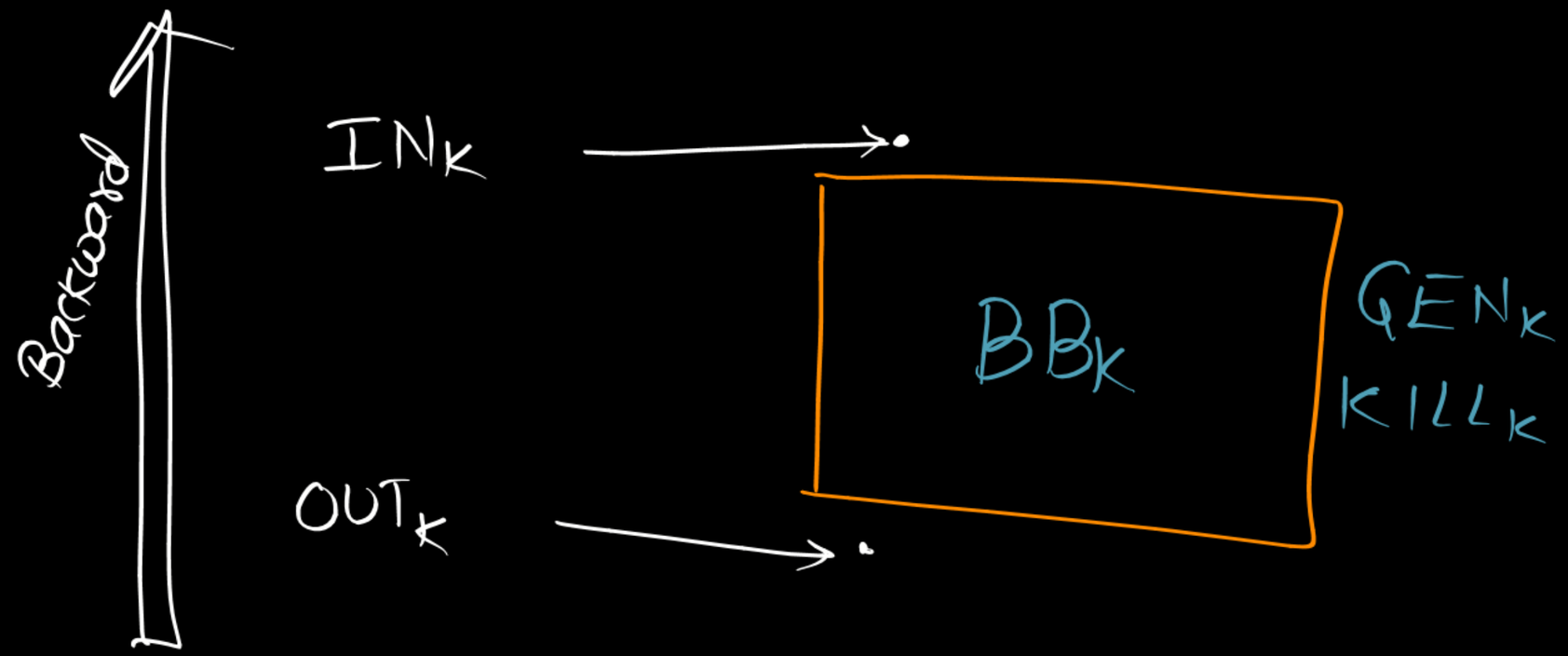
Compute GEN & KILL for each BB.

	GEN Set	KILL Set
BB_1	{m, n, u}	{i, j, a}
BB_2	{i, j}	{i, j}
BB_3	{u}	{a}
BB_4	{a, j}	{i}

(2)

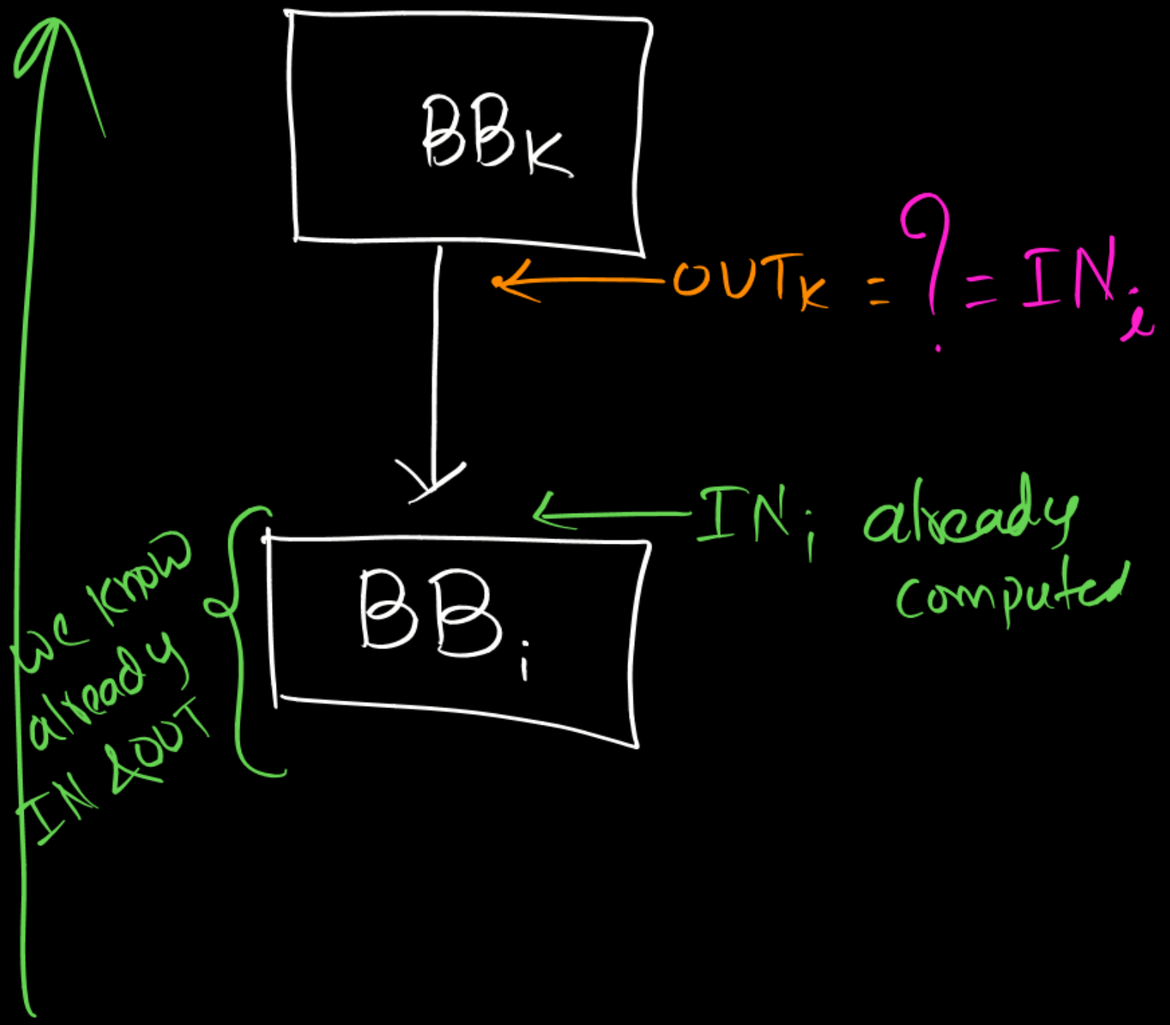


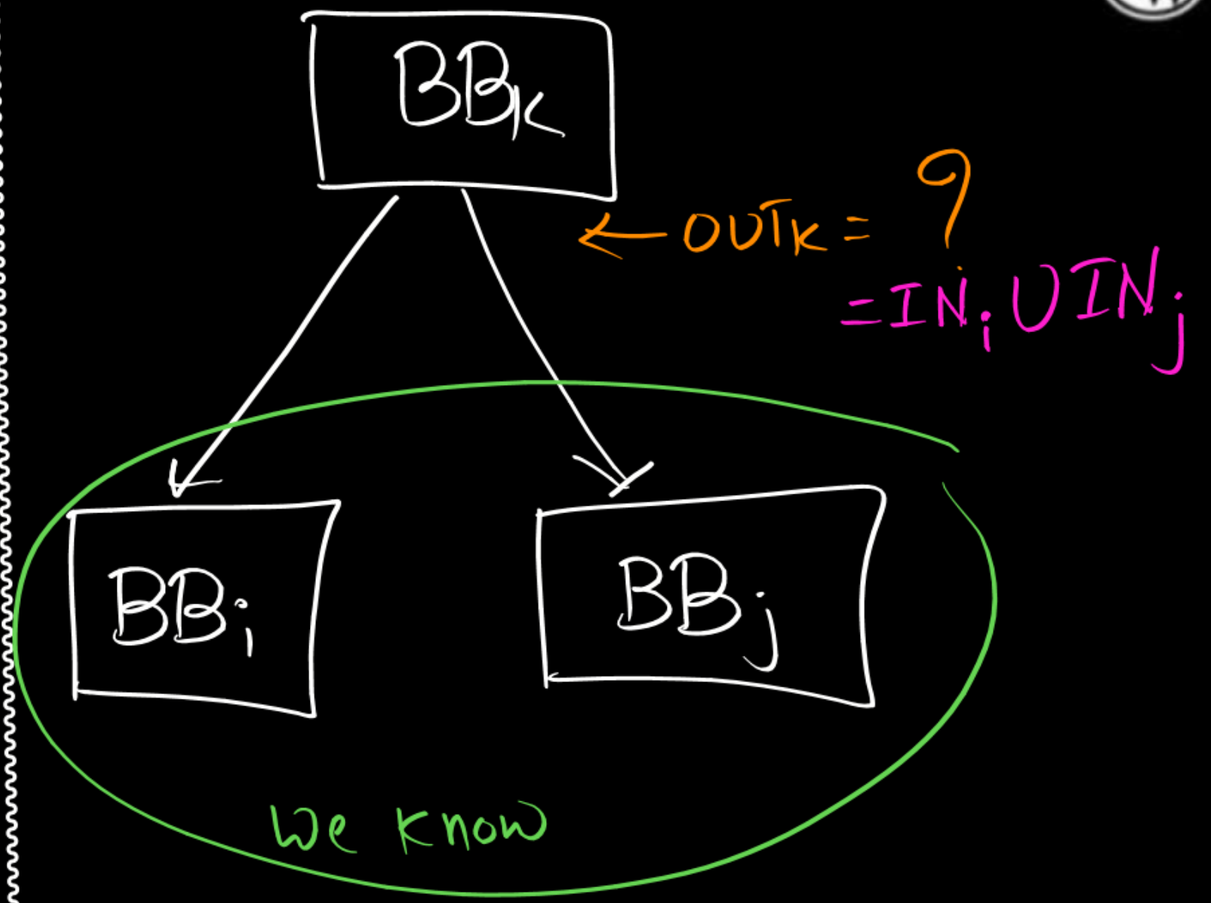
BB	GEN (USE)	KILL (DEF)
1	ϕ	a, b, c, d
2	a	ϕ
3	b	ϕ
4	a, b	c, d, a
5	ϕ	a
6	c	ϕ


Backward
IN_K
BB_K
GEN_K
$KILL_K$
OUT_K

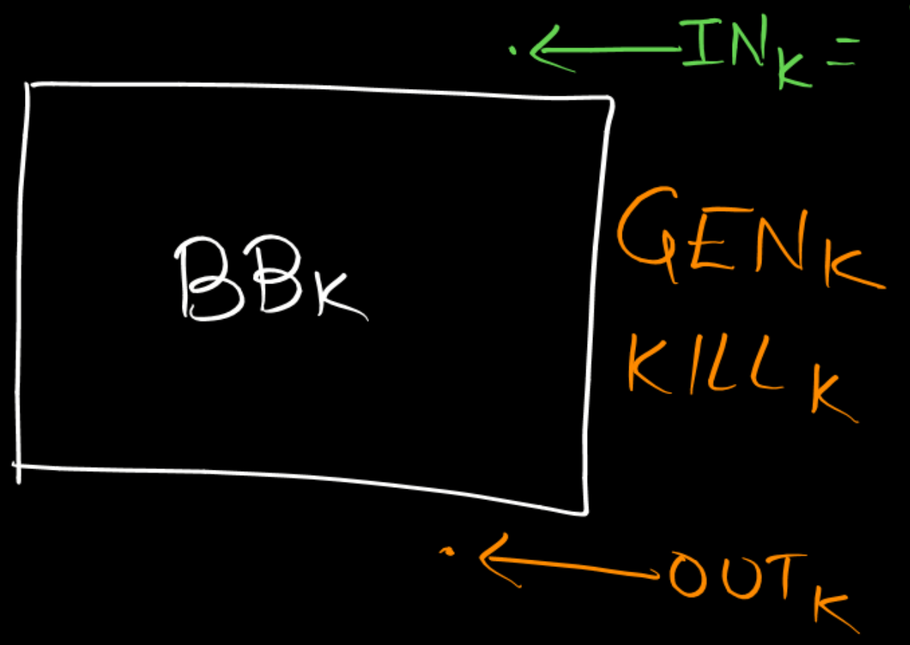
IN and OUT Sets Computation :

OUT_K for BB_K :

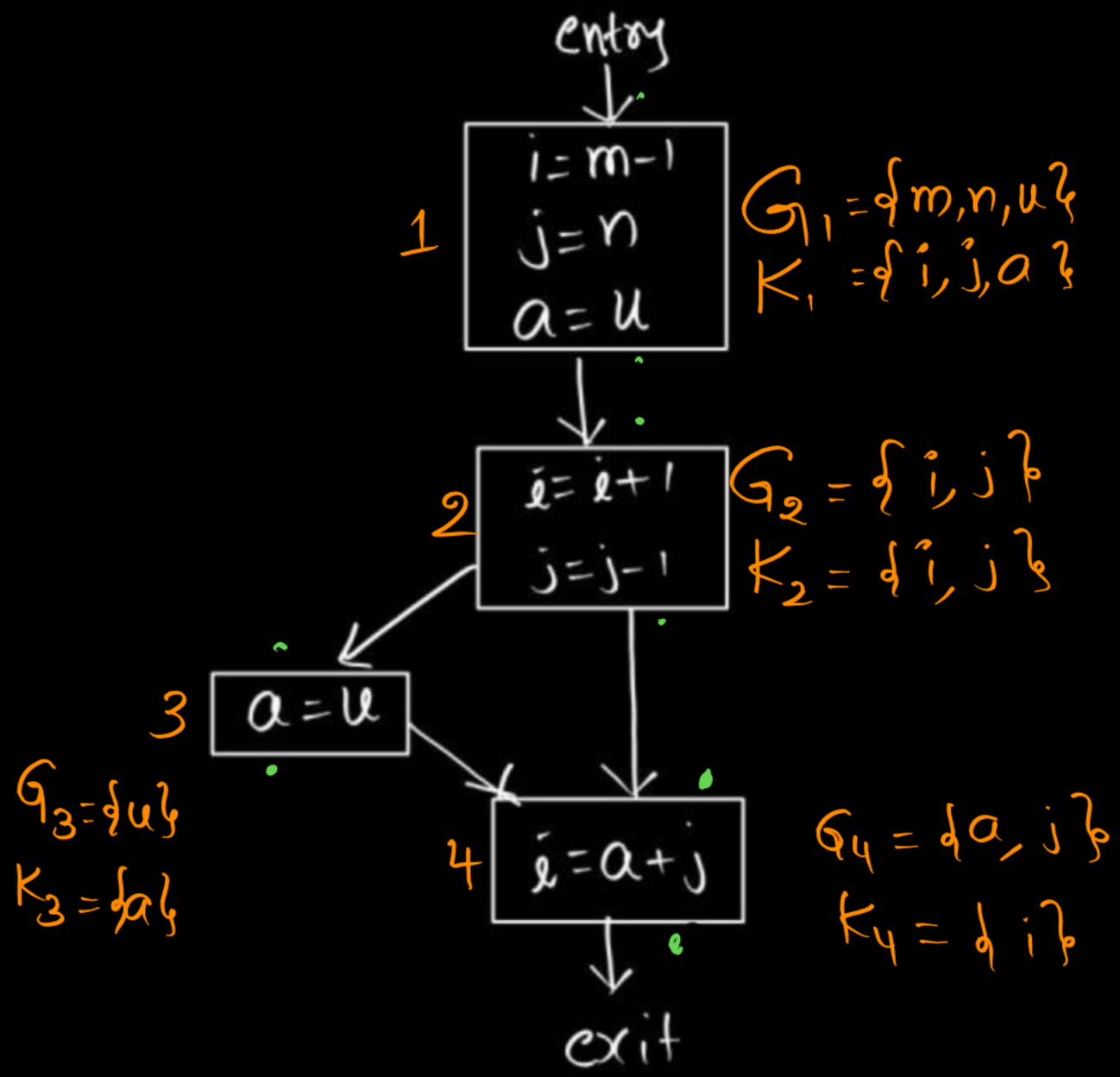

BB_K
$OUT_K = ? = IN_i$
IN_i already computed
BB_i
We know already IN & OUT

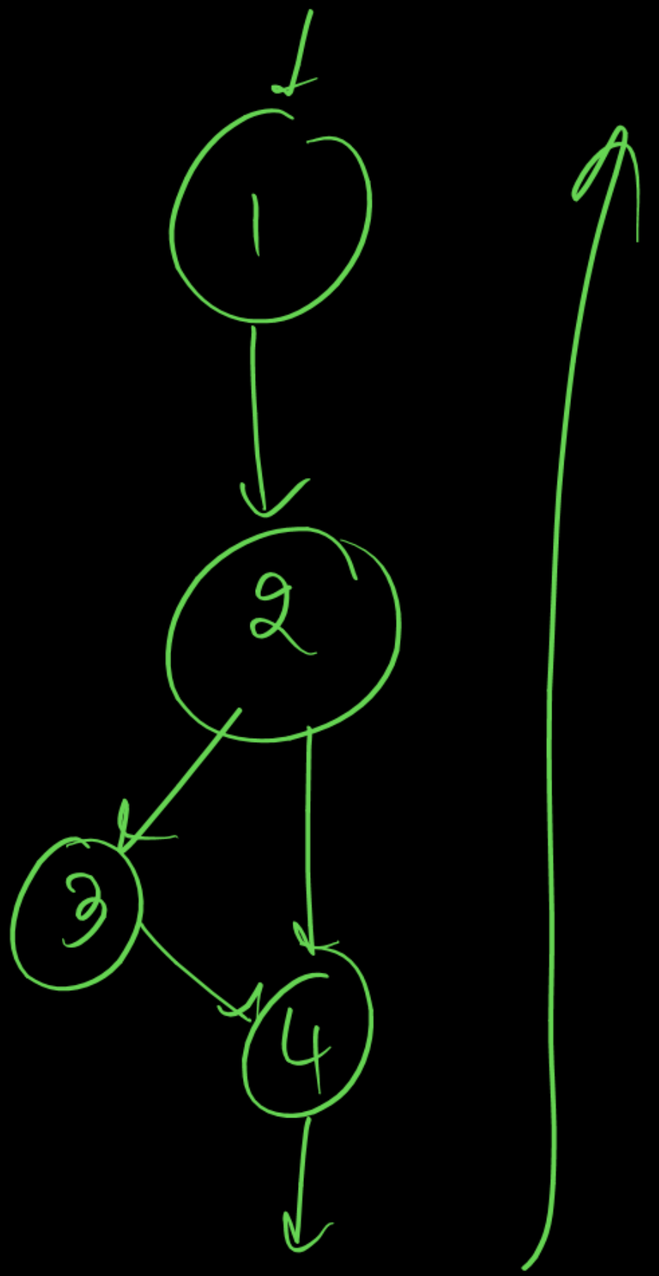

BB_K
$OUT_K = ?$
$= IN_i \cup IN_j$
BB_i
BB_j
We know

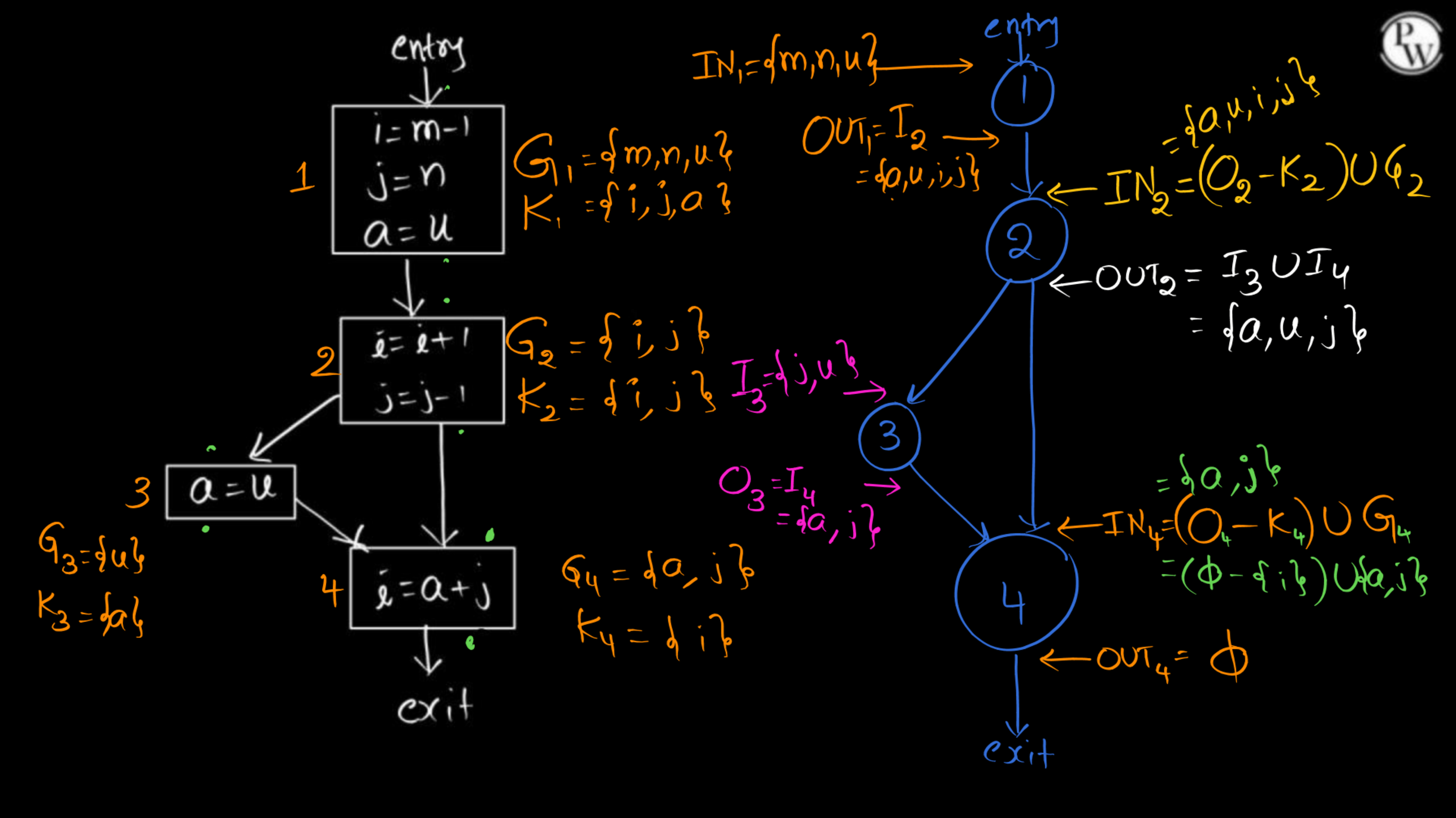
IN_K for BB_K :



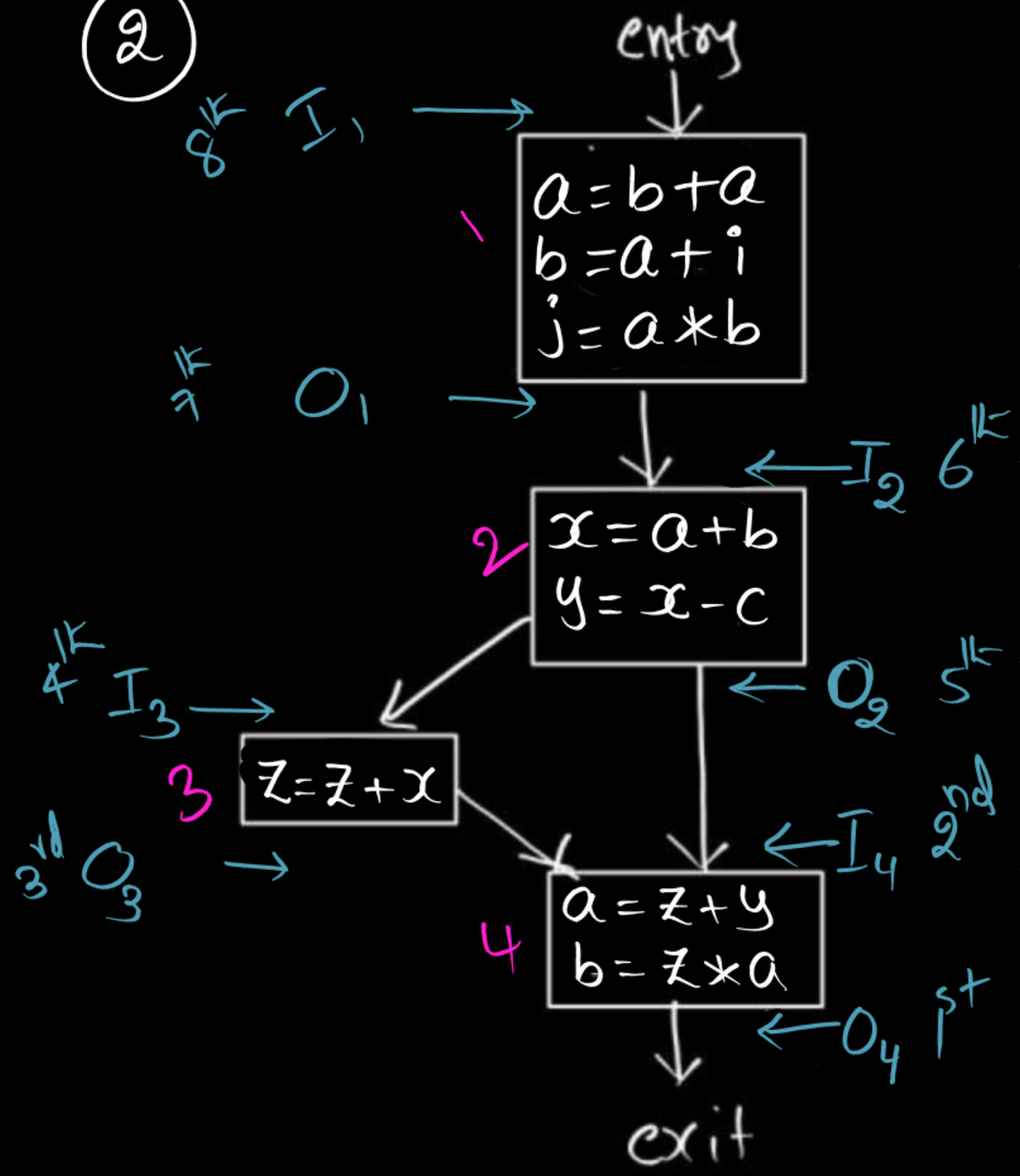

Entry
Backward Analysis
exit


entry
1
i = m-1
j = n
a = u
G1 = {m, n, u}
K1 = {i, j, a}
2
i = i+1
j = j-1
G2 = {i, j}
K2 = {i, j}
3
a = u
G3 = {u}
K3 = {a}
4
i = a+j
G4 = {a, j}
K4 = {i}
exit

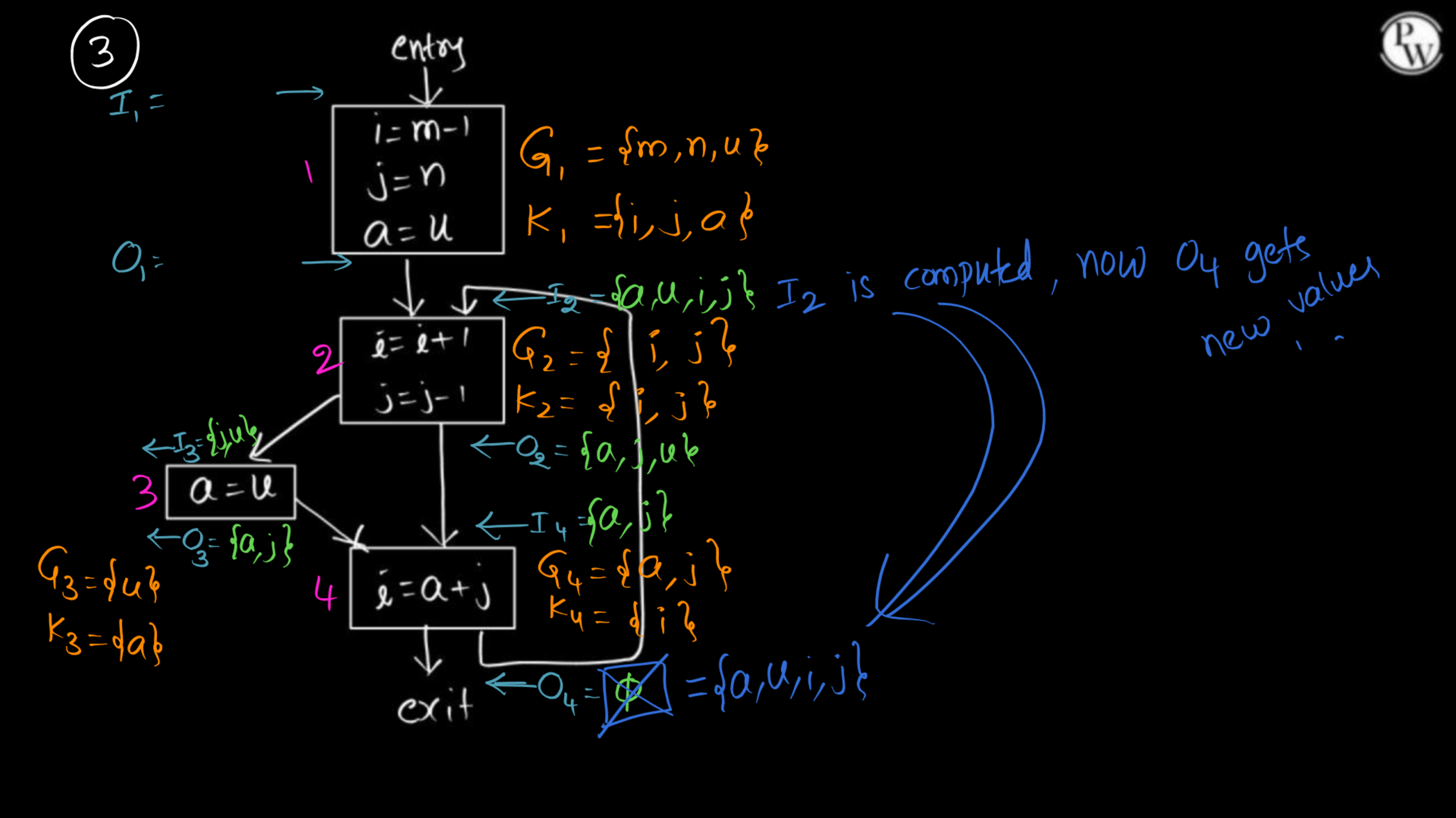

1
2
3
4

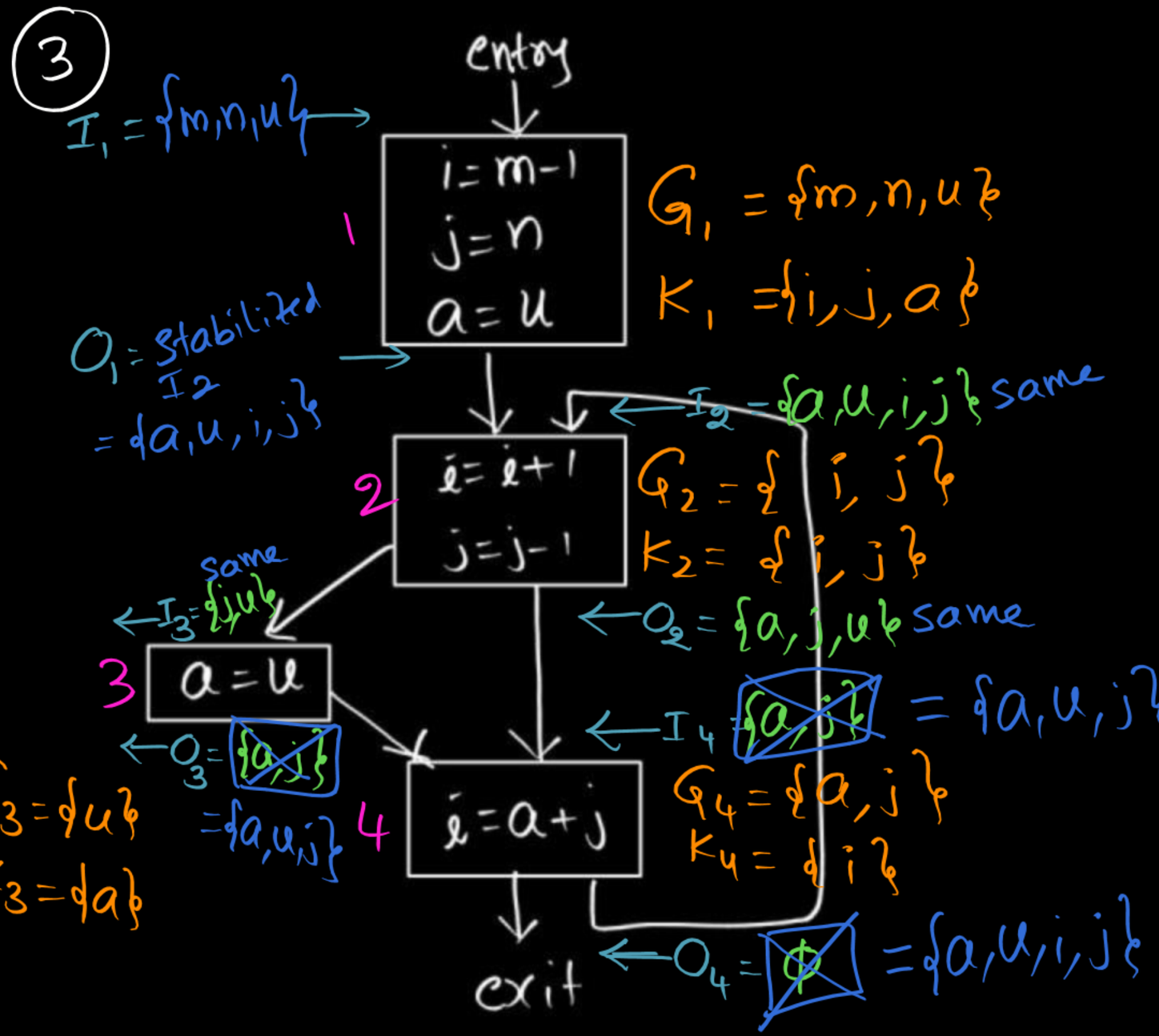

entry
i = m-1
j = n
a = u
1
G1 = {m, n, u}
K1 = {i, j, a}
i = i+1
j = j-1
2
G2 = {i, j}
K2 = {i, j}
a = u
3
G3 = {u}
K3 = {a}
i = a+j
4
G4 = {a, j}
K4 = {i}
exit
IN1 = {m, n, u}
OUT1 = I2 = {a, u, i, j}
IN2 = (O2 - K2) U G2 = {a, u, i, j}
OUT2 = I3 U I4 = {a, u, j}
I3 = {j, u}
O3 = I4 = {a, j}
IN4 = (O4 - K4) U G4 = (Φ - {i}) U {a, j} = {a, j}
OUT4 = Φ
exit

(2)

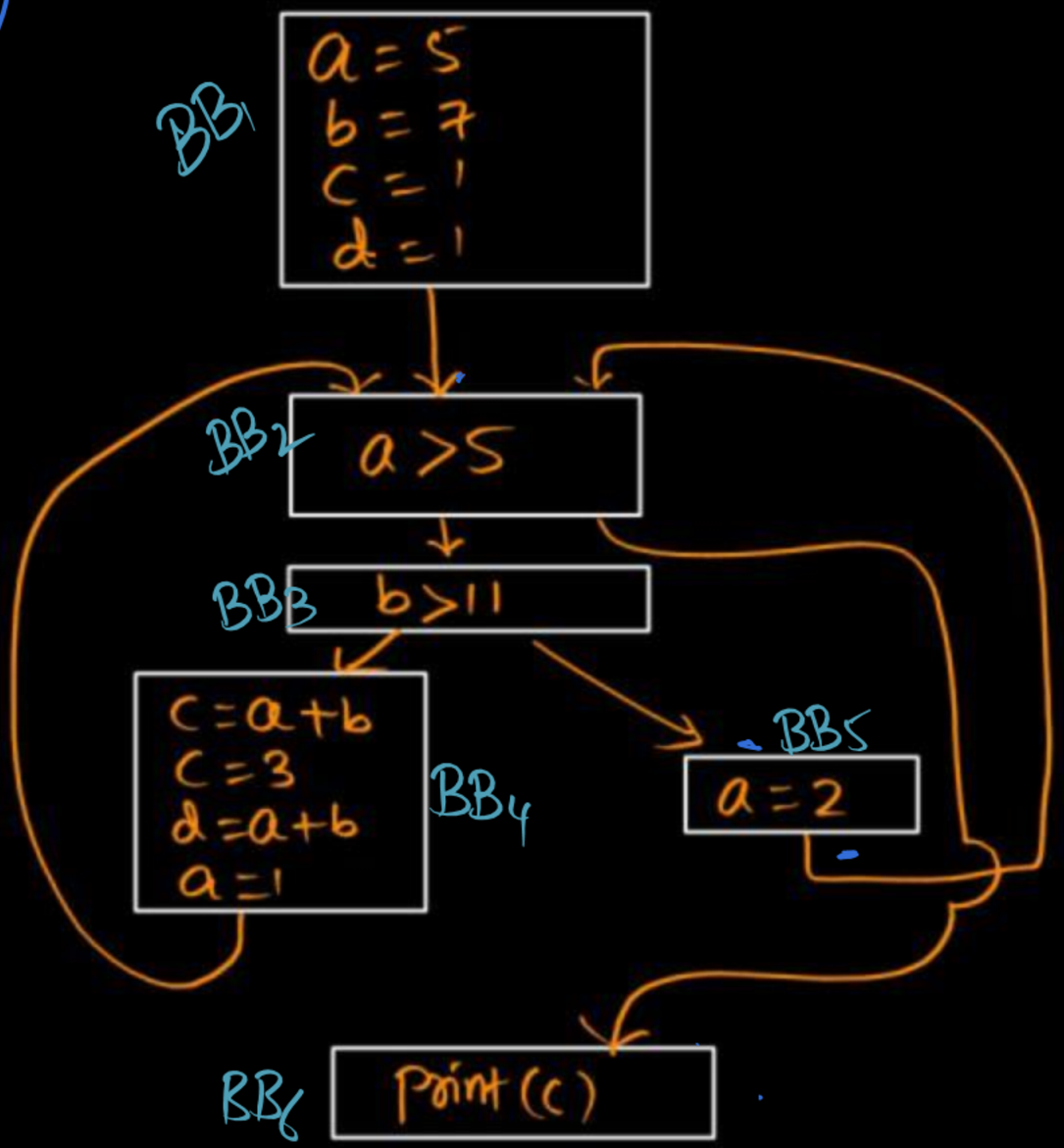


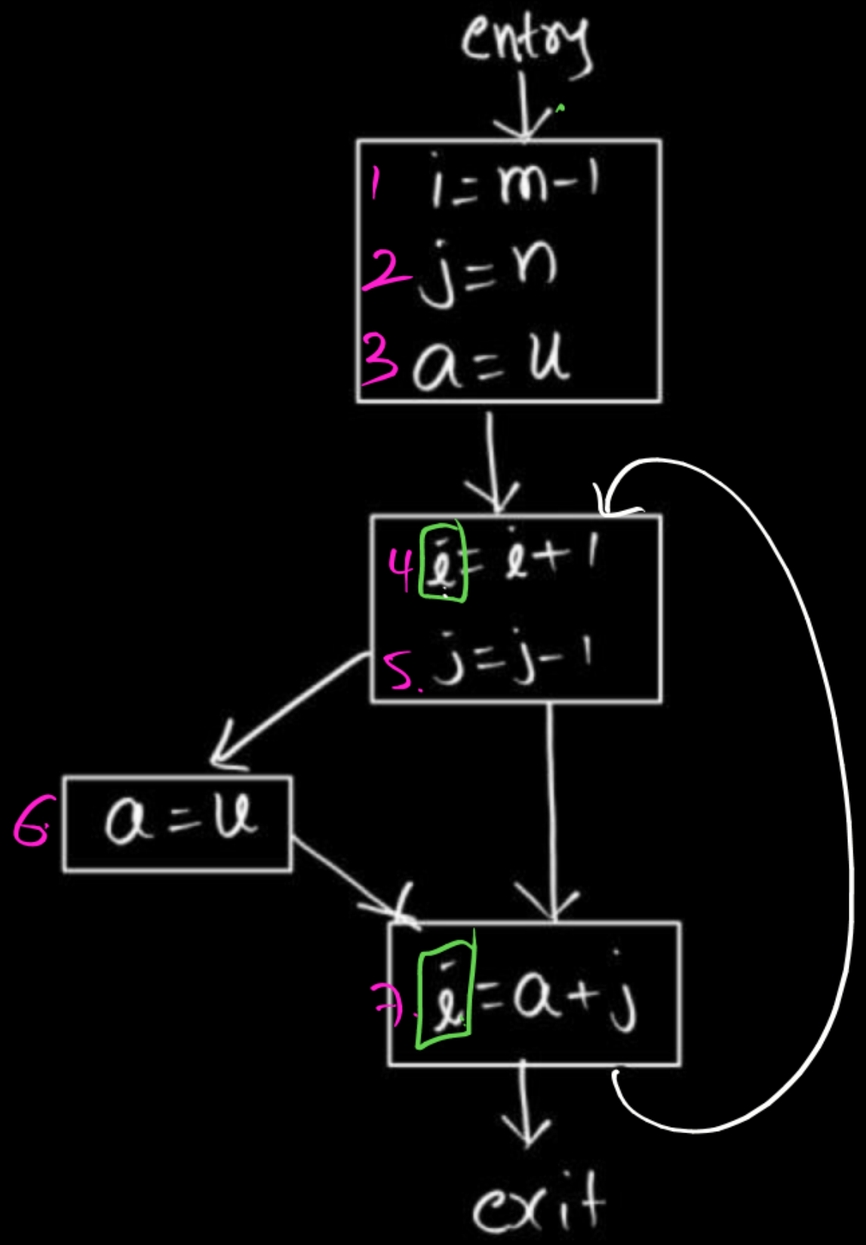
BB	GEN	KILL	OUT	IN
1	a, b, i	a, b, j	a, b, c, z	a, b, i, c, z
2	a, b, c	x, y	x, y, z	a, b, c, z
3	x, z	z	$I_4 = z, y$	x, y, z
4	z, y	a, b	ϕ	z, y


3
entry
I_1 =
O_1 =
1
i = m-1
j = n
a = u
$G_1 = \{m, n, u\}$
$K_1 = \{i, j, a\}$
$I_2 = \{a, u, i, j\}$
2
i = i+1
j = j-1
$G_2 = \{i, j\}$
$K_2 = \{i, j\}$
$O_2 = \{a, j, u\}$
$I_3 = \{j, u\}$
3
a = u
$O_3 = \{a, j\}$
$G_3 = \{u\}$
$K_3 = \{a\}$
$I_4 = \{a, j\}$
4
i = a+j
$G_4 = \{a, j\}$
$K_4 = \{i\}$
exit
$O_4 = \phi = \{a, u, i, j\}$
I_2 is computed, now O_4 gets new values


3
entry
$I_1 = \{m, n, u\}$
i = m-1
j = n
a = u
1
$G_1 = \{m, n, u\}$
$K_1 = \{i, j, a\}$
O_1 = stabilized
$I_2 = \{a, u, i, j\}$
$I_2 = \{a, u, i, j\}$ same
i = i+1
j = j-1
2
$G_2 = \{i, j\}$
$K_2 = \{i, j\}$
same
$I_3 = \{j, u\}$
$O_2 = \{a, j, u\}$ same
a = u
3
$O_3 = \{a, j\}$
$I_4 = \{a, j\} = \{a, u, j\}$
$G_3 = \{u\}$
$K_3 = \{a\}$
$= \{a, u, j\}$
i = a+j
4
$G_4 = \{a, j\}$
$K_4 = \{i\}$
exit
$O_4 = \phi = \{a, u, i, j\}$

H.W. 4





At statement 1 : Live Range(i) = ?

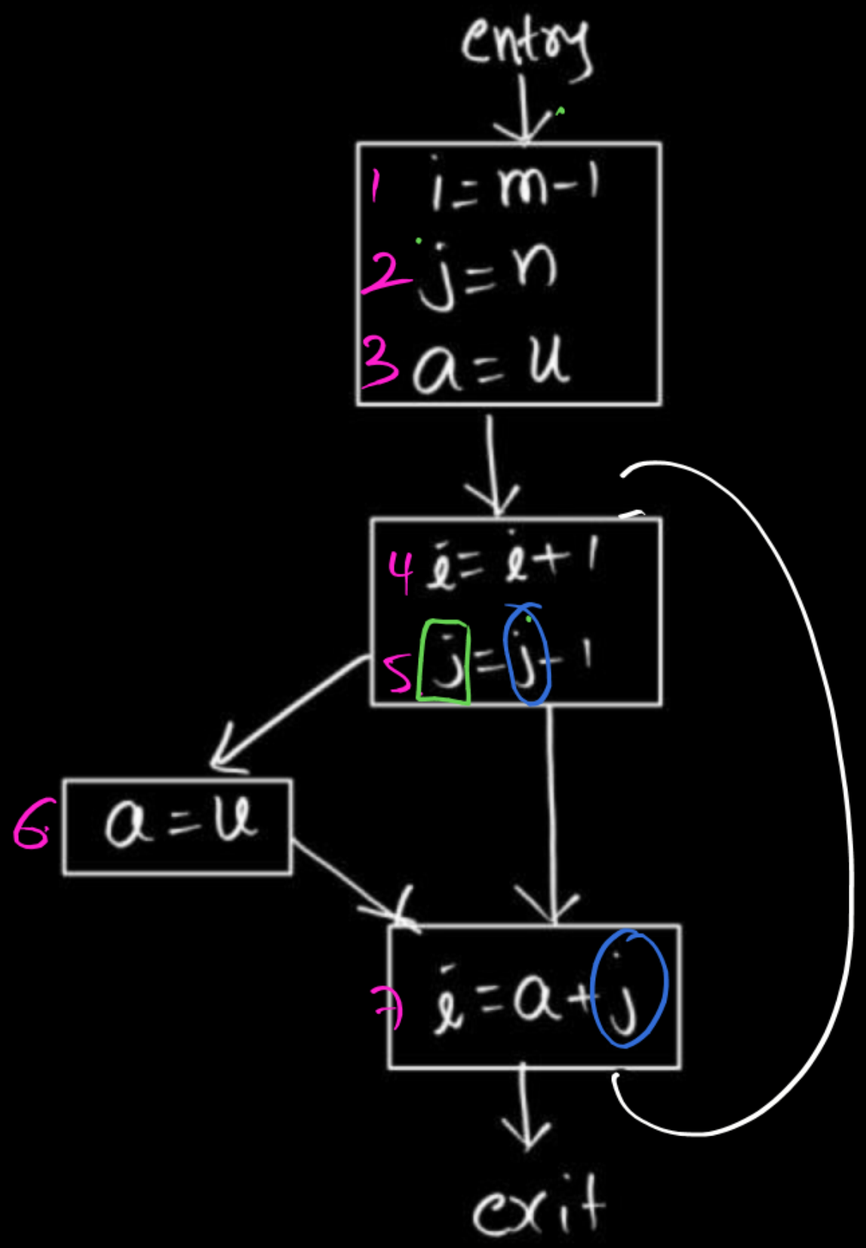
1 → 2 → 3 → 4

At statement 4: Live Range(i) = ?

Nill

At statement 7 : Live Range(i) = ?

7 → 4



At Statement 2 : Live Range (j) = ?

2 → 3 → 4 → 5

At Statement 5 : Live Range (j) = ?

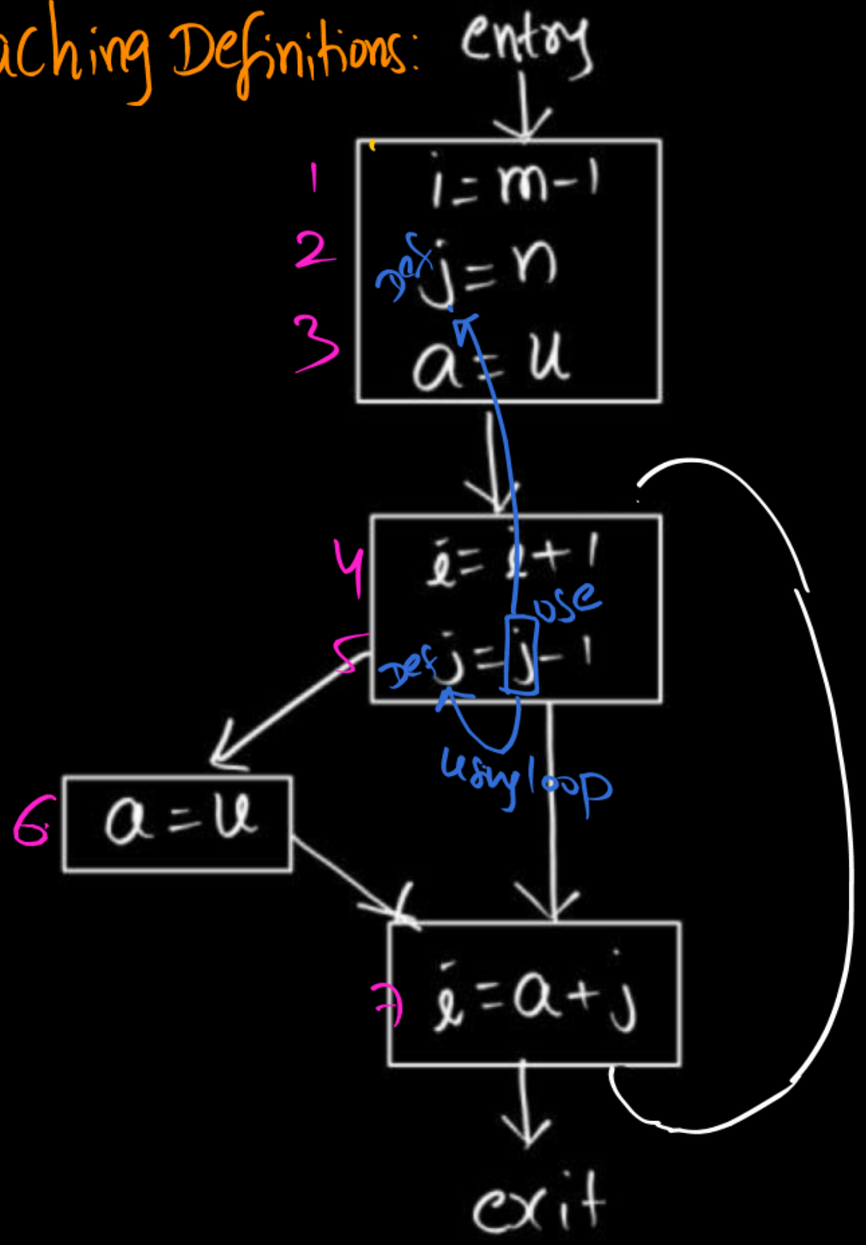
5 → 7

5 → 6 → 7

5 → 7 → 4 → 5

5 → 6 → 7 → 4 → 5

Reaching Definitions:



I) USE - DEF Chains

Source → Destination

At statement 5 (read j) for j:

Source → Destination

5 → 2

5 → 5

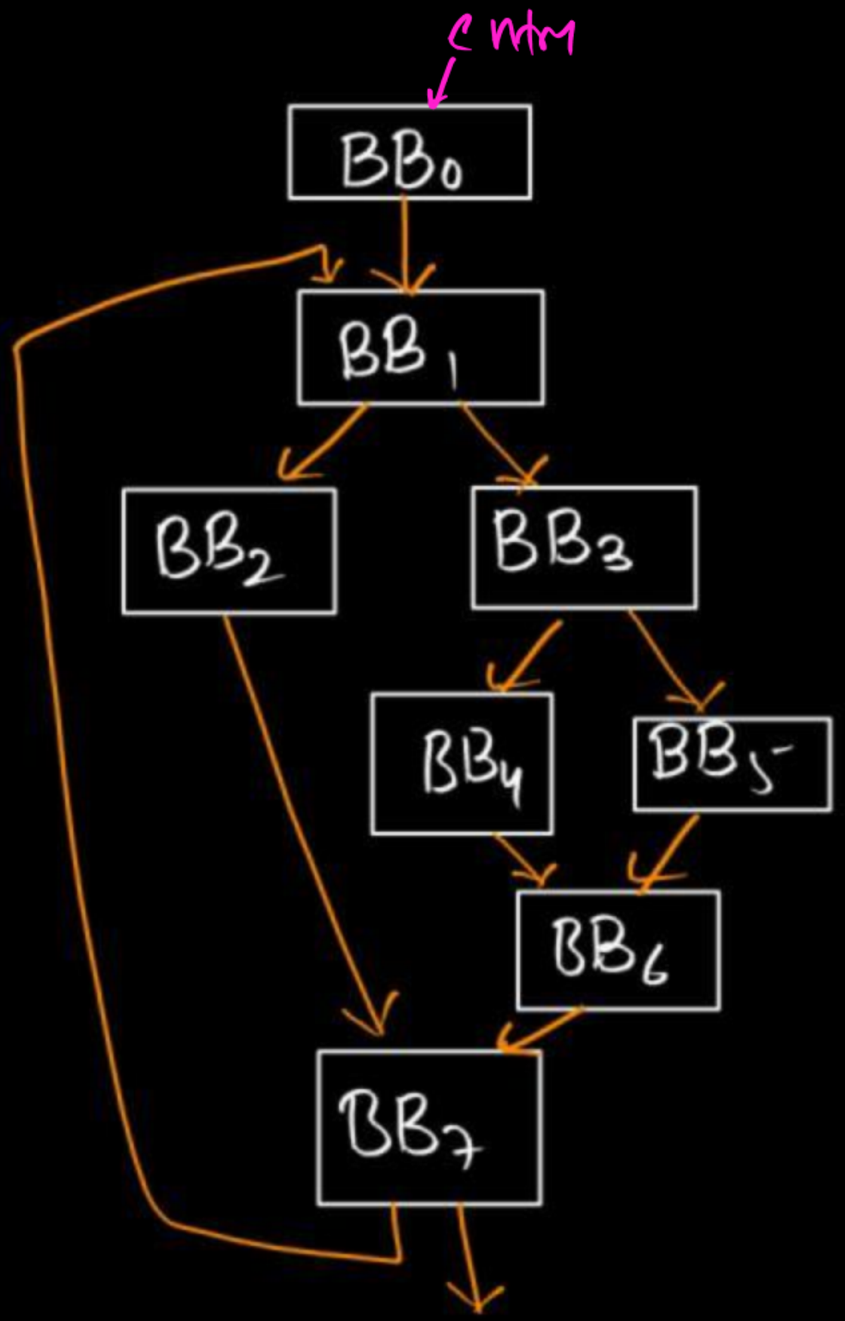
At statement 7 for j: 7 → 5

II) DEF - USE Chains

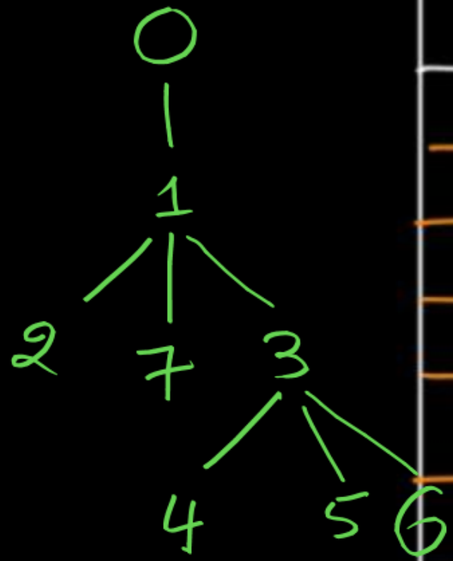
Source → Destination

At statement 5 for j: 5 → 7, 5 → 5

At statement 2 for j: 2 → 5

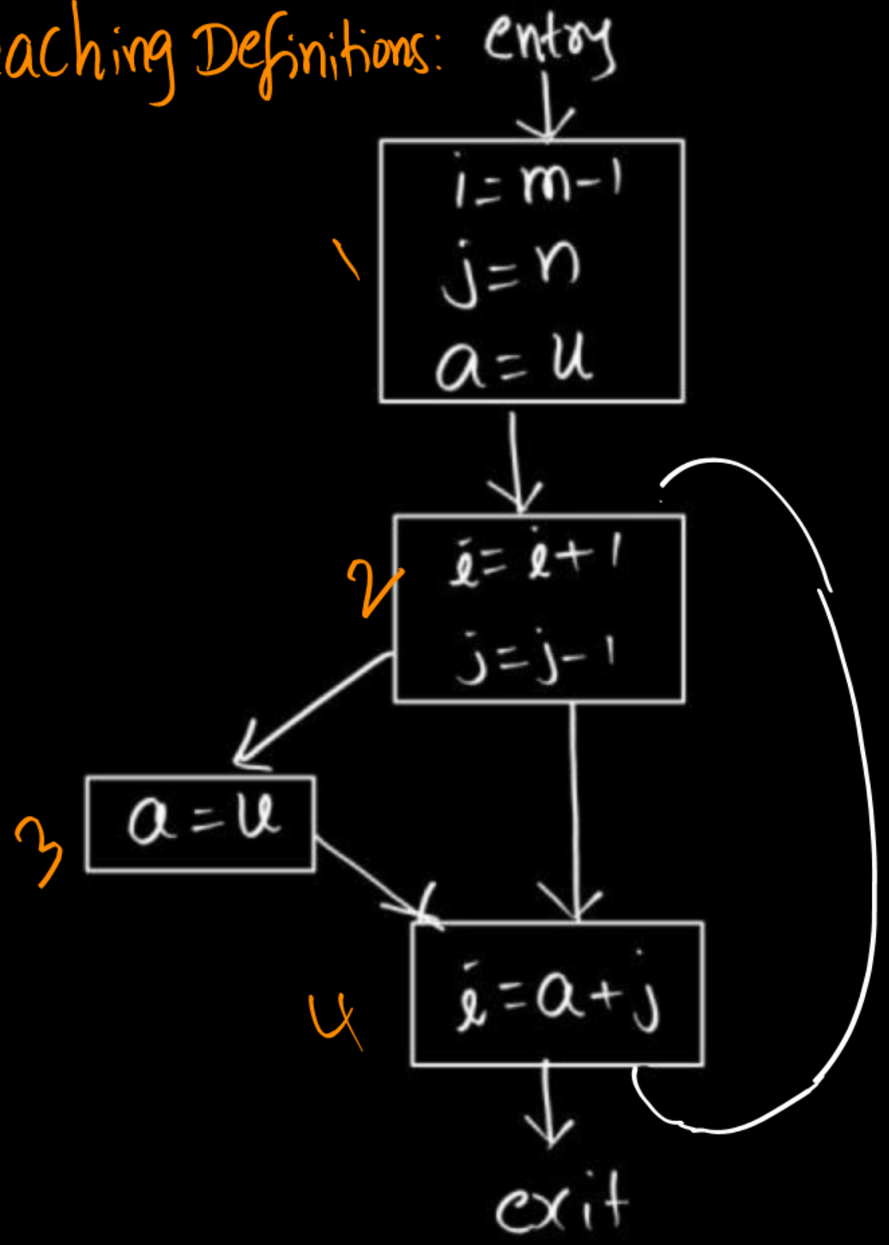


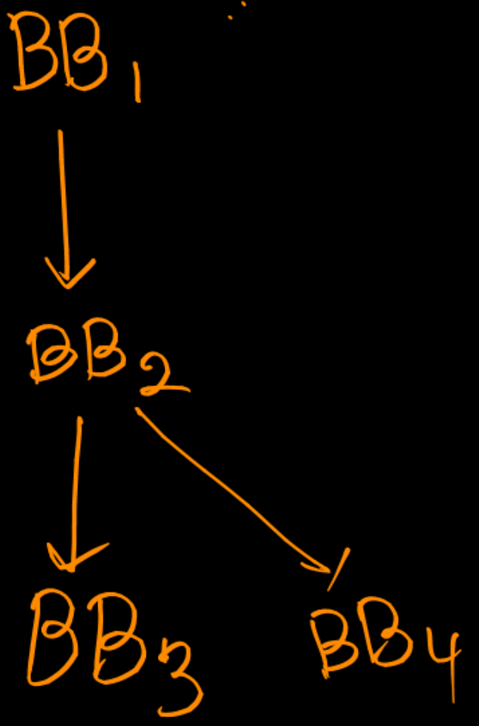
Dominator Tree



BB	Dominated by
0	0
1	0, 1
2	0, 1, 2
3	0, 1, 3
4	0, 1, 3, 4
5	0, 1, 3, 5
6	0, 1, 3, 6
7	0, 1, 7

Reaching Definitions:





Code Generation

- Dependent on m/c
- Registers, Addresses, Instructions, Addressing modes
- Register Allocation
  - Requires program Analysis
  - Uses Graph coloring

Run Time Environment

- Parameter passing
  - Call by value
  - Call by Reference
- Memory Allocation
  - Static
  - Stack
  - Heap
  - } Dynamic
- Activation Record

Static Memory Allocation $\Rightarrow$ Doesn't Support Recursion

Stack Memory Allocation $\Rightarrow$ Supports Recursion

Control Flow Graph

```
    s := 0
    i := 1
L1: if i > n goto L2
    t := i * i
    s := s + t
    i := i + 1
    goto L1
L2: return s
```

H.W:

Q1) Draw CFG

Q2) GEN, KILL, IN, OUT Sets for each BB

Q3) Domintor Tree

Q4) U-D Every use of i, find definitions of i

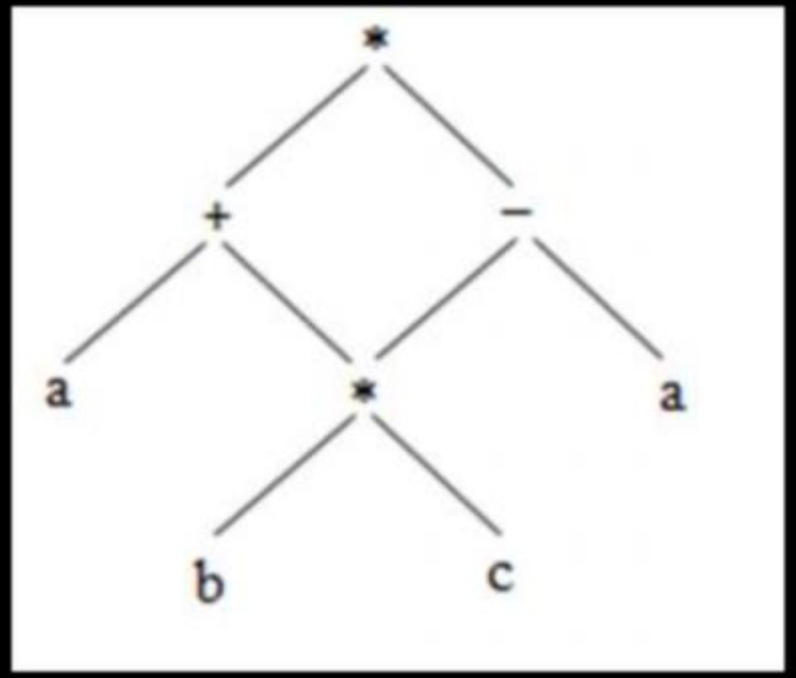
D-U Every def of i, find uses of i

Q5) Find Live variables at every statement

Control Flow Graph

```
1        receive m (val)
2        f0 ← 0
3        f1 ← 1
4        if m <= 1 goto L3
5        i ← 2
6    L1: if i <= m goto L2
7        return f2
8    L2: f2 ← f0 + f1
9        f0 ← f1
10       f1 ← f2
11       i ← i + 1
12       goto L1
13   L3: return m
```

Q1 Find equivalent expression for the following syntax tree.



(a) ((a + b) *c) *(b*(c − a))

(b) a + (b*c − a)

(c) (a + (b*c))*((b*c) − a)

(d) a*(a + b * c) − a

Q2

Find Equivalent Three Address code

X=((a*a)+(a*b))-(a*b)

Q3

Find SSA Code

$t_1 = b*c$

$t_2 = a+t_1$

$t_3 = b * c$

$t_4 = d * t_3$

$t_5 = t_2 + t_4$

Q4 Finding minimum number of variables in 3AC and SSA

$t_1 = a*b$

$t_2 = f/t_1$

$t_3 = a * t_2$

$t_4 = b * t_3$

$t_5 = t_4 + e$

Q5

Finding minimum number of variables in 3AC and SSA

$t_1 = (a + b)$
$t_2 = (c + d)$
$t_3 = t_1 + t_2$
$t_4 = t_1 + e$
$t_5 = t_4 + d$

Q6 Finding minimum number of variables in 3AC and SSA

```
p = q + r
s = p + 1
t = q + r
u = s + t
v = t + u
```

Q7

Finding minimum number of nodes in DAG

p = q + r

s = p + 1

t = q + r

u = s + t

v = t + u

Q8

Three address codes can be implemented by

(a) indirect triples
(b) direct triples
(c) quadruples
(d) none of the above

Q9

An array a[n] is used in the following Pseudo code and each element of array is size of 8 bytes.

```
do
i=i+1;
while (a[i]>v);
```

Which of the following is equivalent 3-address code for above program.

(a)
$L: t_1 = i+1$
$i = t_1$
$t_2 = i*8$
$t_3 = a[t_2]$
if $t_3 > v$ goto L

(b)
$L: t_1 = i+1$
$t_2 = i*8$
$t_3 = a[t_2]$
if $t_3 < v$ goto L

(c)
$L: t_1 = i+1$
$t_2 = i*8$
$t_3 = a[t_2]$
if $t_3 > v$ goto L

(d)
$L: t_1 = i+1$
$i = t_1$
$t_2 = i*8$
$t_3 = a[t_2]$
if $t_3 < v$ goto L

Q10

Consider the following code segment.

```
x = u – t;
y = x * v;
x = y + w;
y = t – z;
y = x * y;
```

The minimum number of total variables required to convert the above code segment to static single assignment form is _____.

(GATE – 16 – SET1)

Q11

Consider the following intermediate program in three address code

$p = a - b$

$q = p * c$

$p = u * v$

$q = p + q$

Which one of the following corresponds to a static single assignment form of the above code? **(GATE – 17- SET1)**

(a)
$p_1 = a - b$
$q_1 = p_1 * c$
$p_1 = u * v$
$q_1 = p_1 + q_1$

(b)
$p_3 = a - b$
$q_4 = p_3 * c$
$p_4 = u * v$
$q_5 = p_4 + q_4$

(c)
$p_1 = a - b$
$q_1 = p_2 * c$
$p_3 = u * v$
$q_2 = p_4 + q_3$

(d)
$p_1 = a - b$
$q_1 = p * c$
$p_2 = u * v$
$q_2 = p + q$

Q12

For a C program accessing X[i] [j] [k], the following intermediate code is generated by a complier. Assume that the size of an integer is 32 bits and the size of character is 8 bits.

$t_0 = i * 1024$

$t_1 = j * 32$

$t_2 = k * 4$

$t_3 = t1 + t0$

$t_4 = t3 + t2$

$t_5 = X [t4]$

Which one of the following statements about the source code for the C program is **CORRECT**?

(GATE – 14 – SET2)

(a) X is declared as "int X[32] [32] [8]"

(b) X is declared as "int X [4] [1024] [32]"

(c) X is declared as "char X[4] [32] [8]"

(d) X is declared as "char X[32] [16] [2]"

Q13

The program below uses six temporary variables a, b, c, d, e, f.

```
a = 1
b = 10
c = 20
d = a + b
e = c + d
f = c + e
b = c + e
e = b + f
d = 5 + e
return d + f
```

Assuming that all operations take their operands from registers, what is the minimum number of registers needed to execute this program without spilling?

(GATE - 10)

(a) 2
(b) 3
(c) 4
(d) 6

Q14

Consider the basic block given below.

$a = b + c$

$c = a + d$

$d = b + c$

$e = d - b$

$a = e + b$

The minimum number of nodes and edges present in the DAG representation of the above basic block respectively are

(GATE – 14 – SET3)

(a) 6 and 6

(b) 8 and 10

(c) 9 and 12

(d) 4 and 4

Q15

Find number of nodes and edges in CFG.

```
(1) X := 20
(2) if X>=10 goto (8)
(3) X := X-1
(4) A[X] := 10
(5) if X<>4 goto (7)
(6) X := X-2
(7) goto (2)
(8) Y := X+5
```

MCQ

Consider the following grammar G.

$S \rightarrow F \mid H$

$F \rightarrow p \mid c$

$H \rightarrow d \mid c$

Where S, F and H are non-terminal symbols, p, d and c are terminal symbols.

Which of the following statements(s) is/are correct?

S1: LL(1) can parse all strings that are generated using grammar G.

S2: LR(1) can parse all strings that are generated using grammar G.

[GATE-2015-CS: 2M]

A. Only S1

B. Only S2

C. Both S1 and S2

D. Neither S1 and S2

MCQ

Among simple LR (SLR), canonical LR, and look-ahead LR (LALR), which of the following pairs identify the method that is very easy to implement and the method that is the most powerful, in that order?

[GATE-2015-CS: 1M]

A. SLR, LALR

B. Canonical LR, LALR

C. SLR, canonical LR

D. LALR, canonical LR

MCQ

Which one of the following is TRUE at any valid state in shift-reduce parsing?

[GATE-2015-CS: 1M]

A. Viable prefixes appear only at the bottom of the stack and not inside

B. Viable prefixes appear only at the top of the stack and not inside

C. The stack contains only a set of viable prefixes

D. The stack never contains viable prefixes

MCQ

Consider the grammar defined by the following production rules, with two operators * and +

$S \rightarrow T * P$

$T \rightarrow U \mid T * U$

$P \rightarrow Q + P \mid Q$

$Q \rightarrow Id$

$U \rightarrow Id$

Which one of the following is TRUE? [GATE-2014-CS: 1M]

A. + is left associative, while * is right associative

B. + is right associative, while * is left associative

C. Both + and * are right associative

D. Both + and * are left associative

MCQ

A canonical set of items is given below

$S \rightarrow L. > R$

$Q \rightarrow R.$

On input symbol > the set has

[GATE-2014-CS: 1M]

(A) a shift-reduce conflict and a reduce-reduce conflict.

(B) a shift-reduce conflict but not a reduce-reduce conflict.

(C) a reduce-reduce conflict but not a shift-reduce conflict.

(D) neither a shift-reduce nor a reduce-reduce conflict.

MCQ

Consider the following two sets of LR(1) items of LR(1) grammar.

X → c.X,c/d	X→c.X,$
X → .cX,c/d	X→.cX,$
X → .d,c/d	X→.d,$

Which of the following statement related to merging of the two sets in the corresponding LALR parser is/are FALSE?

1. Cannot be merged since look aheads are different.
2. Can be merged but will result in S-R conflict.
3. Can be merged but will result in R-R conflict.
4. Cannot be merged since goto on c will lead to two different sets.

[GATE-2015-CS: 2M]

(A) 1 only
(B) 2 only
(C) 1 and 4 only
(D) 1, 2, 3 and 4

MCQ

What is the maximum number of reduces moves that can be taken by a bottom up parser for a grammar with no epsilon and unit productions (i.e. of type $A \rightarrow \varepsilon$ and $A \rightarrow a$) to parse a string with n tokens?

[GATE-2013-CS: 1M]

(A) n/2

(B) n-1

(C) 2n-1

(D) 2^n

MCQ

The grammar S → aSa | bS | c is

[GATE-2010-CS: 2M]

A. LL(1) but not LR(1)

B. LR(1) but not LL(1)

C. Both LL(1) and LR(1)

D. Neither LL(1) nor LR(1)

MCQ

An LALR(1) parser for a grammar G can have shift-reduce (S-R) conflicts if and only if

[GATE-2008-CS: 2M]

A. The SLR(1) parser for G has S-R conflicts

B. The LR(1) parser for G has S-R conflicts

C. The LR(0) parser for G has S-R conflicts

D. The LALR(1) parser for G has reduce-reduce conflicts.

MCQ

Which of the following describes a handle (as applicable to LR-parsing) appropriately?

[GATE-2008-CS: 1M]

A. It is the position in a sentential form where the next shift or reduce operation will occur.

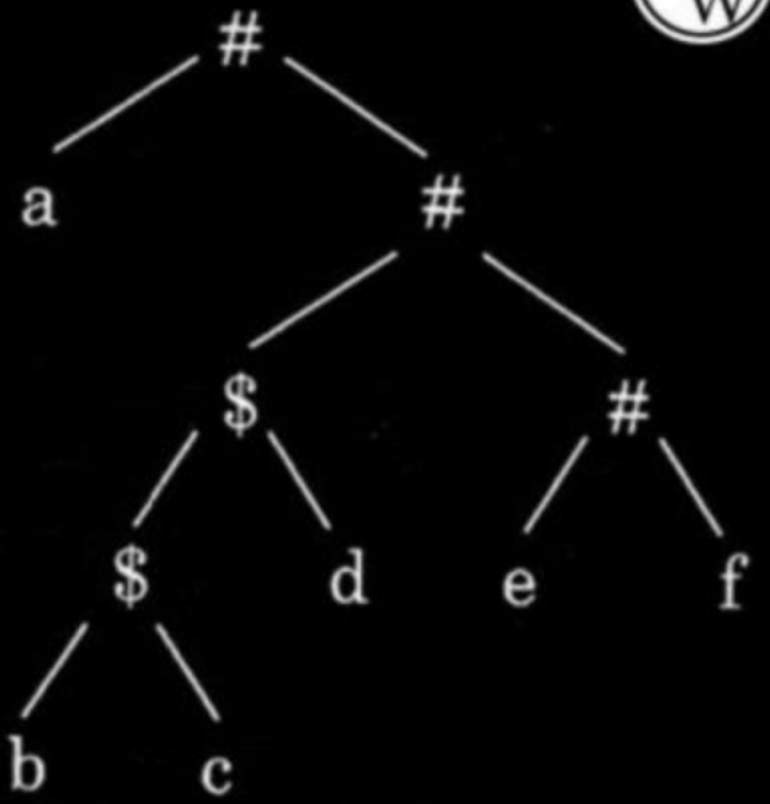
B. It is non-terminal whose production will be used for reduction in the next step

C. It is a production that may be used for reduction in a future step along with a position in the sentential form where the next shift or reduce operation will occur.

D. It is the production p that will be used for reduction in the next step along with a position in the sentential form where the right-hand side of the production may be found.

MCQ

Consider the following parse tree for the expression
a # b $ c $ d # e # f,
involving two binary operators $ and #.
Which one of the following is correct for
the given parse tree? [GATE-2018-CS: 2M]



(A) *$ has higher precedence and is left associative; # is right associative*

(B) *# has higher precedence and is left associative; $ is right associative*

(C) *$ has higher precedence and is left associative; # is left associative*

(D) *# has higher precedence and is right associative; $ is left associative.*

MCQ

Which one of the following statements is FALSE? [GATE-2018-CS: 1M]

A. Context-free grammar can be used to specify both lexical and syntax rules.

B. Type checking is done before parsing.

C. High-level language programs can be translated to different Intermediate Representations.

D. Arguments to a function can be passed using the program stack

NAT

The attribute of three arithmetic operators in some programming language are given below.

OPERATOR	PRECEDENCE	ASSOCIATIVITY	ARITY
+	High	Left	Binary
-	Medium	Right	Binary
*	Low	Left	Binary

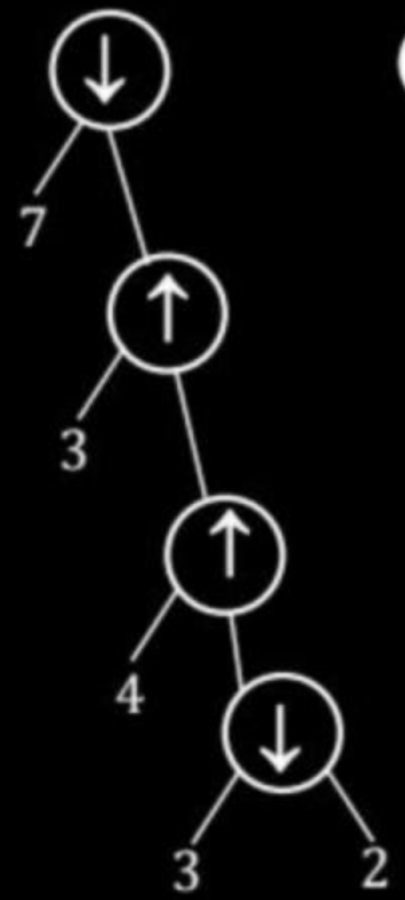
The value of the expression *2 – 5 + 1 – 7 * 3* in this language is _____.

[GATE-2016-CS: 2M]

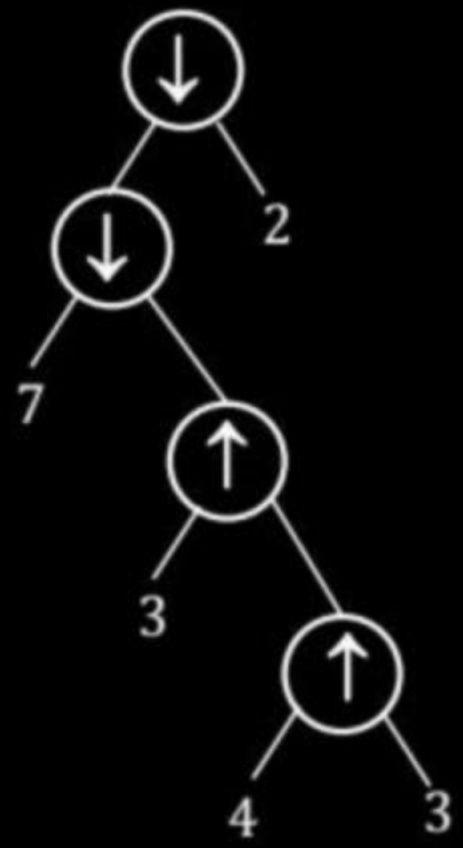
MCQ

Consider two binary operators '↑' and '↓' with the precedence of operator ↓ being lower than that of the operator ↑. Operator ↑ is right associative while operator ↓ is left associative. Which one of the following represents the parse tree for expression (*7 ↓ 3 ↑ 4 ↑ 3 ↓ 2*) [GATE-2011-CS: 2M]

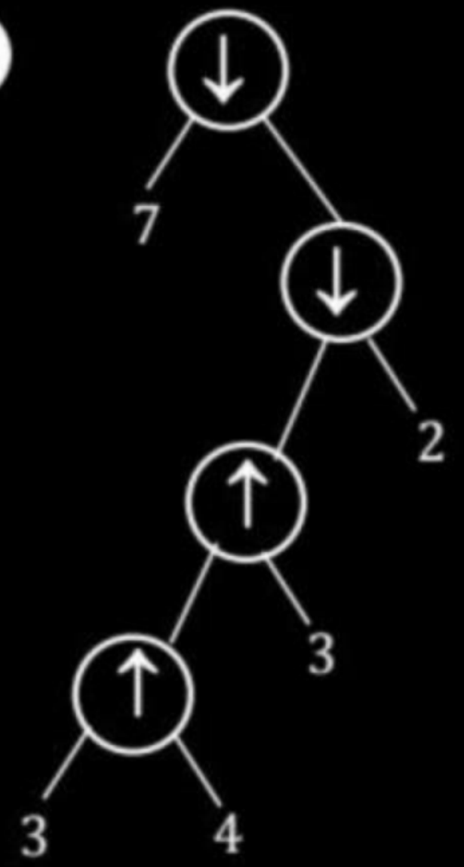
A



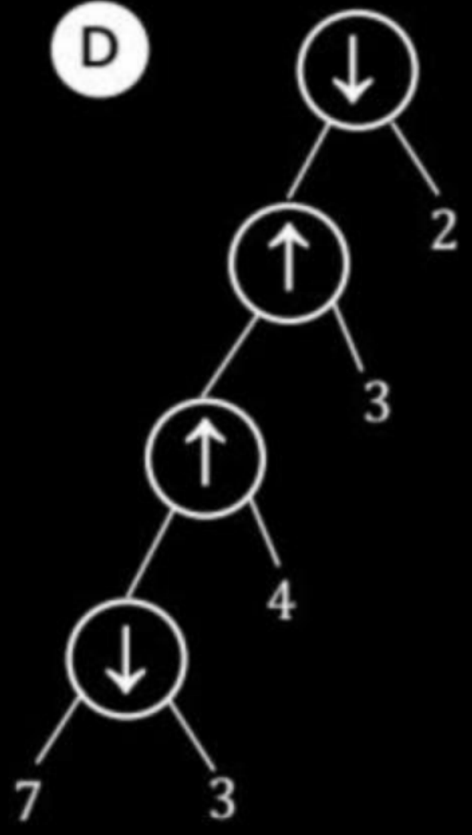
B



C



D



MCQ

A student wrote two context-free grammars G1 and G2 for generating a single C-like array declaration. The dimension of the array is at least one. For example,

int a[10][3];

The grammars use D as the start symbol, and use six terminal symbols int; id [ ] num.

Grammar G1

$D \rightarrow$ int L;
$L \rightarrow$ id [E
$E \rightarrow$ num]
$E \rightarrow$ num][E

Grammar G2

$D \rightarrow$ int L;
$L \rightarrow$ id E
$E \rightarrow E$[num]
$E \rightarrow$ [num]

Which of the grammars correctly generate the declaration mentioned above?

(A) Both G1 and G2
(B) Only G1
(C) Only G2
(D) Neither G1 nor G2

[GATE-2016-CS: 2M]

GATE Journey :

I) Believe in myself

II) Hard Work

III) Time

IV) Schedule

} Pen & paper

V) Focused on Single Task

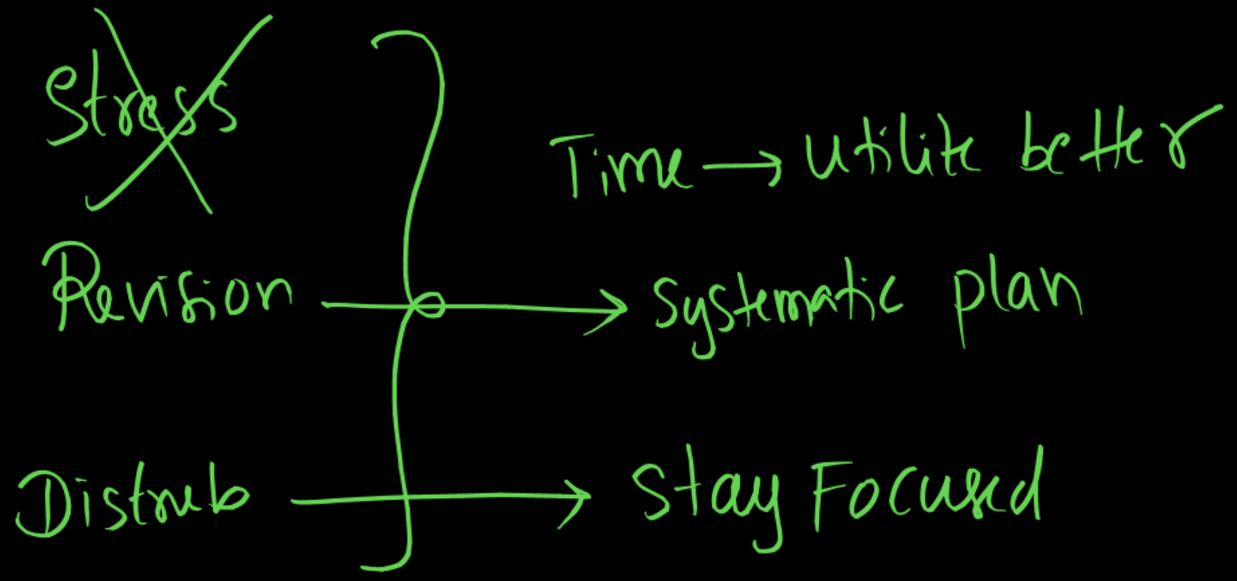
3rd year : 1st Sem

↳ Text Books ⇓ Notes } 13-15 Hr 80 days

3rd year : 2nd Sem

↳ Notes → Revise → practice } 11-13 Hr 85 days

H.W. ⇓ 210 Rank

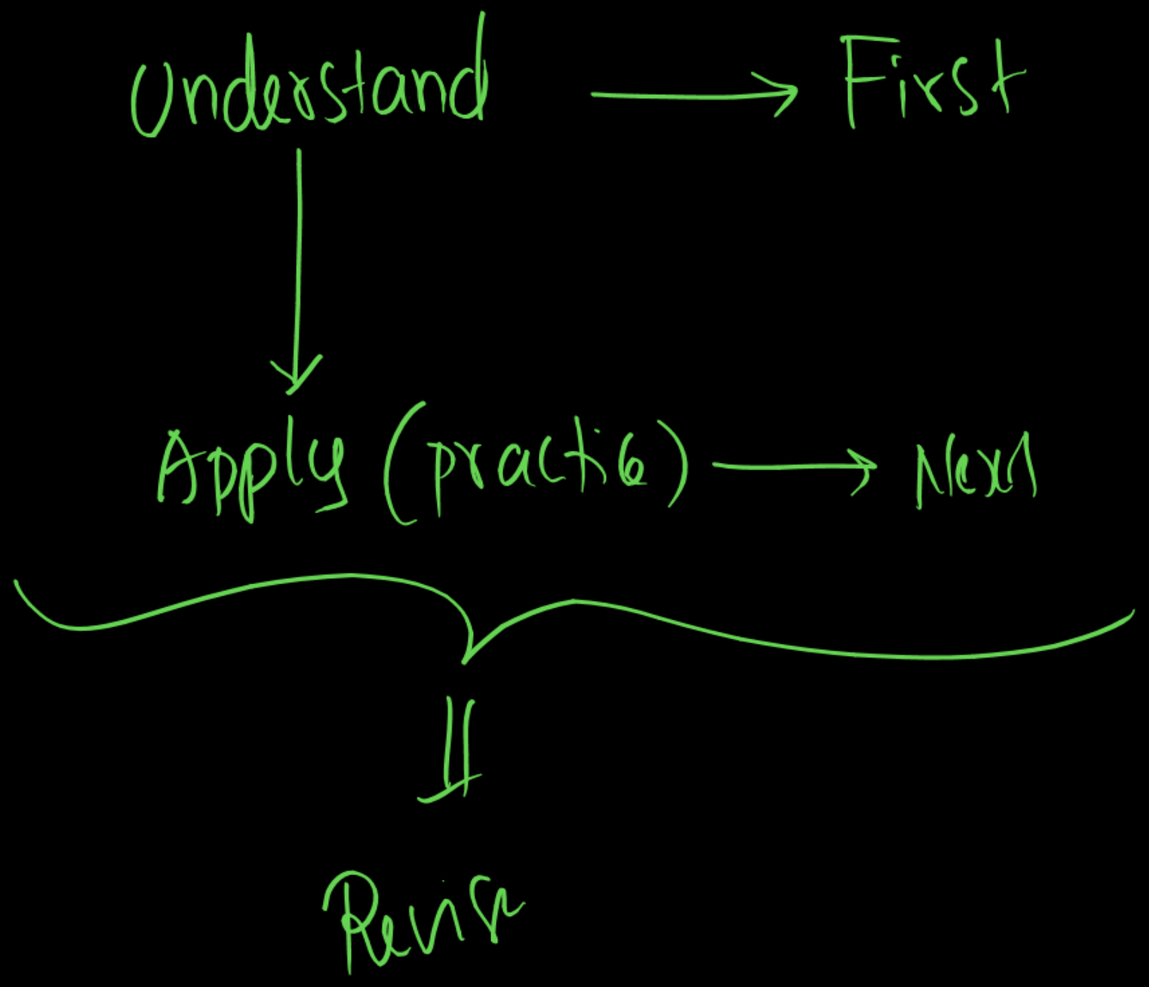


13 Hrs
my time

Ambiguous CFG
⇓
10 Hrs

6 Hrs
your time

Amb CFG
⇓
2 Hrs


Understand
First
Apply (practice)
Next
Revise

Summary

DEVA SIR PKI

Telegram group


Thank you
PW
Soldiers