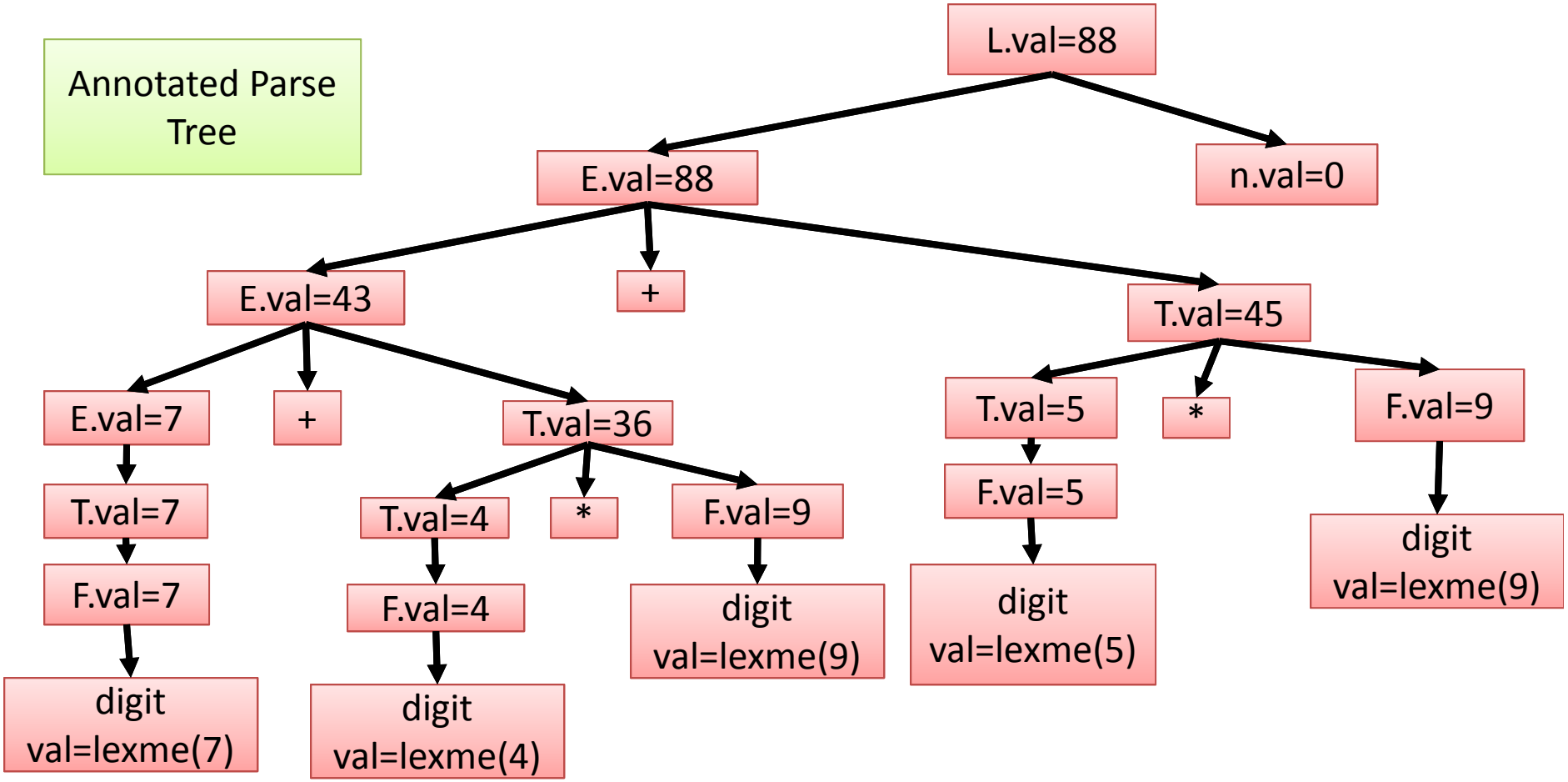
CS536

Mid-Sem-Sol

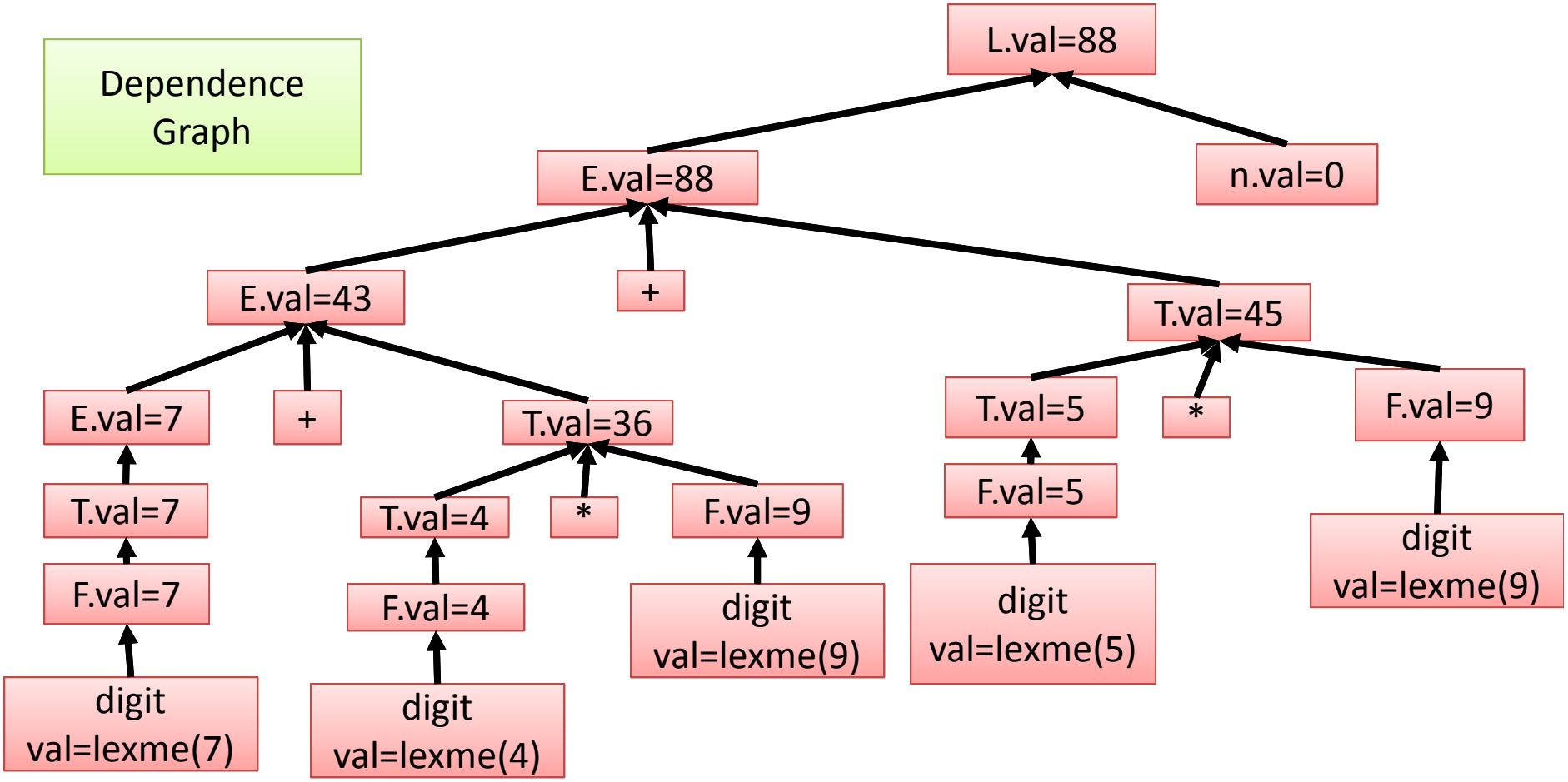
A Sahu

CSE, IIT Guwahati

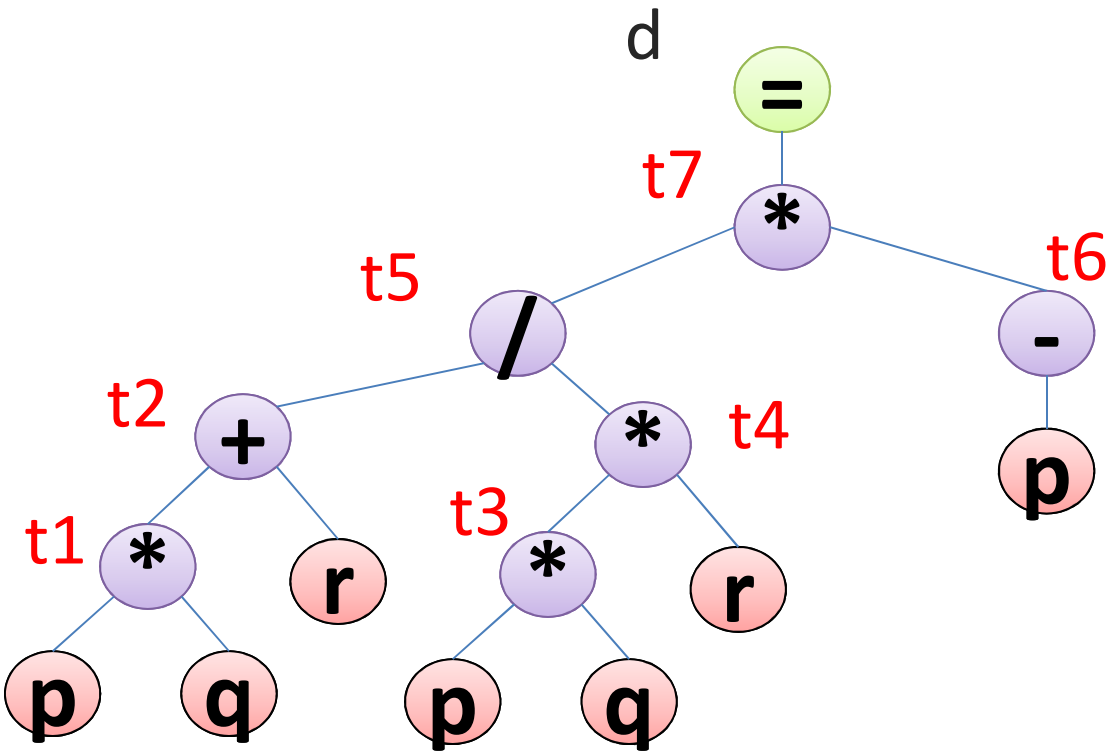
- [10 Marks] Draw **annotated parse tree** for the input string "**7+4*9+5*9**" using the following production rule. Also draw the dependency tree/graph for the same. You assume number is of single digit, ignore " and ", and **n** is end of symbol. Also assume E/E_1 are expressions, T/T_1 are terms and F/F_1 are factors.
  - **$L \rightarrow E\ n$** **$E \rightarrow E_1 + T$** **$E \rightarrow T$** **$T \rightarrow T_1 * F$**
  - **$T \rightarrow F$** **$F \rightarrow (E)$** **$F \rightarrow$ digit**



- [10 Marks] Draw **annotated parse tree** for the input string "**7+4*9+5*9**" using the following production rule. Also draw the dependency tree/graph for the same. You assume number is of single digit, ignore " and ", and **n** is end of symbol. Also assume E/E_1 are expressions, T/T_1 are terms and F/F_1 are factors.
  - $L \rightarrow E\ \mathbf{n}$ $E \rightarrow E_1 + T$ $E \rightarrow T$ $T \rightarrow T_1 * F$
  - $T \rightarrow F$ $F \rightarrow (E)$ $F \rightarrow$ **digit**



- 10=4+4+2 Marks] Draw **AST** for the statement "**d = ((p*q+r) / (p*q*r))*-p**". Assume C language precedence rule and ignore " and ". Convert the resultant AST to three address code (**TAC**) IR. Estimate the number of temporaries needed to be used for the same.




```
t1 = p * q
t2= t1* r
t3 = p*q
t4=t3*r
t5=t2/t4
t6=-p
t7=t5*t6
d=t7
```

Temp Assignment using post order LRN (Left Right Node) fashion

Require 7 temp but can be reduced to 6 for stmt 1 and 3, t1 and t3 can be merged

[10=3+3+4 Marks] Write code translation for the following statements: use C language precedence rule. You may ignore back-patching. **while (E) { if (B) S1 else S2; E=B1 || B2;}**

begin=newlabel(); E.true=newlabel(); E.false=S.next;
S_w.next=begin
S.code= label(begin)||E.code||label(E.true)
||**S_w.code** ||gen('goto' begin)

S→ while (E) S_w

Si.next=newlabel()
So.next=**Sw**.next
Sw.code= **Si.code** || label(Si.next)||**So.code**

Sw→ Si So

B.true=newlabel(); B.false=newlabel();
S1.next=S2.next=Si.next
Si.code=B.code||lebel(B.true)||S1.code
||gen('goto' S.next) ||label(B.false) ||S2.code

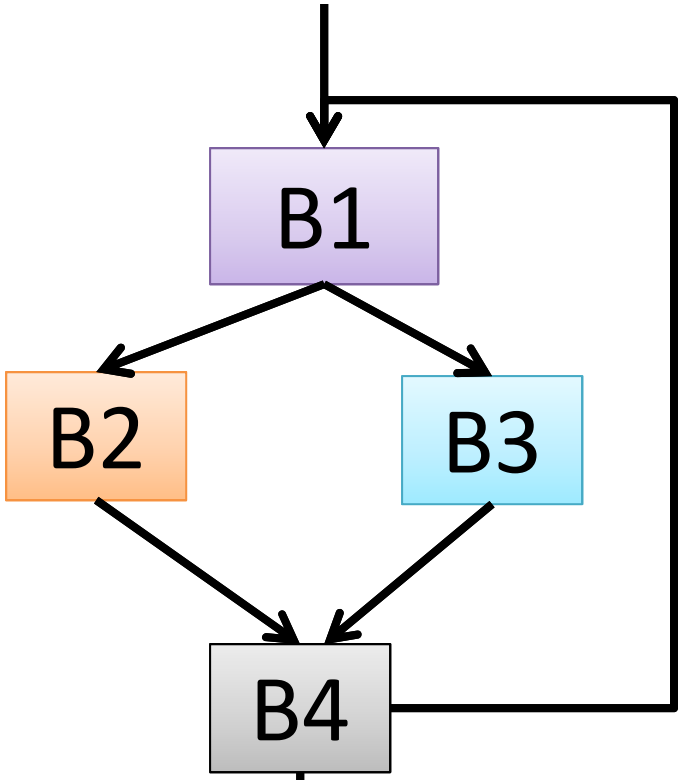
Si→ if (B) S1 else S2

B1.true=Bo.true
B1.false=newlabel()
B2.true=Bo.true; B2.false=Bo.false
Bo.code=B1.code||label(B1.false)||B2.code
So.code = Bo.code|| assign.code ("E=Bo")

So→E= B1 || B2

[10=4+3+3 Marks] Given the sequence of instructions in three address code (TAC), identify all the basic blocks; Draw control flow graph (CFG) for the code, and identify any loop structure present in the code using strongly connected components (SCC) analysis

```
L1:  add   a  b  c
     sub   d  0  a
     mul   e  d  d
     add   e  d  f
     if d<e goto      L2
     mul   f  2  e
     goto             L3
L2:  add   b  d  e
     sub   e  e  1
L3:  add   b  f  c
     if b<f goto      L1
```



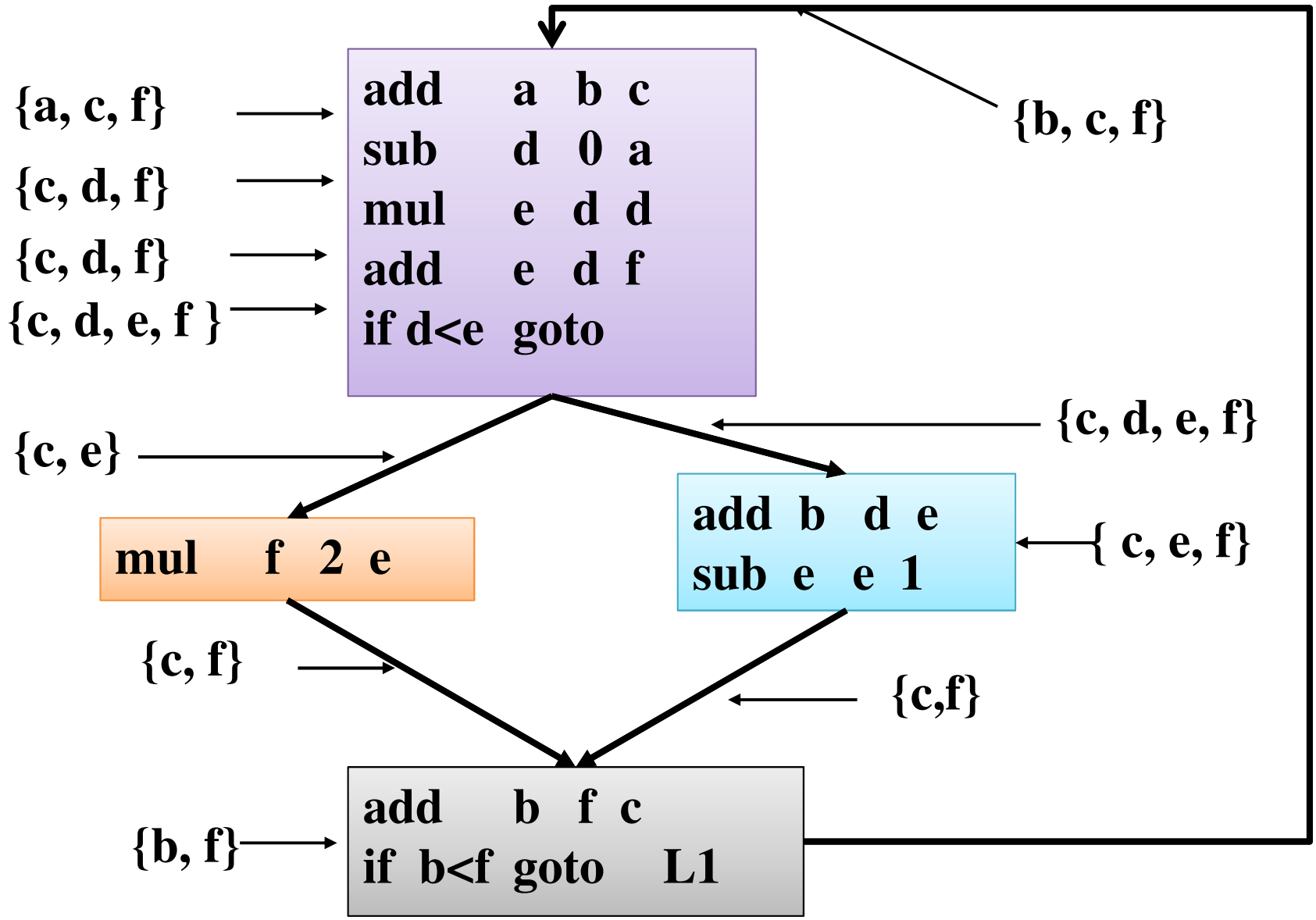
SCC: **B1**, B3, B4 with sequence

SCC: **B1**, B2, B4 with sequence

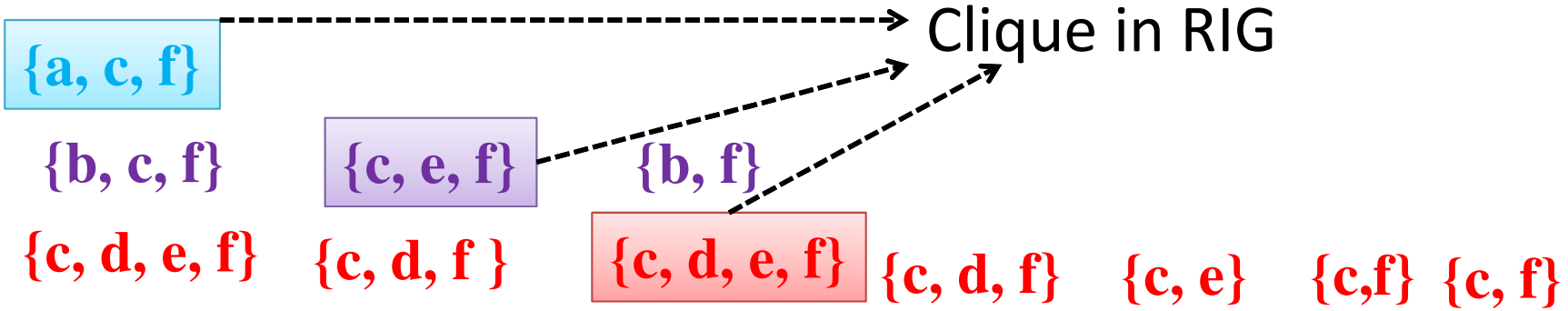
SCC: {B1, B2, B3, B4}

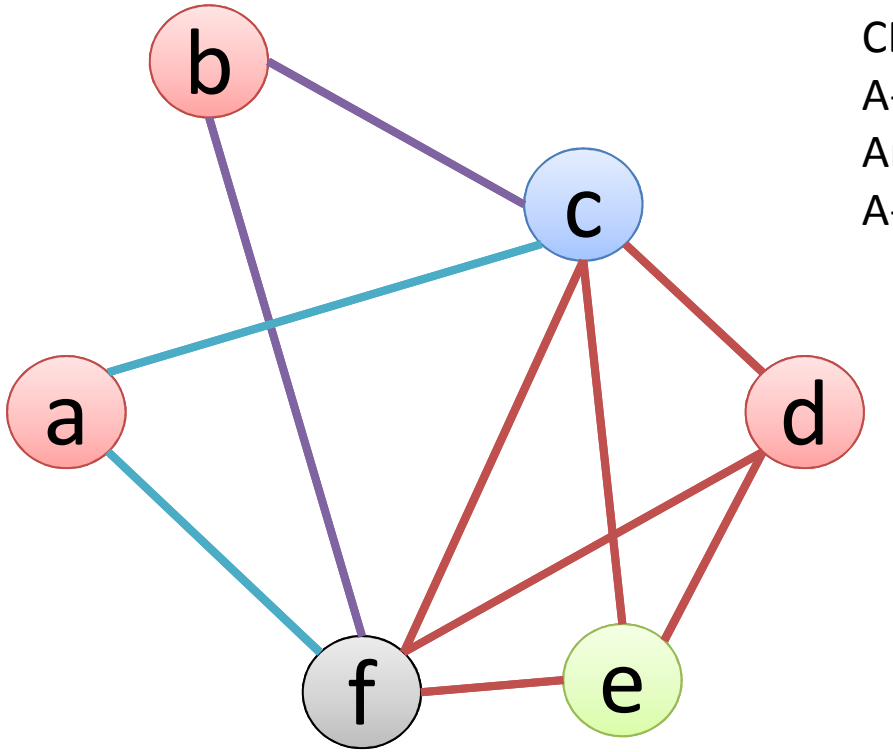
There is a single loop, B1 is unique common entry for both SCC

[10=3+3+4 Marks] For the flow graph of the previous question (Q_4) perform the variable live-ness analysis and draw the register interference graph (RIG), and find minimum number of register required for allocation without spilling.



RIG and Reg. Allocation





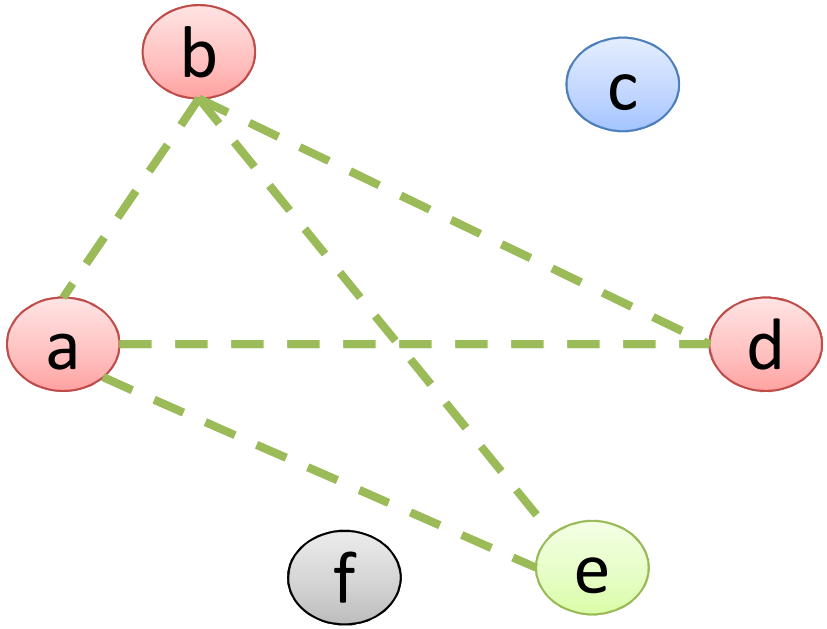
RIG: cdef clique, acf clique

CDEF : 4 Reg

A-B-D/A-B-E can share a register

Answer : 4 Color or 4 Registers

A-D share a register and B-E share another reg.



RCG: Register Compatibility Graph