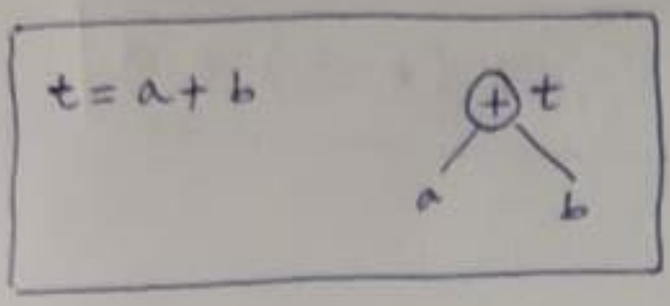
# Directed Acyclic Graph (DAG)
(a variant of Syntax Trees)

OR

## DAG representation of a basic block.

↳ sequence of consecutive statements, executed in sequential manner.




DAG represents the structure of a basic block.

Intuition

(1) In a DAG, internal node represents operator

(2) Leaf node represents identifiers, constants

(3) Internal node also represents result of expressions

### Applications of DAG

(1) Determining the common Sub Expression.

(2) Determining which names are used inside the block & computed outside the block.

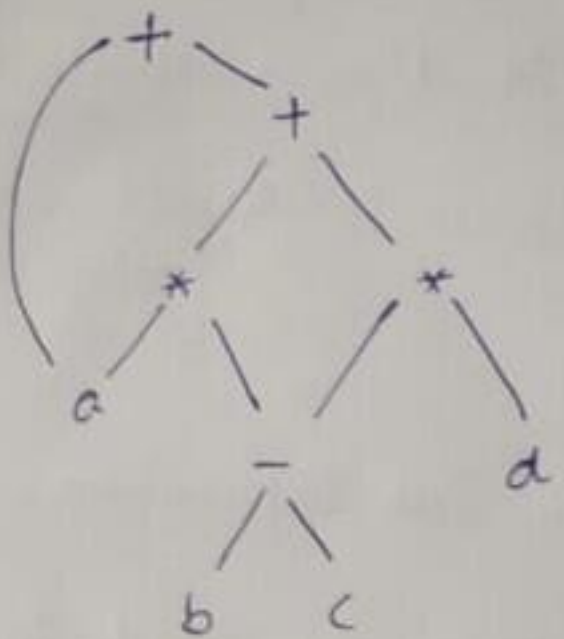
(3) Determining which statements of the block could have their computed value outside the block.

(4) Simplifying the list of Quadruples by eliminating common Sub expressions

# Construction of the DAG for the basic block

Ex 1 Construct DAG for the expression

$$a + a * (b - c) + (b - c) * d$$




① (b - c)

② a * (b - c)

③ (b - c) already there

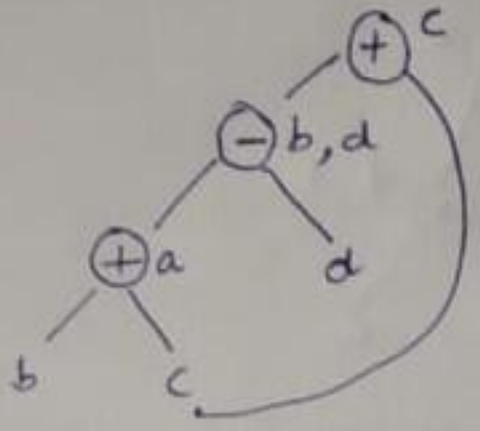
④ (b - c) * d

⑤ a * (b - c) + (b - c) * d

⑥ a + ⑤

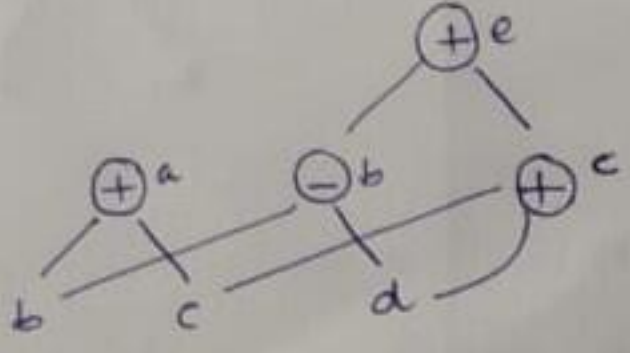
Ex2 Construct DAG for the block

1. $a = b + c$
2. $b = a - d$
3. $c = b + c$
4. $d = a - b$




Ex3 Construct DAG for the following

$a = b + c$
$b = b - d$
$c = c + d$
$e = b + c$



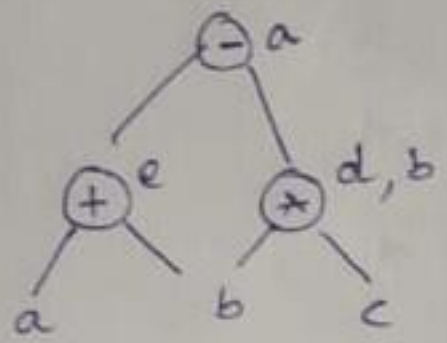
Ex 4

$d = b * c$

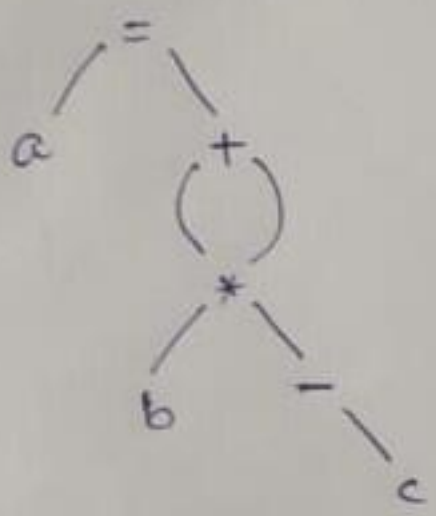
$e = a + b$

$b = b * c$

$a = e - d$




Ex 5 $a = b * -c + b * -c$




Ex 6

$a = (a * b + c) - (a * b + c)$