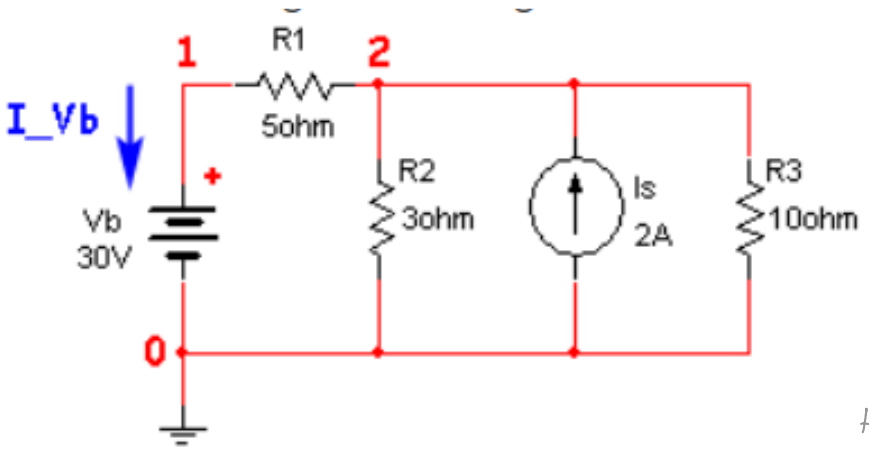

Netlist:

Vb	1	0	30	sources
Is	2	0	2	
R1	1	2	5	components
R2	2	0	3	
R3	2	0	10	
name	node1	node2	value	

type

$$\mathbf{Ax} = \mathbf{z}$$

or

$$\begin{bmatrix} \frac{1}{R1} & -\frac{1}{R1} & 1 \\ -\frac{1}{R1} & \frac{1}{R1}+\frac{1}{R2}+\frac{1}{R3} & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} v\_1 \\ v\_2 \\ I\_Vb \end{bmatrix} = \begin{bmatrix} 0 \\ Is \\ Vb \end{bmatrix}$$

1.) Construct G matrix by analysing components

(Resistors, capacitors, inductors)

let n = highest_node_number (Netlist)

let G = Matrix(n, n) ⇒ some way to construct efficient row x column matrix (all elements zero by default)

```
for com in components:
    G[com.node1 - 1][com.node1 - 1] += 1 / com.value
    if com.node2 == 0:
        continue
    else:
        G[com.node2 - 1][com.node2 - 1] += 1 / com.value
        G[com.node1 - 1][com.node2 - 1] = -1 / com.value
        G[com.node2 - 1][com.node1 - 1] = -1 / com.value
```

(C++ containers are zero indexed)

(node1 can't be 0)

return G

2.) Construct B matrix by analyzing voltage sources:

```
let n = highest_node_number (Netlist)
let m = number_of_voltage_sources (Netlist)
let B = Matrix(n, m)

for i in range(0, m):
    vs = voltage_sources[i]   → ordering of sources shouldn't matter as long as it doesn't change
    B[vs.node1 - 1][i] = 1
    if vs.node2 != 0:
        B[vs.node2 - 1][i] = -1
return B
```

3. Construct C matrix by computing B^T:

(More complex when dealing with dependent sources)

```
let C = transpose(B)   ⇒ from matrix library
```

4.) Construct D matrix:

(More complex when dealing with dependent sources)

```
let m = number_of_voltage_sources (Netlist)
let D = Matrix(m, m)   ⇒ All zero by default
return D
```

5.) Construct $\underline{X}$: (Only need this to understand meaning of solution)

```
let n = highest_node_number (Netlist)
let x = vector()

for i in range(0, n):
    x.push_back("v_{i}")
for vs in voltage_sources:
    x.push_back("I_{vs.name}")
return x^T  (Need column vector)
```

6.) Construct $\underline{Z}$:

```
let n = highest_node_number (Netlist)
let m = number_of_voltage_sources (Netlist)
let z = Array(n+m)  → size

for cs in current_sources:
    z[cs.node1 - 1] = cs.value
    if cs.node2 != 0:
        z[cs.node2 - 1] = - cs.value
for i in range(n, n+m):
    z[i] = voltage_sources[i]
return z^T  (Need column vector)
```

7.) Combine C, B, C, D into matrix A:

Probably not very efficient ⇒ Create matrix A in beginning and construct in place

(Identical algorithm but avoids construction of extra matrixes)

8.) Compute inverse of A:

Use library