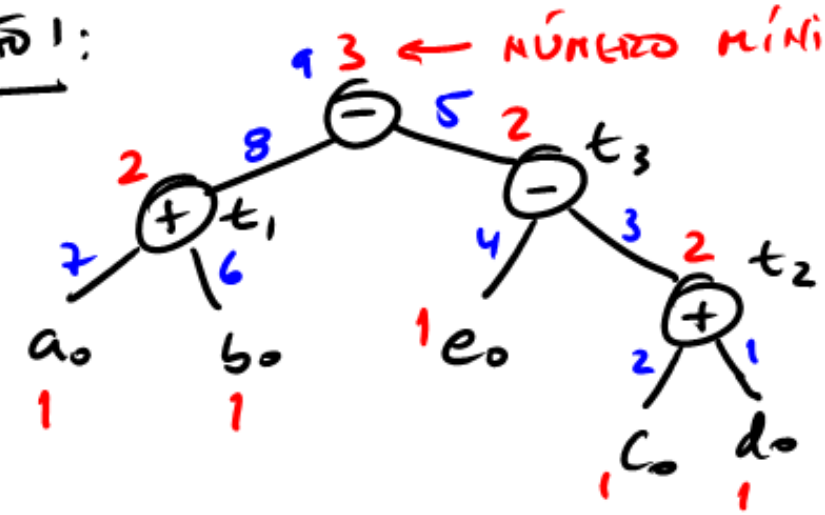
mc910-2s10-gabarito-ex2

QUESTÃO 1:



$$l = \begin{cases} \max(l_1, l_2), l_1 \neq l_2 \\ l_1 + 1, l_1 = l_2 \end{cases}$$

(a) USANDO O ALGORITMO DE SETHI-ULLMAN PARA ANOTAR EM VERMELHO O NÚMERO DE REGISTRADORES NECESSÁRIOS EM CADA SUB-ÁRVORE. NOTA: FOLHAS NA MEMÓRIA.

(b) A ORDEM DO ESCACONAMENTO É DADO EM AZUL E SEGUE PRIMEIRO PARA A SUB-ÁRVORE QUE DEMANDA MAIS REGISTRADORES

(c) NÚMERO DE REGISTRADORES É 2.

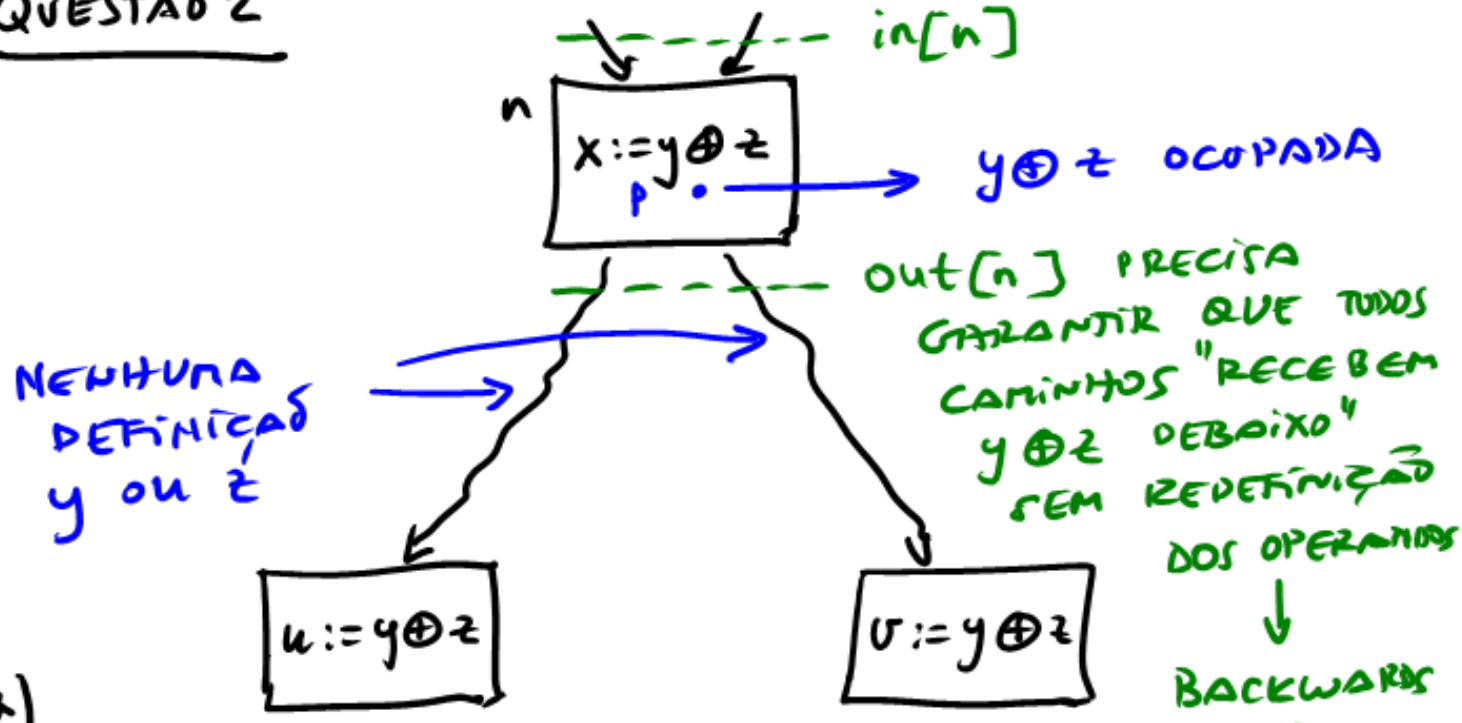
PRECISO FAZER SPILL !!

ESCOLHIDO O VERTICE t_3.

```
r1 := M[d0]
r2 := M[c0]
r1 := r1 + r2
r2 := M[e0]
r1 := r2 - r1
M[t3] := r1      <- SPILL
r1 := M[b0]
r2 := M[a0]
(*)
```

```
(*) r1 := r1 + r2
    r2 := M[t3]  <- RELOAD
    r1 := r1 - r2
```

QUESTÃO 2



(a)

$$\begin{cases} out[n] = \bigsqcap_{s \in succ[n]} in[s] \\ in[n] = gen[n] \cup (out[n] - kill[n]) \end{cases}$$

(b) NÃO FORAM SOLICITADAS TODAS AS ITERAÇÕES, SOMENTE O RESULTADO FINAL, QUE PODE SER OBTIDO DE FORMA "AD-HOC" DO CFG.

	in[n]	out(n)
1	—	a+b
2	a+b	a+b, c-a
3	a+b, c-a	a+b, c-a
4	a+b, c-a, c+1	a+b, c-a
5	a+b, c-a	—
6	b*d	—

INÍCIO

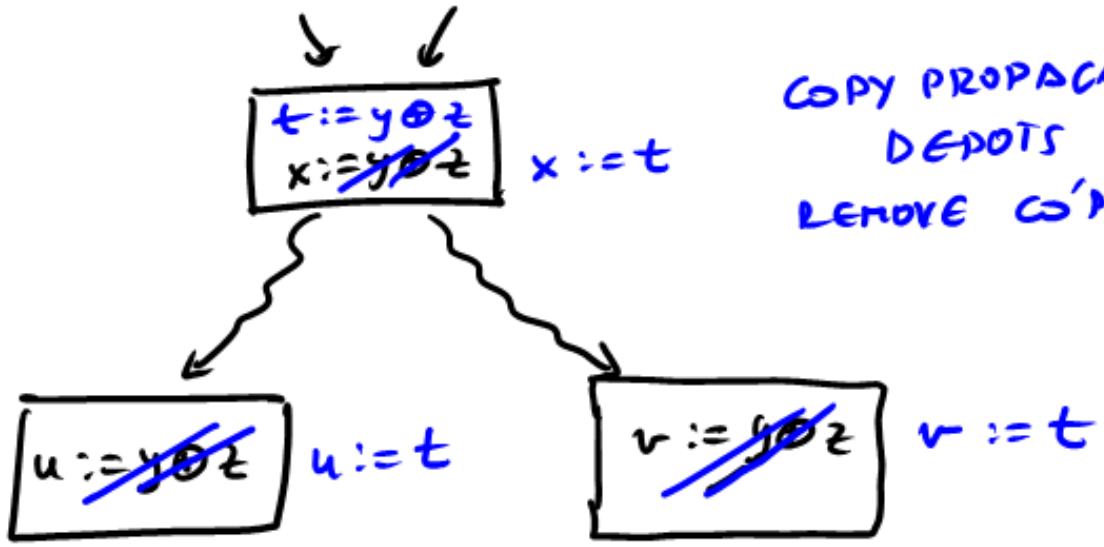
$out[B_c] = \emptyset$

$in[B_c] = gen[B_c]$

$\forall B_i \neq B_c$

$in[B_c] = \bigcup - kill[B_i]$

(c)



COPY PROPAGATION DEPOIS REMOVE CÓPIAS

A OTIMIZAÇÃO É CHAMADA DE "CODE HOISTING". PERMITE REDUZIR O TAMANHO DO PROGRAMA, HABILITAR CODE MOTION PARA FORA DE LAÇOS, REALIZAR EXECUÇÃO MAIS CEDO NO PIPELINE, ETC...

QUESTÃO 3

(a)

for each block n do in[n] = ∅;
while changes to any of in's occur do
    for each block n do {
        $out[n] = \bigcup_{s \in succ[n]} in[s]$
        in[n] = use[n] ∪ (out[n] - def[n])
    }

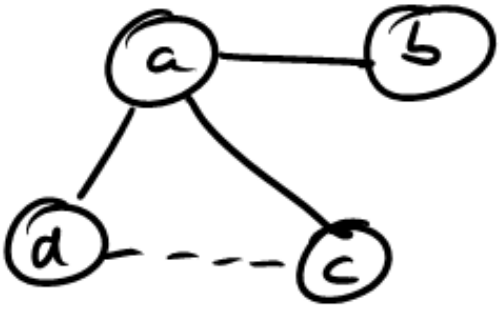
(b) ORDEM DOS BLOCOS É REVERSA
B_3, B_2, B_1 e B_0

				1		2
	use[n]	def[n]	in[n]	out[n]	in[n]	out[n]
0	—	a	—	a	—	a
1	—	b, d	a	a, d	a	a, d
2	—	c, d	a	a, d	a	a, d
3	a, d	—	a, d	—	a, d	—

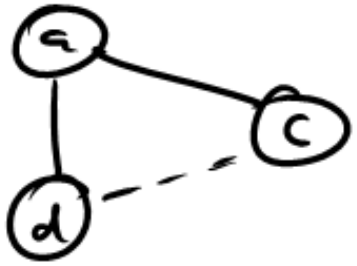
(c)

① PROPAGAR out[n] PARA DENTRO DE CADA BLOCO n.

② CONSTRUIR GRAFO DE INTERFERÊNCIA



③ DEGREE (b) = 1 < 2
SIMPLIFY



④ DEGREE (a) = 2 ≥ 2 → NAO SIMPLIFY

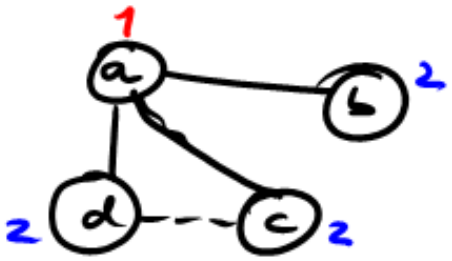
⑤ DEGREE (c&d) = 1 < 2 → COALESCING POR BRIGGS

⑥ DEGREE (a) = 1 < 2 → SIMPLIFY a

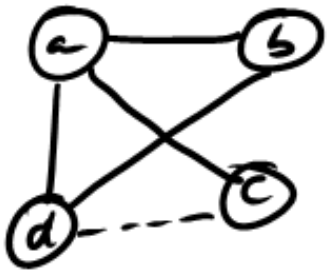
c&d

⑦ DEGREE (c&d) = 0 < 2 → SIMPLIFY c&d

⑧ POP (SELECT &c)



(d) TROCA DE INSTRUÇÕES → NOVO GRAFO



① DEGREE(a) = DEGREE(b) = 2 ≯ 2
NÃO CONSEGUE SIMPLIFY

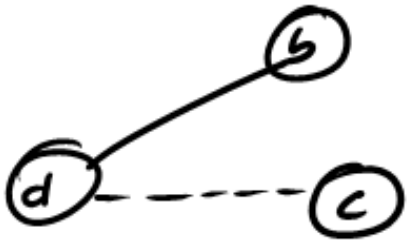
② DEGREE(c&d) = 2 ≯ 2
NÃO CONSEGUE COALESCING (BRIGGS)

③ a VIZINHO c&d } NÃO CONSEGUE COALESCING
DEGREE(b) = 2 ≯ 2 } (GEORGE)

④ IDENTIFICAR SPILL ; $\text{CUSTO}(n) = \frac{\text{USOS}(n)}{\text{degree}(n)}$

a : 2/3 = 0.6
b : 2/2 = 1
c : 2/1 = 2
d : 3/2 = 1.5

} ESCOLHE-SE a.

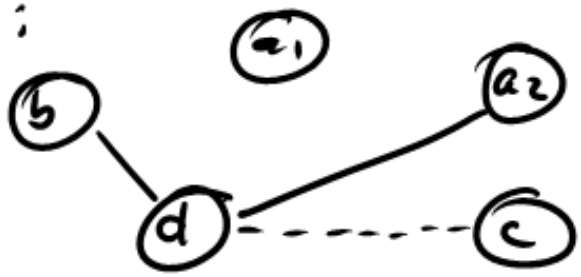


⑤ REESCREVENDO CÓDIGO

B_0 : $a_1 \leftarrow \ldots$
$M[a] \leftarrow a_1$

B_3 : $a_2 \leftarrow M[a]$
$\ldots \leftarrow a_2$
$\ldots \leftarrow d$

NOVO GRAFO:



⑥ RECOMEÇANDO ...

⑦ DEGREE $(a_1) = 0 < 2$

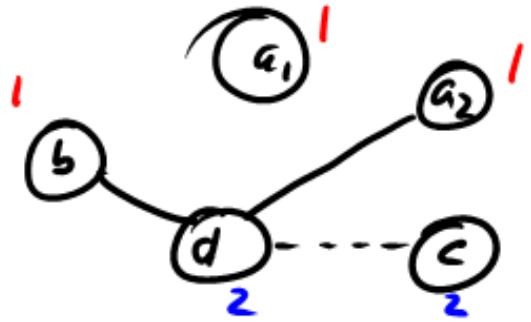
DEGREE (a_2) = DEGREE $(b) = 1 < 2$

$\Rightarrow$ SIMPLIFY a_1, a_2, b

⑧ COMESCE c&d, QUALQUER HEURÍSTICA

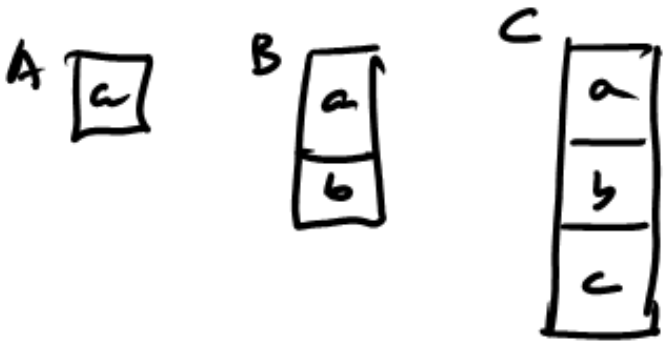
⑨ DEGREE $(c\&d) = 0 < 2 \Rightarrow$ SIMPLIFY

⑩ POP (SELECT) :



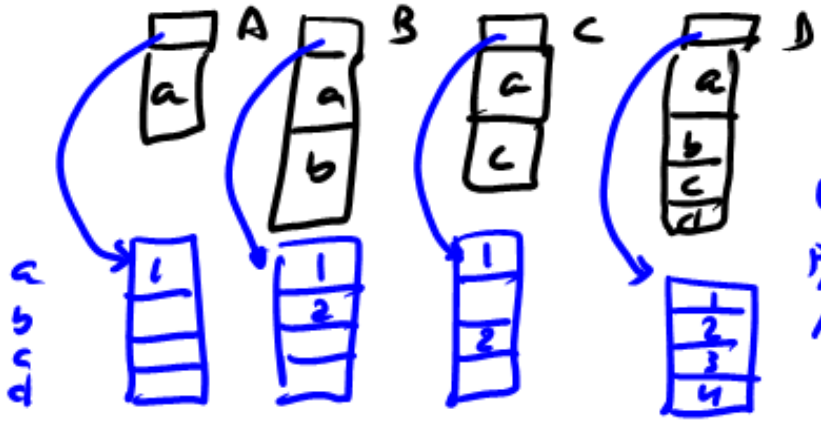
QUESTÃO 4

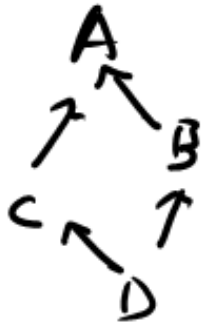
- JAVA USA HERANÇA SIMPLES



PREFIXAÇÃO RESOLVE

- C++ USA HERANÇA MÚLTIPLA





USAR COLORAÇÃO P/ ALOCAR CAMPOS A SLOTS DO DESCRITOR