3-2

Saturation = how many registers you <u>cannot</u> use

"graph coloring"

W : queue

W ← vertices (G)

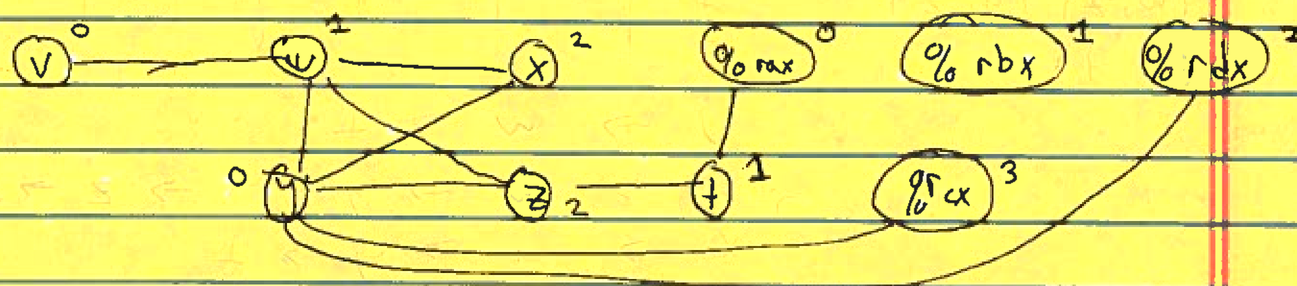
while W is not empty

  select vertex v from W where sat(v) is maximum

  find the lowest color that is not in { color(u) | u ∈ adj(v) }

  color[v] ← c   move-biasing : { c(u) | u ∈ M(v) }

  remove v from W   − sat(v) ≠ look in here first



coloring : G → assignment (V → C)

  × (V → C)

  × M

C → reg or stack   (Color → Xarg)

σ   for i = 0 to 13

record how many stack vars   & r = rbx ... r15

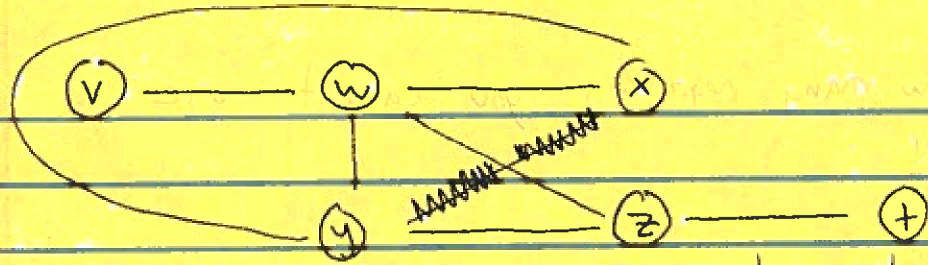
  σ[i] = r

  for i = 14 to biggest number in C

  σ[i] = stack var (i − 13)   %rsp(8×i)

ξ(v) = σ[c(v)]

6-1



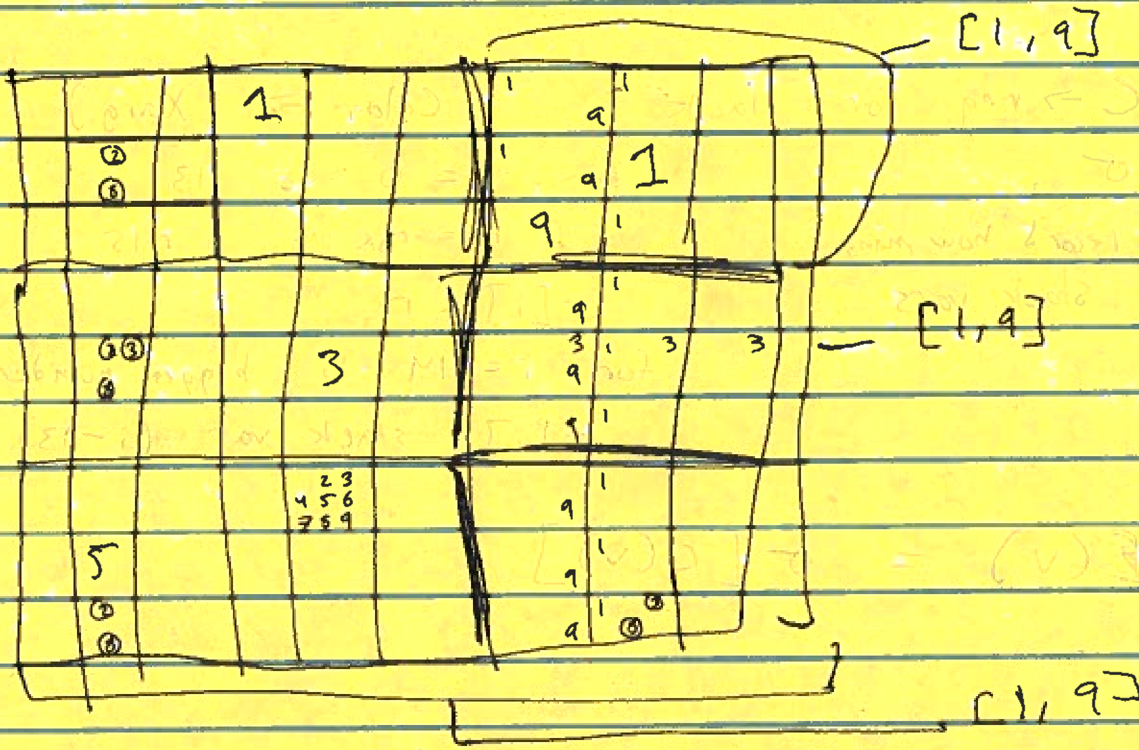
Adjacency Matrix

	v	w	x	y	z	t
v		x				
w	x		x	x	x	
x		x		x		
y		x	x		x	
z		x		x		x
t					x	

Adjacency List
(v, w) (w, x)
(w, y) (x, y)
(w, z) (y, z)
(z, t)

v → w → ⊥
w → v → y → x → z → ⊥
x → w → y → ⊥
y → w → x → z → ⊥
z → w → y → t → ⊥
t → z → ⊥

"Pencil marks"
⇒ record can/can't infor



-4 typec : e ⇒ ~~Bool~~ ty (or error)

typec Γ e ⇒ ty

$\Gamma \vdash e : ty$

$\vec{(V \Rightarrow ty)}$

$\Gamma \vdash x : \Gamma(x)$ $\quad \Gamma \vdash t/f : Bool$

$\Gamma \vdash n : S64$

$$\frac{\Gamma \vdash e_L : S64 \quad \Gamma \vdash e_R : S64}{\Gamma \vdash (+\ e_L\ e_R) : S64}$$

typec Γ (+ e_L e_R) =
  ty_L = typec Γ e_L
  ty_R = typec Γ e_R
  if ty_L != S64, error
    ty_R != S64, error
  S64

$$\frac{\Gamma \vdash e_L : S64}{\Gamma \vdash (-\ e_L) : S64}$$

$$\frac{\Gamma \vdash e_L : S64 \quad \Gamma \vdash e_R : S64}{\Gamma \vdash (cmp\ e_L\ e_R) : Bool}$$

$$\frac{\Gamma \vdash e_c : Bool \quad \Gamma \vdash e_T : ty_R \quad \Gamma \vdash e_F : ty_R}{\Gamma \vdash (if\ e_c\ e_T\ e_F) : ty_R}$$

$$\frac{\Gamma \vdash e_A : Bool}{\Gamma \vdash (not\ e_A) : Bool}$$

$$\frac{\Gamma \vdash xe : T_x \quad \Gamma[x \to T_x] \vdash b : T_b}{\Gamma \vdash (let\ x = xe\ in\ b) : T_b}$$

6-3

main BEGIN ⇒ pushq rbp
movq rsp, rbp

FC = now how many stack-vars rather how ~~many~~ vars

← save

subq FC, rsp
jmp BODY

END ⇒ addq FC, rsp ← restore
popq rbp

callee = rsp, rbp
r12 - r15

save = pushq r12, push r13, ... push r15
restore = pop r15, ... pop r12

R_2 := e := .... | true | false | (and e e)
| (or e e) | (not e) | (cmp e e)
| (if e e e) | (- e e)
cmp := == | < | ≤ | ≥ | >
ty := S64 | Bool

interp σ true = true
false = false
(and e_L e_R) = (if e_L e_R false)
(or e_L e_R) = (if e_L true e_R)
x = y || z

interp σ (- e_L e_R) =
(in σ e_L) − (in σ e_R)
= interp σ (+ e_L (- e_R))
Expr * BinSub (E* L, E* R) =
new BinAdd (L, new UnNeg(R))

I σ (not e_A) = if (I σ e_A) false o.w. true
I σ (if e_C e_T e_F) = if (I σ e_C) then
I σ e_T
o.w. I σ e_F
I σ (cmp e_L e_R) =
(I σ e_L) cmp (I σ e_R)

(1 ≤ true)