15-750
2/5/16

Strongly Connected Components

Input: Directed graph $G=(V,E)$

$v, w \in V$

Def $v \equiv w$ if $\exists$ path from v to w and
$\exists$ path from w to v

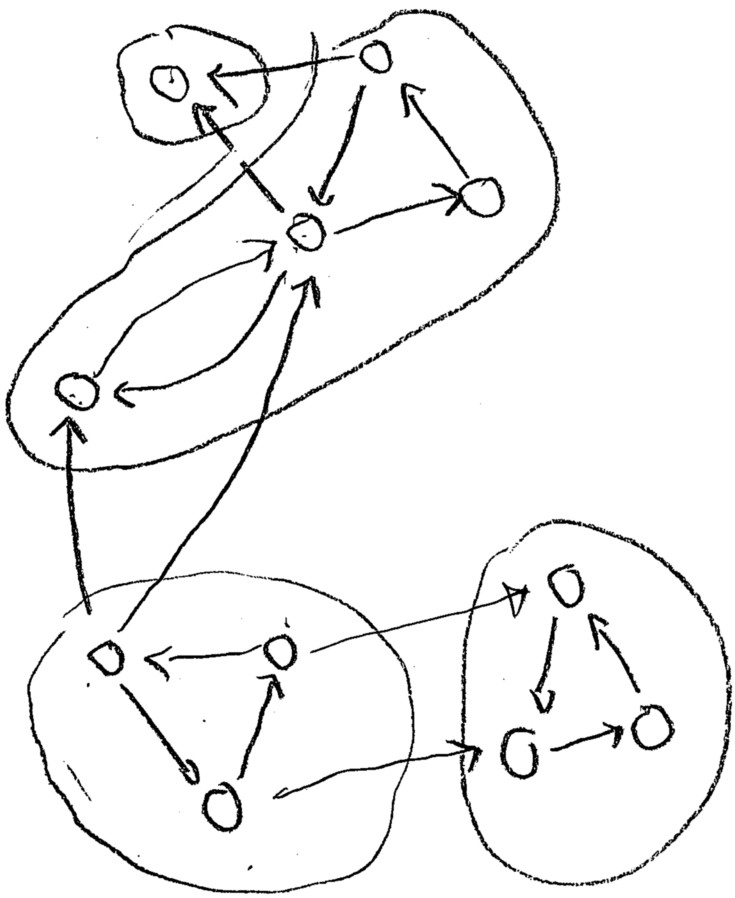
Def R on a set X is an equivalence relation

Reflexive: wRw
Symmetric: $vRw \Rightarrow wRv$
Transitive: $vRw \wedge wRx \Rightarrow vRx$

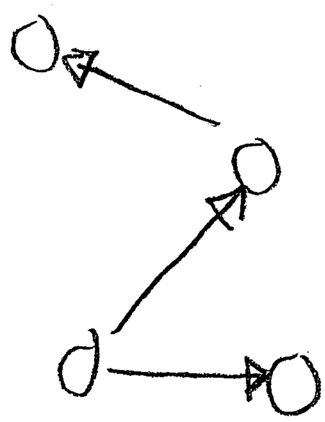
Claim $\equiv$ is an equivalence relation on V

Def The equivalence class are the strongly connected components or strong component

An Example:



Condensation



Alg: DFS(G)

1) $\forall u \in V$ color(u) ← white ; time ← 0

2) $\forall u \in V$ <u>if</u> color(u) ≡ white <u>then</u> DFS-visit(u)

what order?

DFS-visit(u)

1) color(u) ← gray ; push-time(u) ← time++

2) $\forall v \in$ adj(u)

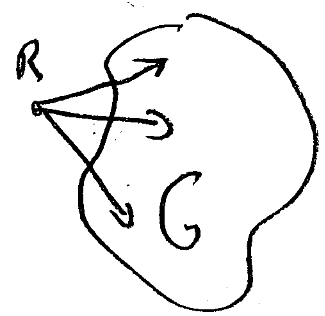
<u>if</u> color(v) ≡ white <u>then</u> DFS-Visit(v)

3) color(u) ← black ; pop-time(u) ← time++

note: dfs(u) ≡ push-time(u)

Conceptually Simplify DFS

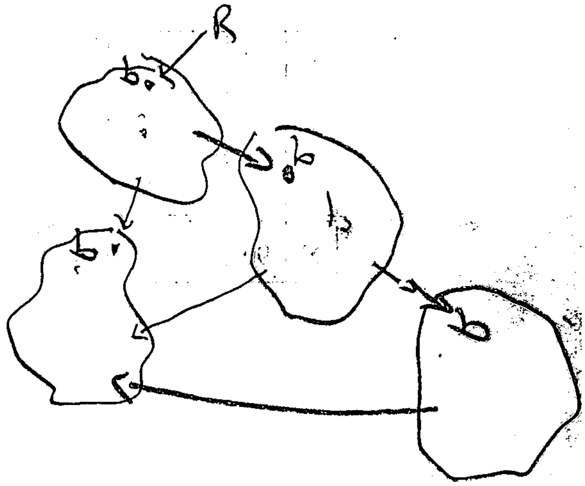
Add a super vertex R

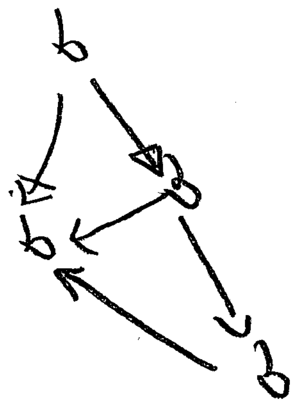


1) Add new vertex R
2) Add new edges from R to all old vertices
3) run DFS(R)

<u>Fix the DFS</u>

<u>Def</u> For each SC the first vertex visited $b \in SC$ is called the <u>base</u> node of SC.





1) In our case, DFS Forest is a Tree rooted at r.
T is an ordered Tree by push-time.

2) We can ignore forward edges! why?

3) All cross edges go from right to left.

Def Let T be our DFS Tree
$T_v \equiv$ subtree rooted at v.

Lemma 1 If $T_v \cap T_w = \emptyset$ then
$pushtimes(T_v) < pushtimes(T_w)$ or
$pushtimes(T_w) < pushtimes(T_v)$

Pf Note: pushtimes are just
preorder times for T.

<u>Lemma 2</u> If b is the base for X, a SC, then

1) If P is a path from b to $v \in X$ & $w \in P$
   then $w \in X$
2) If $v \in X$ then v is a descendant of b in T.

<u>Pf</u>

<u>Claim 1</u> Suppose $w \in P$, a path from b to v.

a) bPw is a path from b to w
b) the path from w to b will consist of
   i) wPv in P.
   ii) path from v to b (by hypothesis).

Pf of claim 2

Will show: P is a path from b to v then

$$V(P) \subseteq T_b$$

Pf induction $|P|$

1) $|P| = 1$ then done

Consider first edge $e = (x, y)$ of P st $y \notin T_b$

Then e is either a cross edge

or a back edge

In either case $push(y) < push(b)$

A contra!

Finding the Base Nodes

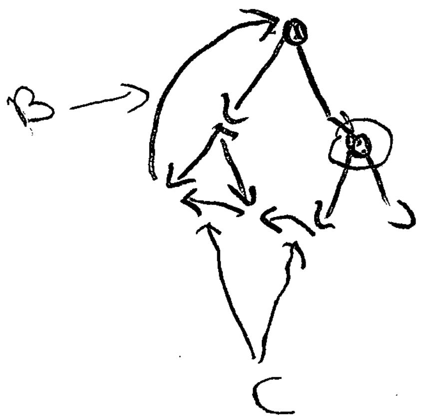
Note: SC are just the forest of T obtained by removing the edge into each base node.

Idea: Use something like lowpoint from Biconnected components

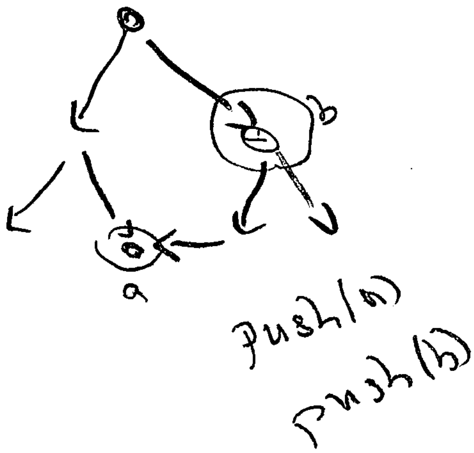
Prob: Tree, Back, Forward edge are OK

Cross edges are a problem.

eg need them



They can give wrong answer



Solution: Determine when a Cross edge
Leaves the SC.

Remove each SC when base is poped.

Main Tool:

$u \in X \subseteq V$ X is SC,

$\text{lowlink}(u) = \min\{\text{push}(v) \mid v \in \text{Path of form } T^*[C,B]^+ \;\&\; \underline{\underline{v \in X}}\}$

Claim: $\text{lowlink}(u) \geq \text{push}(u)$ iff u is a base node.

Component(G)

1) Init(S: Stack), time ← 0, ∀v color(v) ← white

2) If color(u) = white then Strong(u)

Strong(u)

1) color(u) ← gray ; dfs(u) ← lowlink(u) ← time++

push(u, S)

2) ∀v ∈ Adj(u)

if color(v) = white then

strong(v) ; ll(u) = min{ll(u), ll(v)}

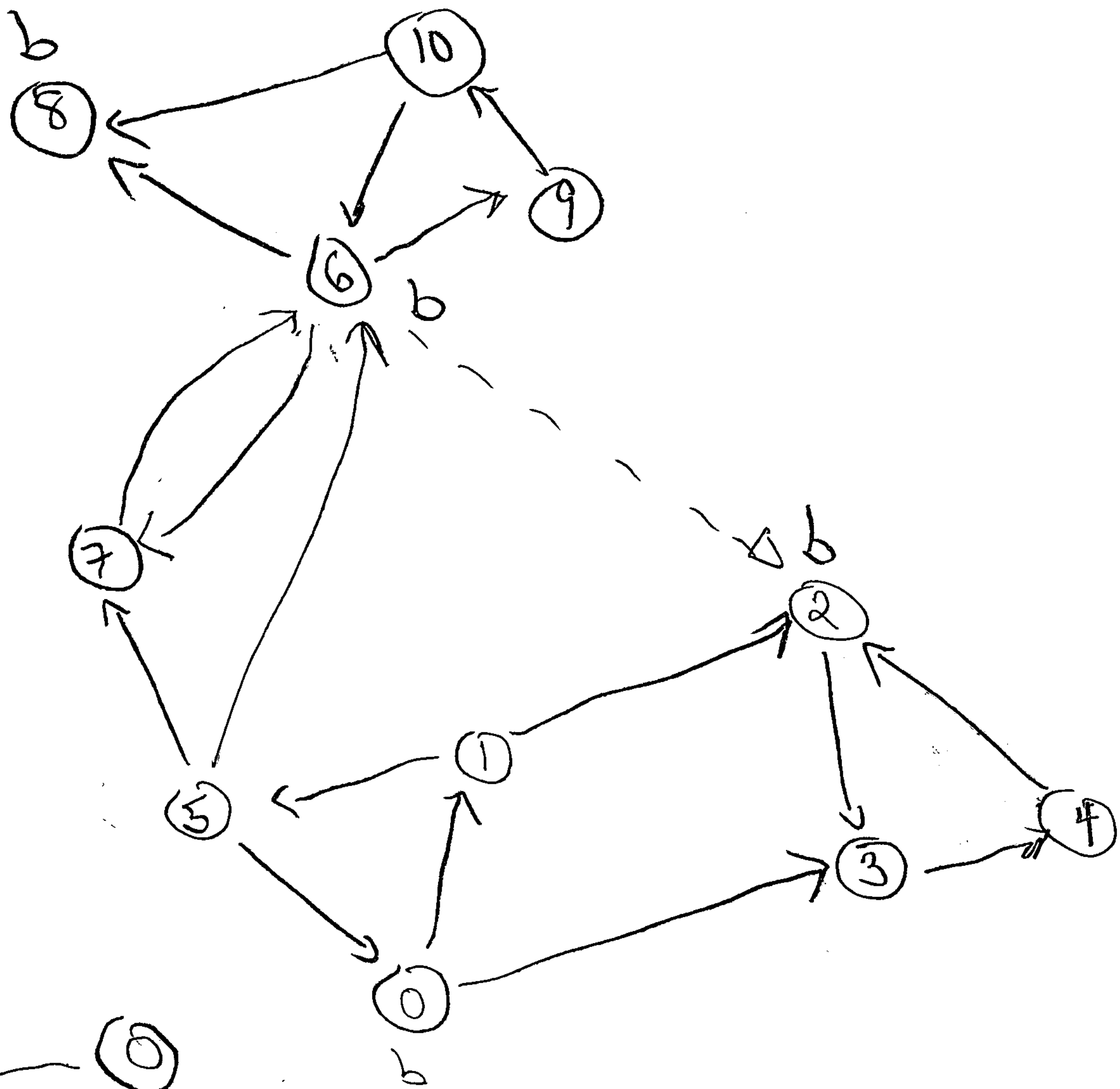
else if [v ∈ S]

then ll(u) ← Min{ll(u), dfs(v)}

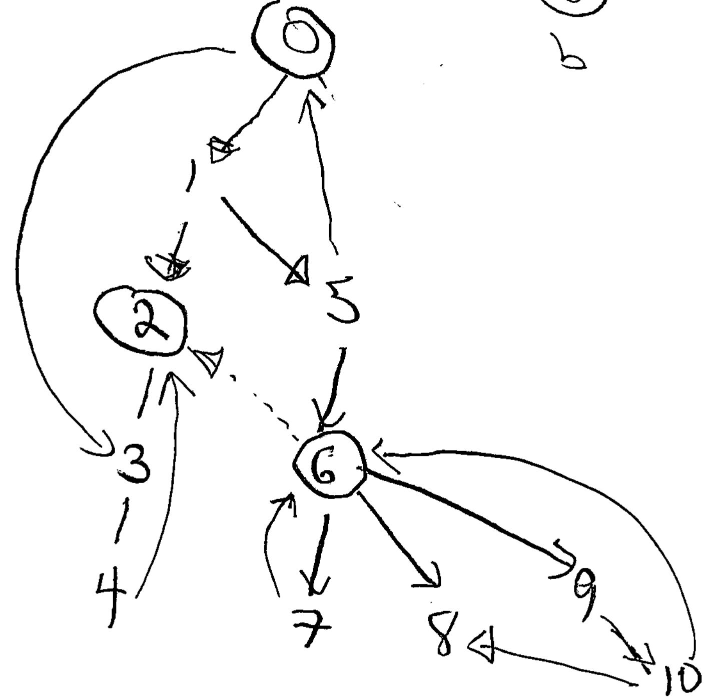
3) if ll(u) = dfs(u)   [u is a base vertex]

then while [top(S) ≠ u] pop(S) ; pop(S)

end Strong



b
8
10
6
9
b
7
b
2
1
5
4
3
0
b



0
1
2
5
3
6
4
7
8
9
10