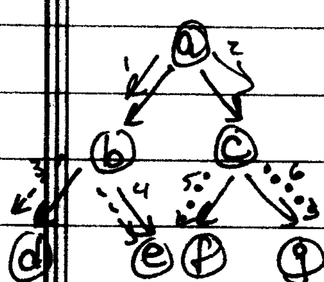
Last time: Breadth-First Search
This time: Depth-First Search

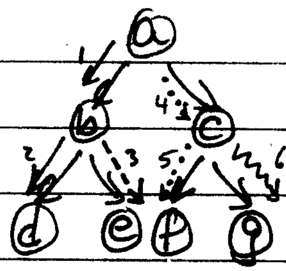
Breadth First



FIRST EXAMINE THE ENTIRE ADJACENCY LIST OF ONE VERTEX BEFORE EXAMINING ADJACENCY LISTS OF DESCENDANTS
- traverse breadth before depth

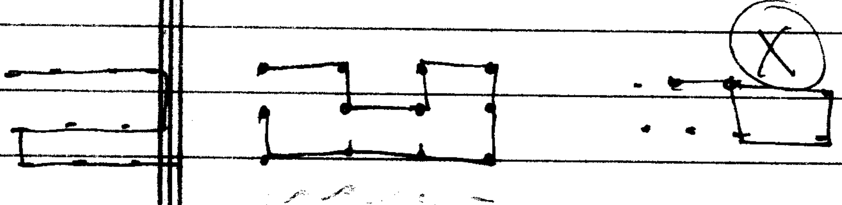
examining all of a's children before examining grandchildren

Depth-First



Progress recursively through adjacency lists.
- fully explore descendants of a's left child before examining right child.
- at every step, try to go deeper
- when can't, then backtrack and follow branches.

→ Example: Protein conformational search



ALGORITHM

Analysis

DFS(G)

for each u ε V
  do color[u] ← white
time ← 0
(initialize all vertices white, time zero)

for each vertex u ε V
  do if color[u] = white
    then DFS-VISIT(u)
(Check each vertex and explore only white ones.)

$\Theta(V)$

$\Theta(V)$, not including DFS-VISIT calls

Called |V| times in aggregate; called once for each white vertex which is immediately grayed

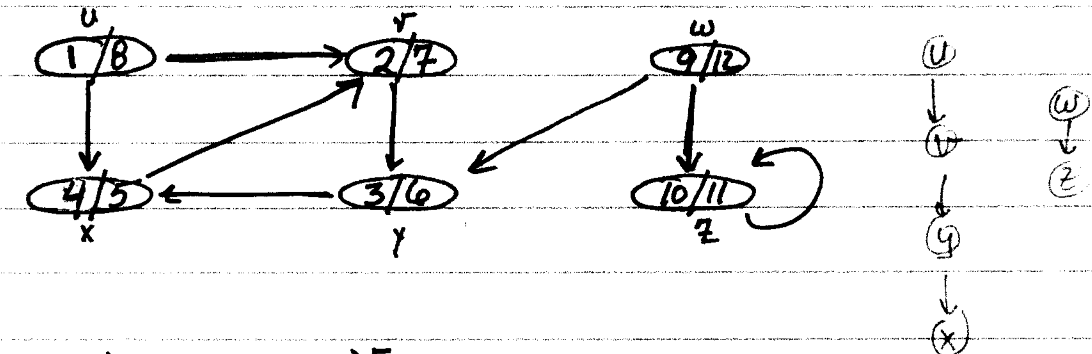
DFS-VISIT(u)

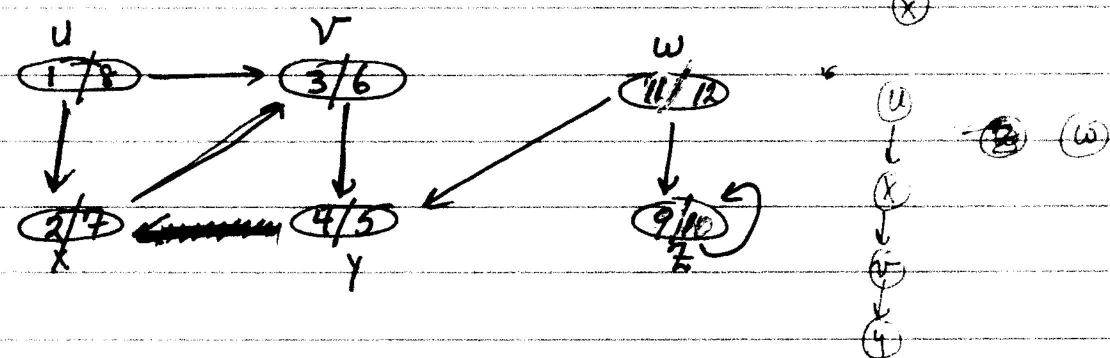
color[u] ← gray (gray each vertex as start to explore)
time ← time + 1 (← update time)
d[u] ← time (← discovery time)
for each v ε Adj[u]
  do if color[v] = white
    then DFS-VISIT(v)
(check adjacency list and explore unexplored members via depth first search)
color[u] ← black (when done with adjacency list, mark found black)
f[u] ← time ← time + 1 (record finishing time for black vertex)

Called |E| times in aggregate ⇒ $\Theta(E)$

Total: $\Theta(V + E)$ ← linear

Example: discover/finish times





Properties

~~① Vertices traversed in search form depth-first forest of trees~~

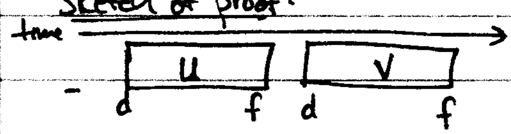
② Discovery and finishing times have parenthesis structure.

That is, representing discovery by "(u" and finishing by "u)" ⟹ Properly nested structure

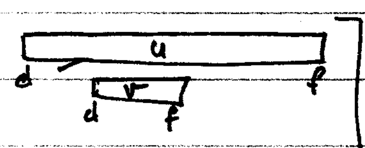
(u (v (y (x x) y) v) u) (w (z z) w)

(u (x (v (y y) v) x) u) (z z) (w w)

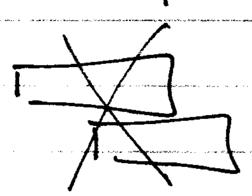
Sketch of proof:



If $d[v] > f[u]$, then disjoint and neither vertex discovered while other was gray. Neither is descendant of other.



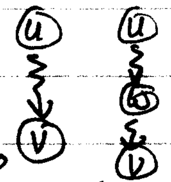
If $d[v] < f[v]$, then v is a descendant of u. As such, all of its adjacency list will be finished ~~before that of u~~, so $f[v] < f[u]$ ⟹ One interval inside other



In fact, this condition ⟺ v is a descendant of u in the depth-first forest

③ The White-Path Theorem: In a DFS of (directed or undirected graph), vertex v is a descendant of vertex u iff at time $d[u]$, vertex v can be reached by a path consisting entirely of white vertices.

Sketch of Proof: ($\Rightarrow$)

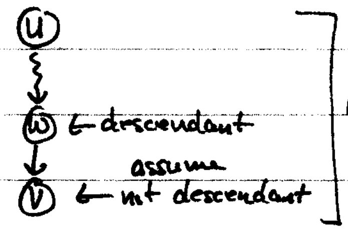


v is a descendant of u

w is any path along u→v route. w also a descendant of u.

Corollary of ② states $d[w] > d[u]$, so all w ($\neq v$) were white when u discovered

($\Leftarrow$)



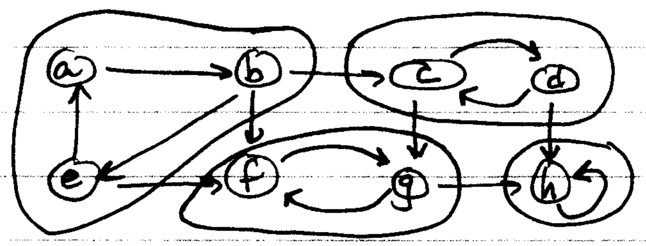
$f[w] \le f[u]$ Corollary of ②

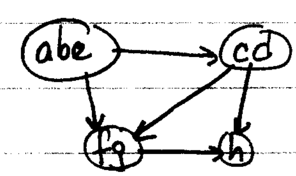
v must be discovered after u discovered ~~but before w finished~~

$d[u] < d[v] < f[w] \le f[u]$

Property ⑥ $\Rightarrow$ $\boxed{\;u\;}$ $\boxed{v}$ & v is descendant of u

Definition: Strongly Connected Component of digraph $G=(V,E)$ is maximal set of vertices $C \subseteq V$ such that all pairs of vertices of C are reachable from each other. (mutually reachable)





Not NDO

In analysis of certain types of engineering systems, want to examine state space and find whether appropriate states of system are reachable from one another and others appropriately unreachable.

<u>Note:</u> G^T is transpose of $G=(V,E)$. $G^T=(V,E^T)$, so all edges have direction reversed. Can compute in $O(V+E)$ from adjacency list. G and G^T have same strongly connected components.

→ STRONGLY-CONNECTED-COMPONENTS(G)

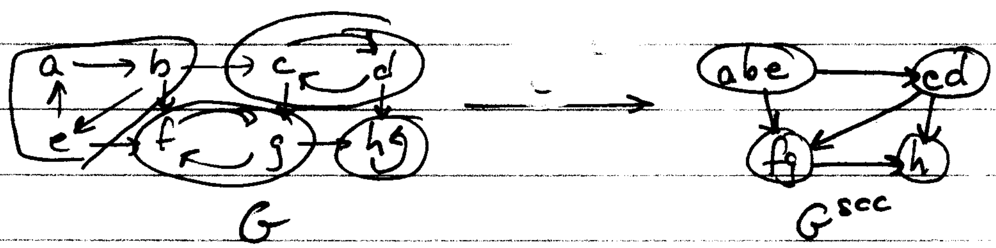
Analysis

1. Call DFS(G) $\Longrightarrow f[u]$ — $\Theta(V+E)$
2. Compute G^T — $\Theta(V+E)$
3. Call DFS(G^T), but in main loop consider vertices in decreasing $f[u]$ order. — $\Theta(V+E)$
4. Output vertices of each tree in depth-first forest as s.c.c. — $\Theta(V)$ ← Can be embedded in step 3

Total $\Theta(V+E)$

<u>Key Idea</u> ← Why this works

Consider G^{SCC}: Strongly connected components of G represented as single vertex; edges in G^{scc} correspond to ~~one~~ edges ~~in~~ in G.

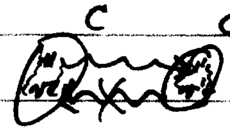


G $\qquad\qquad\qquad\qquad$ G^{scc}

<u>Lemma</u>: C, C' distinct s.c.c.'s with $u, v \in C$; $u', v' \in C'$, and path $u \rightsquigarrow u'$ exists in G. Then $v' \rightsquigarrow v$ does not exist in G.

<u>Proof:</u> $v' \rightsquigarrow v$ existing $\Rightarrow$ $\underbrace{u \rightsquigarrow u'}_{\text{given}} \rightsquigarrow \underbrace{v'}_{\text{same s.c.c.}}$ & $\underbrace{v' \rightsquigarrow v}_{\text{assumed}} \rightsquigarrow u$.

Thus, u & v reachable from each other. Contradiction.



strongly connected components

<u>Notation</u>: $d(U) = \min_{u \in U} \{d[u]\}$ ← earliest of set of discovery times

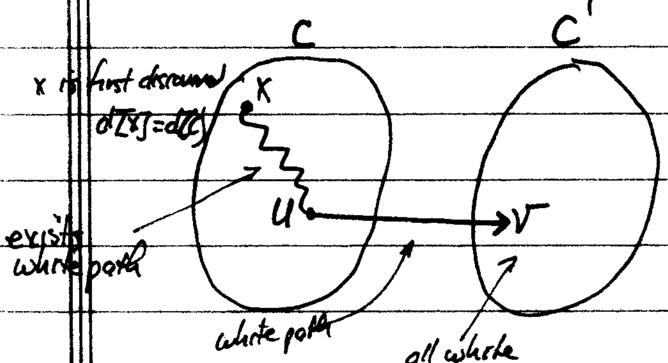
$f(V) = \max_{v \in V} \{f[v]\}$ ← latest of set of finishing times

<u>Lemma</u>: Let C & C' be distinct S.C.C's of digraph $G=(V,E)$. Let edge $(u,v) \in E$ with $u \in C$ and $v \in C'$. Then $f(C) > f(C')$. E^T

$f(C) < f(C')$ in G

<u>Sketch of Proof</u>:

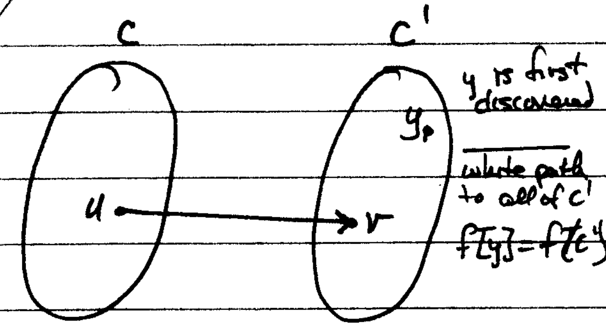
Case 1: $d(C) < d(C')$



By White-path theorem, all vertices in C' are descendants of x and ~~thus have earlier~~ finishing times

$f(C) = f[x] > f(C')$

Case 2: $d(C) > d(C')$



No vertex in C reachable from C' (previous lemma), so at $f[y]$, C is all white. Thus $f(C) > f(C')$

So, whether we begin C or C' first, C' will always finish first!!

Corollary

So, here's how STRONGLY-CONNECTED-COMPONENTS works:

→ DFS of G^T starts with $f(C')$ with maximum finishing time and explores this SCC completely.

→ By previous corollary, THERE ARE NO EDGES LEADING OUT OF THIS SCC in G^T

→ When we finish, we start new tree with an element of next S.C.C. (C''), whose only edges out (if any) return to C', which has already been finished.