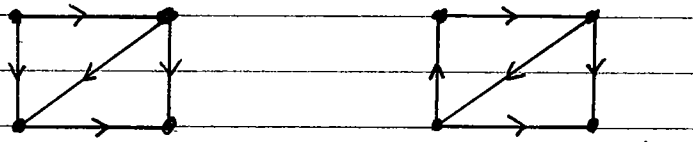
23.4 Topological Sort

A directed graph is called Acyclic if it contains no directed cycles

Ex.



Acyclic Not Acyclic

Lemma

A directed graph G is acyclic iff DFS(G) yields no back edges.

Proof

Equivalently we show G contains a back edge iff it contains a directed cycle.

($\Rightarrow$) If G contains a back edge it obviously contains a directed cycle.

($\Leftarrow$) Suppose G contains a directed cycle C. Let v be the first vertex on C to be discovered and let (u,v) be the edge preceding v along C. Then at time $d[v]$ the vertices of C form a path of white vertices from v to u. By the white path theorem, u becomes a descendent of v. Therefore (u,v) is a back edge.

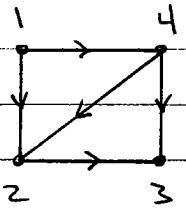
///

LET $G = (V, E)$ BE A DIRECTED ACYCLIC GRAPH (DAG).

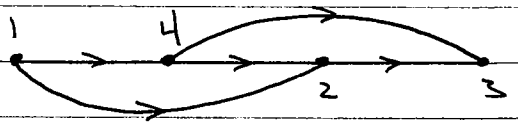
DEFN

A TOPOLOGICAL SORT OF V IS A LINEAR ORDERING OF THE VERTICES SUCH THAT $(u,v) \in E$ IF AND ONLY IF u APPEARS BEFORE v IN THE ORDERING.

EX.



G



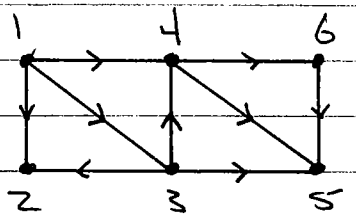
A TOPOLOGICAL SORT OF $V(G)$.

A TOPOLOGICAL SORT OF A DAG CAN BE VIEWED AS AN ORDERING OF V ALONG A HORIZONTAL LINE SUCH THAT ALL DIRECTED EDGES GO FROM LEFT TO RIGHT.

TO PERFORM A TOPOLOGICAL SORT ON A DAG CALL DFS(G), AND AS EACH VERTEX IS FINISHED INSERT IT ONTO A STACK. WHEN DFS IS COMPLETE THE STACK CONTAINS THE VERTICES IN TOPOLOGICALLY SORTED ORDER.

EQUIVALENTLY ONE CAN SIMPLY RUN DFS THEN SORT THE VERTICES IN DESCENDING ORDER BY FINISH TIMES $f[u]$.

Ex

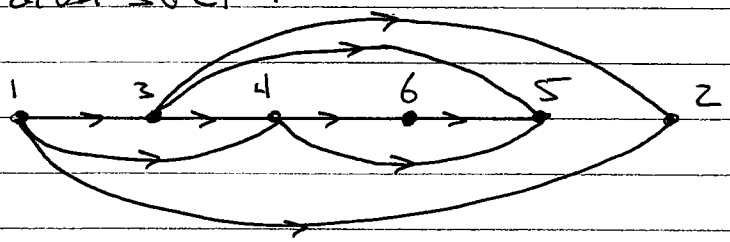


u	$d[u]$	$f[u]$
1	1	12
2	2	3
3	4	11
4	5	10
5	6	7
6	8	9

FINISH TIMES $f[u]$: 12 11 10 9 7 3

VERTICES u: 1 3 4 6 5 2

TOPOLOGICAL SORT:



THEOREM 22.12 ON P. 551 ASSERTS THAT THIS PROCEDURE ALWAYS YIELDS A TOPOLOGICAL SORT OF A DAG.

23.5 Strongly Connected Components

Let $G=(V,E)$ be a directed graph. A subset $U \subseteq V$ is said to be *strongly connected* iff every vertex in U is reachable from every other vertex in U. i.e. for all $x, y \in U$, G contains both a directed x-y path and a directed y-x path.

We call $U \subseteq V$ a *strongly connected component* of G iff it is

(1) strongly connected, and
(2) U is maximal with respect to (1), i.e. whenever $U \subsetneq W \subseteq V$ then W is not strongly connected.

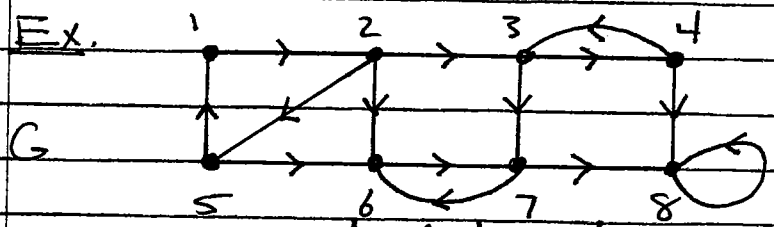
DFS can be used to find the strong components of G as follows:

- Call DFS(G). As vertices are finished, place them on a stack.
- Compute the transpose graph G^T, i.e. reverse all edge directions in G.
- Call DFS(G^T) processing vertices in main loop of DFS by decreasing finish times from first call, i.e. just pop vertices off the stack.

WHEN THIS SECOND CALL TO DFS IS COMPLETE, THE VERTICES IN EACH TREE OF THE RESULTING DFS FOREST CONSTITUTE THE STRONG COMPONENTS OF G.

SEE THEOREM 22.16 (P. 556) AND THE PRECEDING LEMMAS FOR A PROOF THAT THIS PROCEDURE CORRECTLY DETERMINES THE STRONG COMPONENTS OF ANY DIRECTED GRAPH.

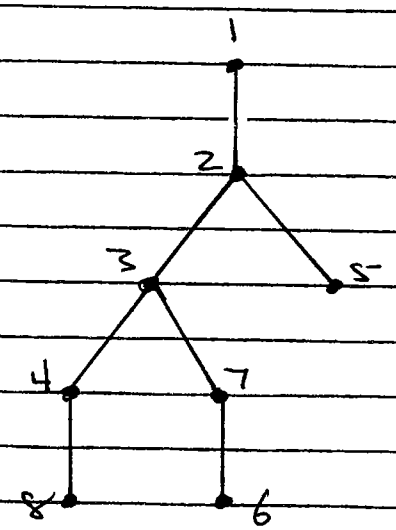
Ex.

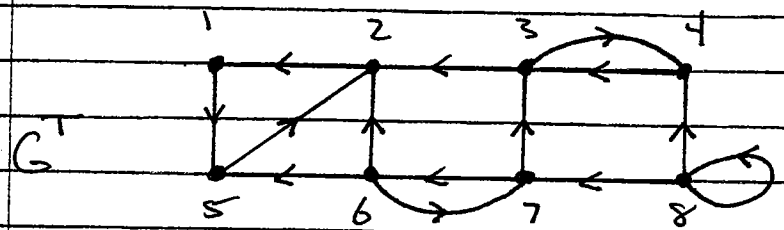


G

	d	f
1 : 2	1	16
2 : 3 5 6	2	15
3 : 4 7	3	12
4 : 3 8	4	7
5 : 1 6	13	14
6 : 7	9	10
7 : 6 8	8	11
8 : 8	5	6

DFS FOREST:

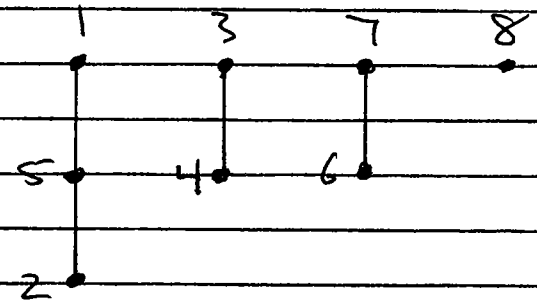




G^T

	d	f
1 : 5	1	6
2 : 1	3	4
5 : 2	2	5
3 : 2 4	7	10
7 : 3 6	11	14
6 : 2 5 7	12	13
4 : 3	8	9
8 : 4 7 8	15	16

DFS FOREST:



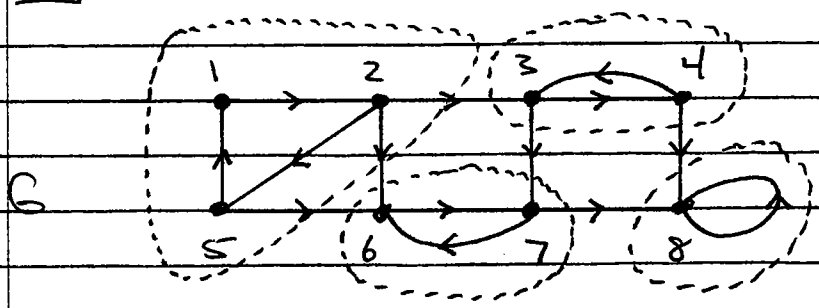
STRONG COMPONENTS OF G:

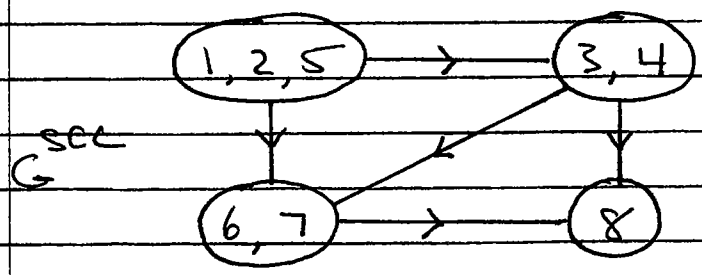
$C_1 = \{1, 2, 5\}$
$C_2 = \{3, 4\}$
$C_3 = \{6, 7\}$
$C_4 = \{8\}$

NOTICE THAT THE STRONG COMPONENTS OF G AND G^T ARE ONE AND THE SAME.

THE COMPONENT GRAPH OF G (ALSO CALLED CONDENSATION GRAPH), DENOTED G^{SCC}, IS THE DIRECTED GRAPH WHOSE VERTICES ARE THE STRONG COMPONENTS OF G, AND WHICH HAS A DIRECTED EDGE FROM C_i TO C_j IFF THERE EXIST $x \in C_i$ AND $y \in C_j$ SUCH THAT $(x, y) \in E(G)$.

Ex. Following the preceding example



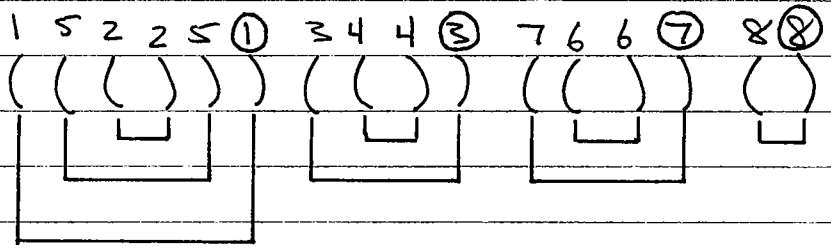


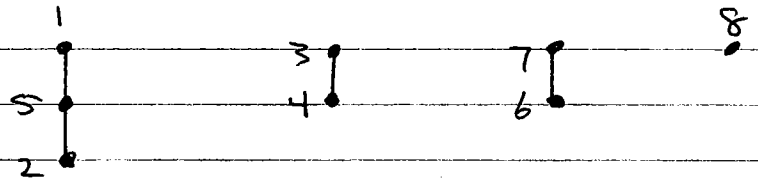
Observe that G^{SCC} is necessarily acyclic.

There is a simple way to extract the strong components of G from the state of the stack after the second call to DFS(G^T). We assume here that in this second call also, vertices are placed on a stack as they are finished.

Vertices will be grouped in this stack into strong components, delimited by those vertices with Nil parents.

EX. RETURNING TO THE PREVIOUS EXAMPLE





PARENT: n n 7 n 3 n 1 5

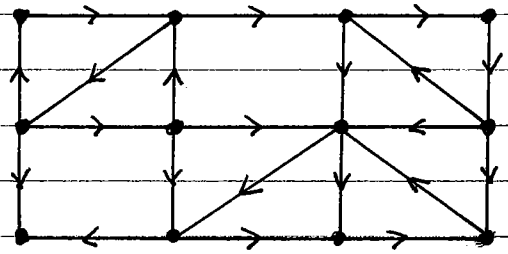
STACK: ⑧ ⑦ 6 ③ 4 ① 5 2

STRONG COMPONENTS: {8}, {7, 6}, {3, 4}, {1, 5, 2}.

EXERCISE

RUN THIS PROCEDURE ON THE GRAPH BELOW:

a.)



b.)