Depth-First- Search(DFS)

- To search ‘deeper’ in the graph whereas possible.
  - When all of u’s have been explored. The search backtracks to the edge leaving the vertex from which v’s was discovered.
  - If any undiscovered vertex remains, then one of them is selected as new source and the search is repeated from that source.
  - Each vertex is initially white(w)
    - Grey(G) when it is discovered.
    - Black(B) when it is finished.
  - Each vertex has two timestamp
    - pre[v] = when it is discovered (grey)
    - post[v] = when it is finished (black)
  - The input graph may be undirected or directed
  - The variable ‘time’ is global variable that we use for time stamping.

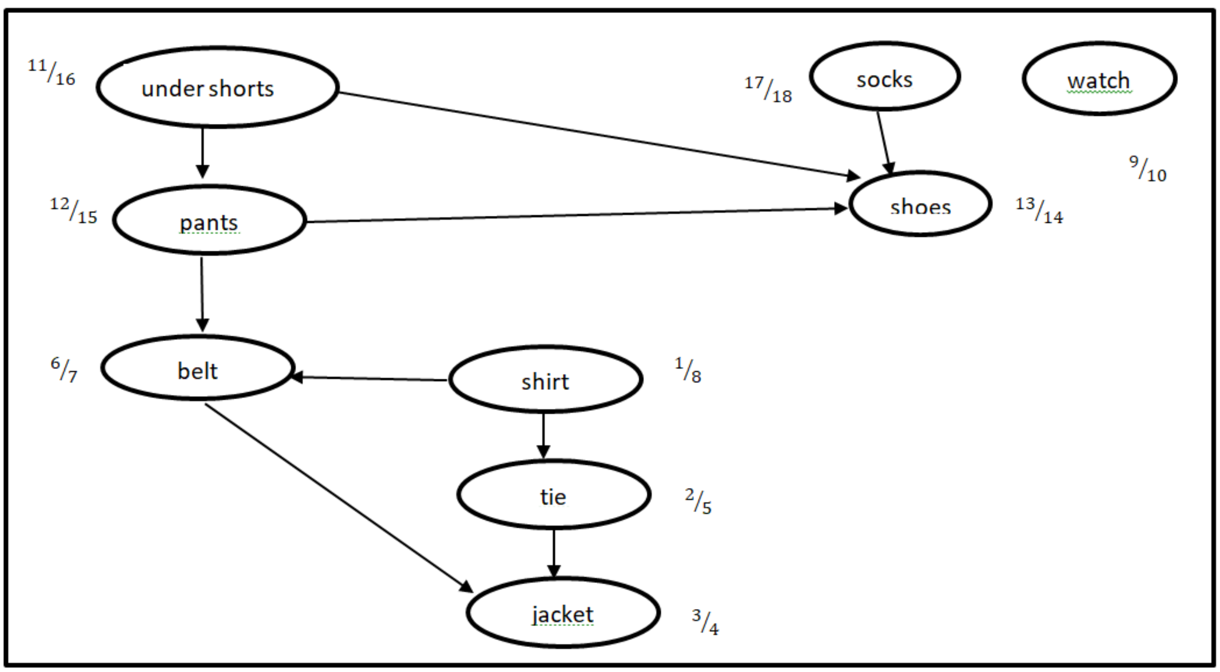
```
DFS(G) {

    for each vertex uϵV[G] do{
        color[u] = W; p[u] = NIL; // p[u] is parent of u
    }
    time = 0;
    for each vertex uϵV[G] do {
        if color[u] = W then DFS_VISIT(u);
    }
}
DFS_VISIT(u)  {
    color[u] = G;
    time = time+1;
    pre[u] = time;
    for each  v ϵ Adj[u] do {
        if color[v] = W then{
            p [v] = u;
            DFS_VISIT(v);
        }
    }
    color[u] = B;
    time = time+1;
    post[u] = time;
}
```

Topological sort(TS):

- DFS can be used to perform a topological sort of a directed acyclic grahp(DAG)
- A directed grahp having no cycles is called a DAG
- A topological sort of a DAG is a linear ordering of all its vertices such that if G contains an edge(u,v) then it appears before v in the ordering
- TS of a graph can be viewed as an ordering of its vertices along a horizontal line so that all directed edges go from left to right.

Problem:

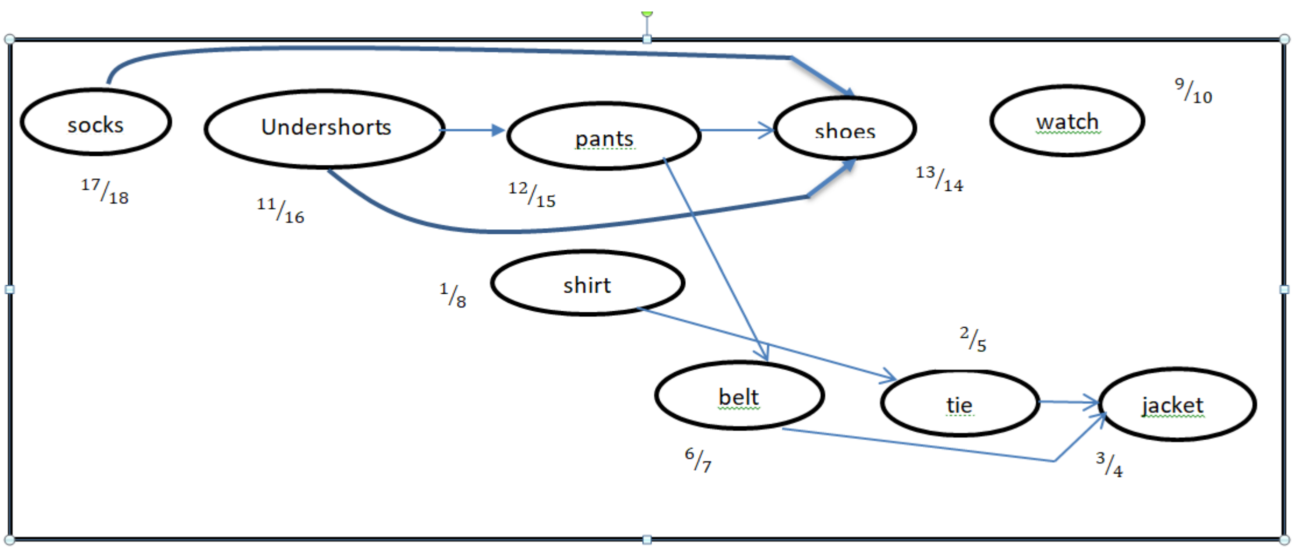
- Prof Bose topologically sorts his clothing when getting dressed
- Each directed edge(u,v) means that garment u must be put on before garment v.
- Graph:



Execution of DFS:

shirts	Tie	Jacket	belt	pants	undershorts	shoes	socks	watch	time	Pre	Post	parents(p)
G	W	W	W	W	W	W	W	W	1	pre(shirts) = 1		p(shirts) = NIL
G	G	W	W	W	W	W	W	W	2	pre(tie) = 2		p(tie) = shirts
G	G	G	W	W	W	W	W	W	3	pre(jacket) = 3		p(jacket) = tie
G	G	B	W	W	W	W	W	W	4		post(jacket) = 4	
G	B	B	W	W	W	W	W	W	5		post(tie) = 5	
G	B	B	G	W	W	W	W	W	6	pre(belt) = 6		p(belt) = shirt
G	B	B	B	W	W	W	W	W	7		post(belt) = 7	
B	B	B	B	W	W	W	W	W	8		post(shirt) = 8	
B	B	B	B	W	W	W	W	G	9	pre(watch) = 9		p(watch) = NIL
B	B	B	B	W	W	W	W	B	10		post(watch) = 10	
B	B	B	B	W	G	W	W	B	11	pre(undershorts) = 11		p(undershorts) = NIL
B	B	B	B	G	G	W	W	B	12	pre(pants) = 12		p(pants) = undershorts
B	B	B	B	G	G	G	W	B	13	pre(shoe) = 13		p(shoe) = pants
B	B	B	B	G	G	B	W	B	14		post(shoe) = 14	
B	B	B	B	B	G	B	W	B	15		post(pants) = 15	
B	B	B	B	B	B	B	W	B	16		post(undershorts) = 16	
B	B	B	B	B	B	B	G	B	17	pre(shocks) = 17		p(shocks) = NIL
B	B	B	B	B	B	B	B	B	18		post(shocks) = 18	

- After topological sort of graph G



- Topological_sort (G){

  Call DFS(G) to compute finishing time post(v) for each vertex v as each vertex is finished;

  Insert it into the front of linked list;
  Return the linked list of vertices;
  }

- Complexity
  - The DFS takes $O(|V| + |E|)$
  - O(1) is taken to insert each of the |V| vertices onto the front of the linked list
  - Overall $O(|V| + |E|)$

Strongly connected components:

- A directed graph is strongly connected if every two vertices are reachable from each other.
- A strongly connected component of directed graph $G = (V, E)$ is a maximal set of vertices $C \subseteq V$ such that for every pair of vertices u and v in C , we have both $u \rightarrow v$ and $v \rightarrow u$; i.e., vertices u and v are reachable from each other.

S-C-C(G){

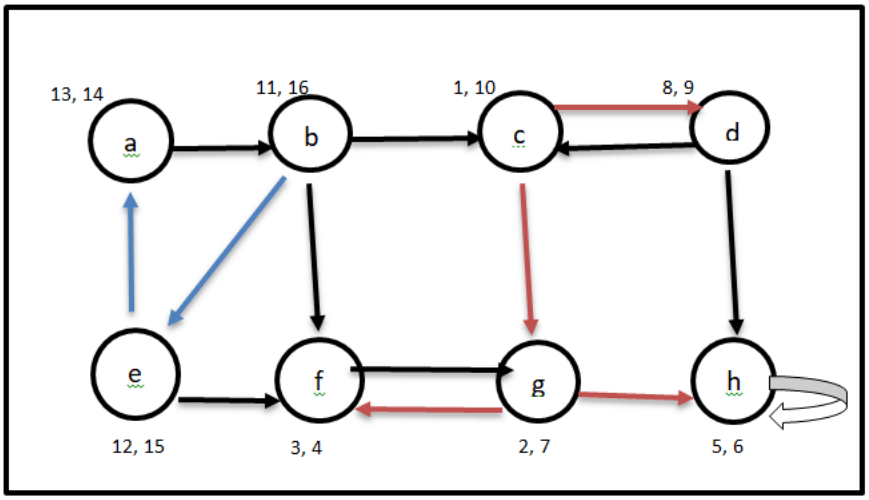
1. Call DFS(G) to compute the finishing times post[u] for each vertex u.
2. Compute G^T
3. Call DFS(G^T); but in the main loop of DFS, consider the vertices in order of decreasing post[u](as computed in line 1)
4. Output the vertices of each tree in the DFS forest formed in line 3 as a separate strongly connected components.

}

Suppose that G has strongly connected components: $C_1, C_2, C_3, \dots \dots . C_K$. The vertex set is

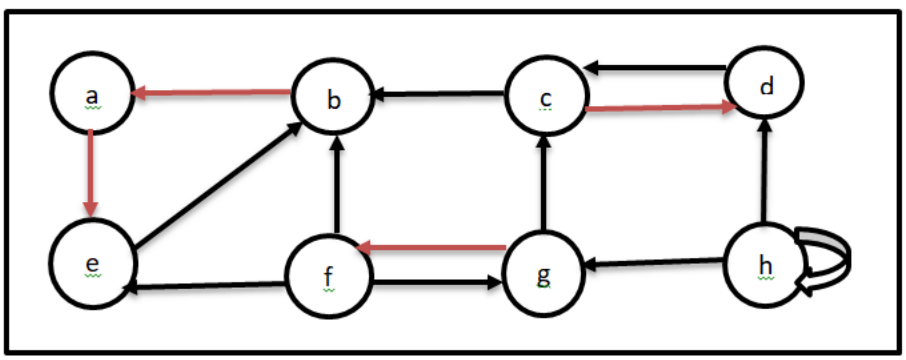
$V_{scc} = \{v_1, v_2, \dots v_k\}$ and it contains a vertex v_i for each strongly connected components.

Graph G



- pre(a) (discovery time) = 13, post(a) (finishing time) = 14,

Graph G^T

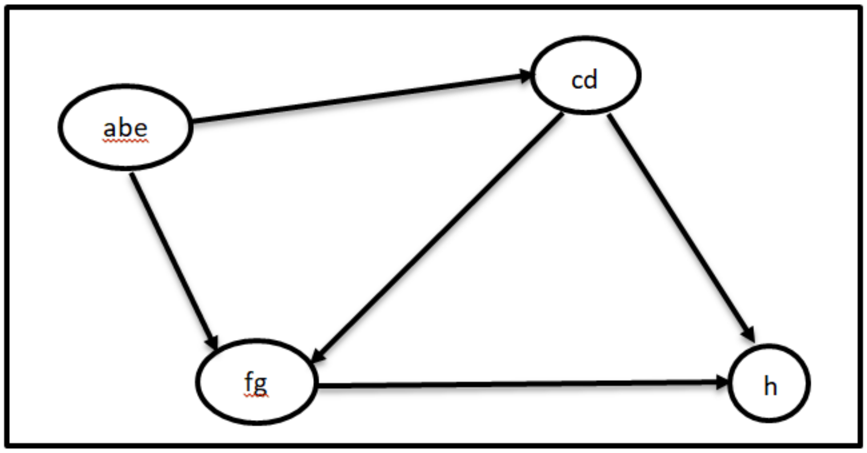


Strongly connected components:

→ Vertices b, c, g, h are the roots of the DFS trees produces by the execution of DFS on G^T.

→ Directed Acyclic Graph (DAG) containing the strongly connected components {a, b, e}, {c, d}, {f, g}, {h}

→ G^{SCC}



→ G^{SCC} obtained by contracting all edges within each strongly connected component of G so that only a single vertex remains in each component.