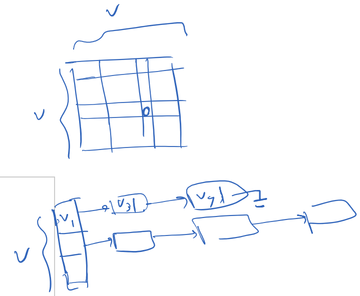
Graph Basics

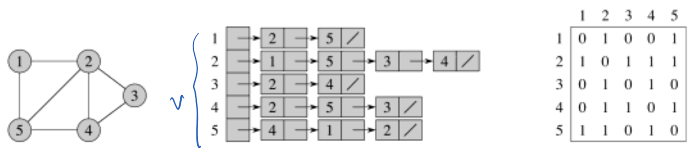
$G = \langle V, E \rangle$ → edges; V → vertices

$G = (V, E)$

$E \subseteq V \times V$

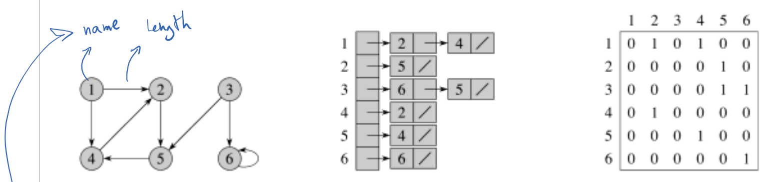
representing in memory { adjacency matrix / adjacency list





(a) (b) (c)

	adj lists	adj matrix
Space Complexity:	$\Theta(\lvert V\rvert + \lvert E\rvert)$	$\Theta(\lvert V\rvert^2)$
time complexity of listing adj vs to u :	$\Theta(\text{degree}(u))$	$\Theta(V)$
time complexity of checking $(u,v) \in E$:	$O(\text{degree}(u))$	$\Theta(1)$



(a) (b) (c)

attributes for vertices or edges

showing attribute d notationally: $v.d$, $(u,v).d$

Breadth-First Search (BFS)

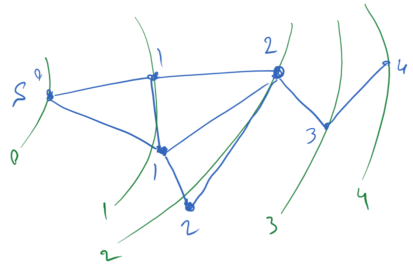
input: $G = \langle V, E \rangle$, source vertex $s \in V$

Outputs: $v.d$ distance from s to v

→ (smallest number of edges that connect them)

BFS(V, E, s)

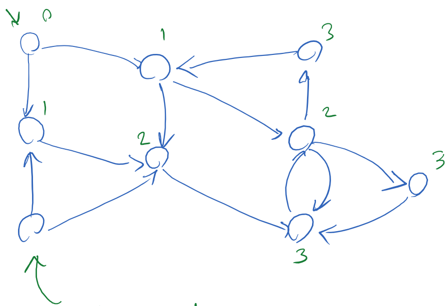
└→ (smallest number of edges that connect them)



```
BFS(V, E, s)
  for each u ∈ V − {s}
      u.d = ∞
  s.d = 0
  Q = ∅
  ENQUEUE(Q, s)
  while Q ≠ ∅
      u = DEQUEUE(Q)
      for each v ∈ G.Adj[u]
          if v.d == ∞
              v.d = u.d + 1
              ENQUEUE(Q, v)
```

Queue

Time complexity $= O(V+E)$



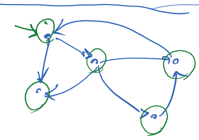
not visited

Q: distance of enqued vertices

$i \quad i \quad \ldots i \quad i+1 \quad i+1 \ldots i+1$

monotonically increasing

Depth-First Search (DFS)



timestamps $\begin{cases} \text{discovery time } (v.d) \\ \text{finishing time } (v.f) \end{cases}$

No source node. We will discover all vertices

coloring vertices:

- white : undiscovered vertex
- gray : discovered "
- black : finished "

times range: $1 \ldots 2|V|$

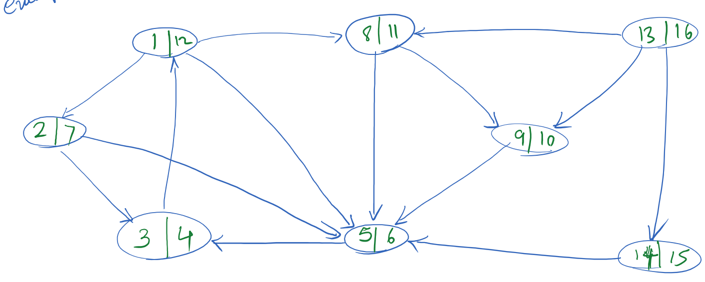
$$1 \leq v.d < v.f \leq 2|V|$$

```
DFS(G)
  for each u ∈ G.V
      u.color = WHITE
  time = 0
  for each u ∈ G.V
      if u.color == WHITE
          DFS-VISIT(G, u)
```

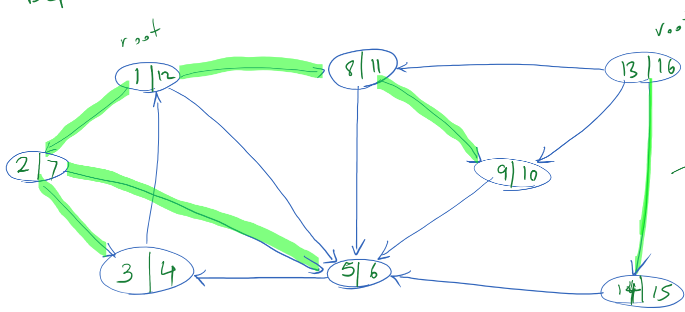
```
DFS-VISIT(G, u)
  time = time + 1
  u.d = time
  u.color = GRAY                  // discover u
  for each v ∈ G.Adj[u]           // explore (u, v)
      if v.color == WHITE
          DFS-VISIT(v)
  u.color = BLACK
  time = time + 1
  u.f = time                      // finish u
```

Time: $\Theta(V+E)$

example DFS:



Depth-First Forest (containing Depth-First Trees)



when explored

Parenthesis Theorem

for any pair u, v of vertices, one of the following holds:

- neither u or v is a descendant of the other

  $u.d < u.f < v.d < v.f$ or $v.d < v.f < u.d < u.f$

- u is descendant of v

  $v.d < u.d < u.f < v.f$

- v is descendant of u

$v.d < u.d < u.f < v.f$

- v is descendant of u

$u.d < v.d < v.f < u.f$

✓ ( ) [ ]

✓ ([ ])

× ( ]) [× $u.d < v.d < u.f < v.f$

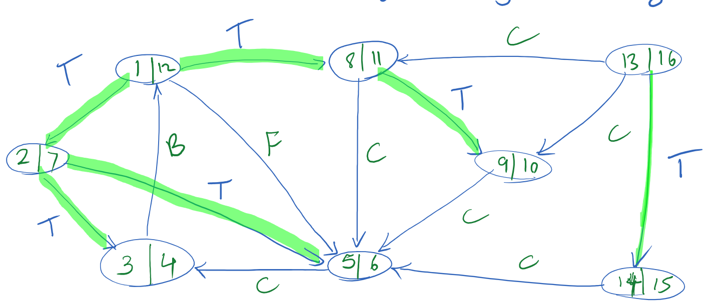
× ([ )]

white-path Theorem

v is descendant of u iff at time u.d, there is path $u \rightsquigarrow v$ consisting of only white vertices.

labling edges: - Tree edge

- Back edge (u,v) where u is descendant of v
- Forward edge (u,v) where v is descendant of u but not a tree edge
- Cross edge : any other edge



Theorem:

In DFS of undirected graphs, we have only T and B edges. (No F or C edges)

Topological Sort

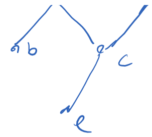
partial orders : a, b , $a \nless b$ nor $a \ngtr b$

transitivity, $a < b$, $b < c \implies a < c$

$a > c$

$d > c$

total order



partial order

$a > c$

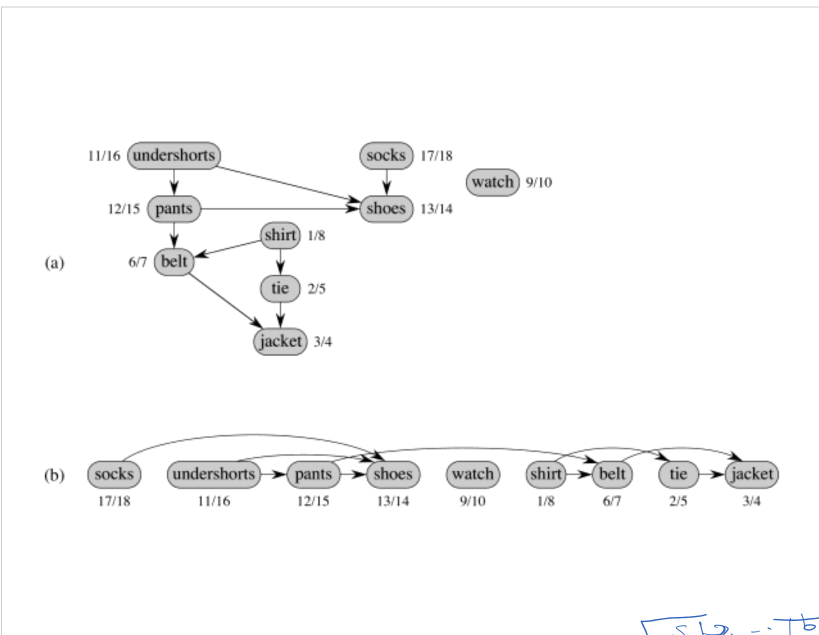
$d > c$

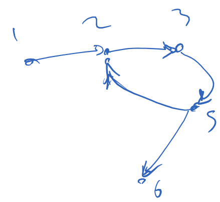
$a \not< b \quad a \not> b$

↳ topological sort: $a < d < b < c < e$

$d < a < b < c < e$

d, b, e



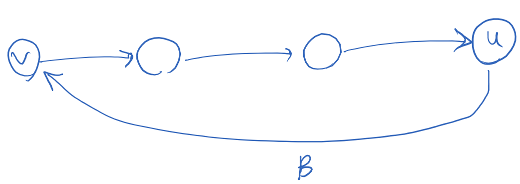


1 2 3 5 6

a graph with cycle has no topological sort corresponding

Lemma: directed graph is acyclic iff a DFS of graph yield no back edges.

① if we have back edge ⟹ cycle



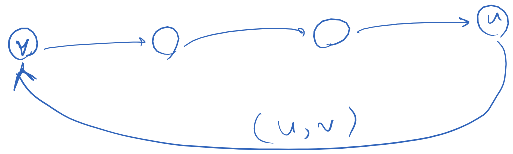
$v \to u$

$u \to v$ ⟹ cycle

$v \to u \to v$

② cycle ⟹ back edge

- v is first vertex visited in the cycle



- v is first vertex visited in the cycle
- at time v.d, there will be white-path

$v \rightsquigarrow u$

$\Downarrow$ white-path theorem

u is descendant of v

$\Downarrow$ & (u,v)

(u,v) is a back-edge

directed acyclic graph

Topological sort of a dag: linear ordering of vertices s.t. $(u,v) \in E$ then u appears before v.

Topological_Sort(G)

call DFS(G) &
output vertices in order of decreasing finishing times $\Big\}$ $\Theta(V+E)$

Correctness: if $(u,v) \in E$ then $u.f > v.f$ ⟵ show

As we explore (u,v), u is gray and v is:

- is v gray? $\Longrightarrow$ v is an ancestor of u
  $\Longrightarrow$ (u,v) is a back edge
  $\Longrightarrow$ contradiction with being dag

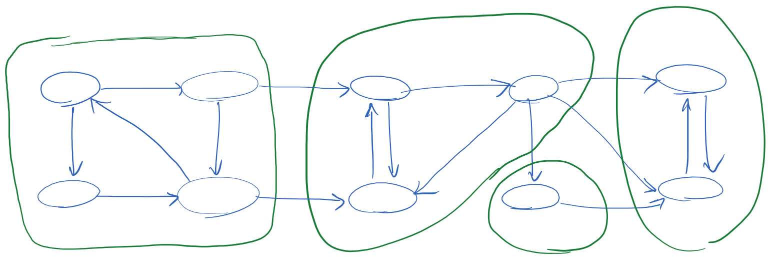
- is v white? $\Longrightarrow$ v becomes descendant of u
  parenthesis th. $\Longrightarrow$ $u.d < v.d < \boxed{v.f < u.f}$

- is v black? $\Longrightarrow$ v is already finished $\Longrightarrow$ $\boxed{v.f < u.f}$

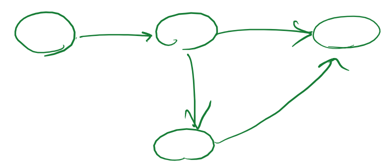
- is v black? $\Longrightarrow$ v is already finished $\Longrightarrow$ $\boxed{v.f < u.f}$

Strongly Connected Components:

maximal set of vertices $C \subseteq V$ s.t. for all $u, v \in C$, both $u \rightsquigarrow v$ & $v \rightsquigarrow u$



Component Graph



G^T = transpose of G

$(u,v) \in E \Longleftrightarrow (v,u) \in E^T$

SCC(G)

call DFS(G) to compute finishing times $u.f$ for all u
compute G^T
call DFS(G^T), but in the main loop, consider vertices in order of decreasing $u.f$
  (as computed in first DFS)
output the vertices in each tree of the depth-first forest formed in second DFS
  as a separate SCC