Graphs: 2 Basic Alg'ms

Breadth-First Search (BFS)
Depth-First Search (DFS)

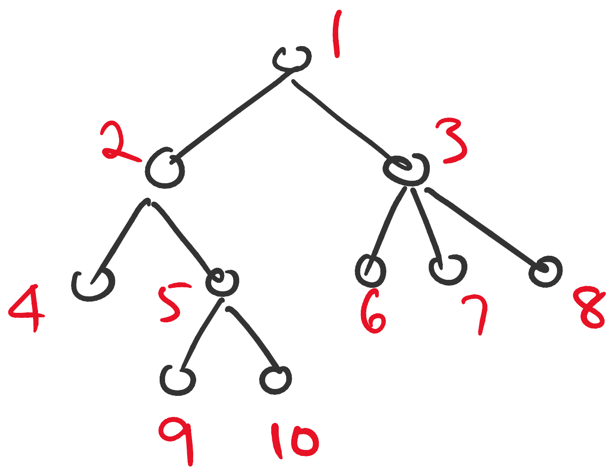
Basic Problems

- find all vertices reachable from s
- find all connected components

⋮

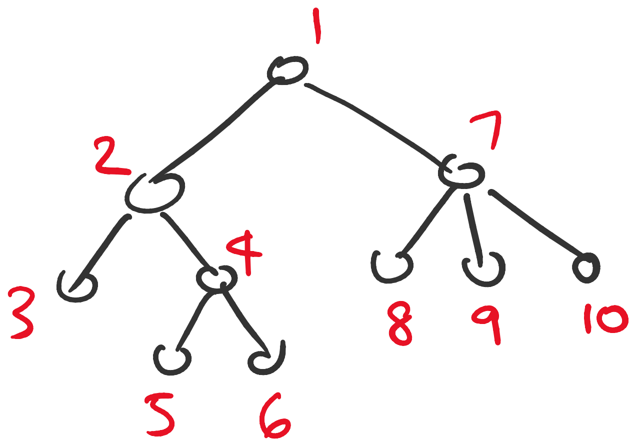
Ex trees

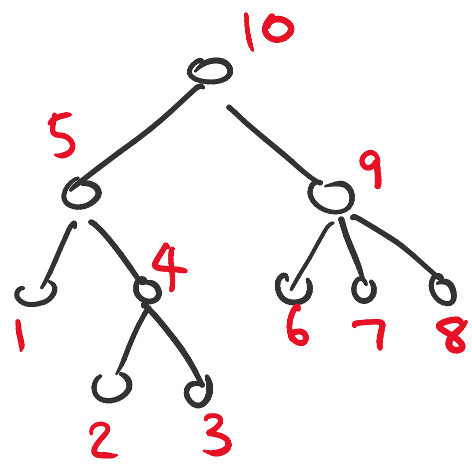
BFS



DFS

discovery order
= pre-order



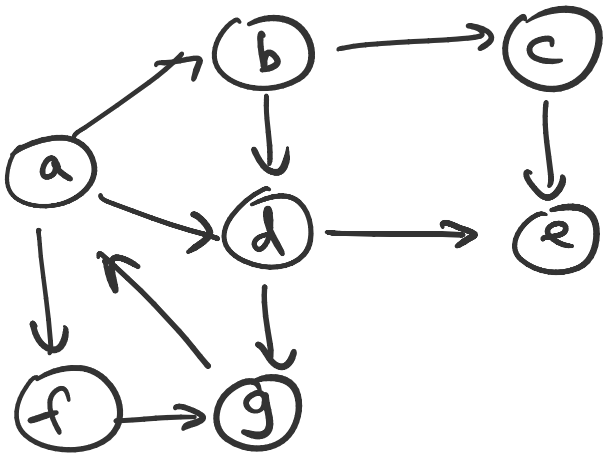


finish order
= post-order

Extension to graphs

main diff: don't re-visit a vertex
if we have seen it before!

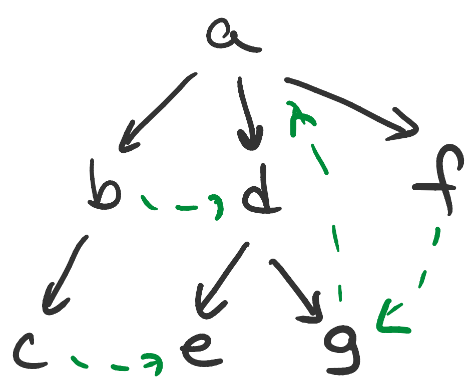
Ex



DFS tree

back edge: (g,a)
forward edge: (a,d)
Cross edge: (d,e), (f,g)

BFS tree



back edge: (g,a)
cross edge: (b,d), (c,e), (f,g)

no forward edges in BFS

for undir graphs, no <u>cross</u> edges in DFS

Implementation:

BFS(G,s) :

```
  // idea 1. mark vertices when visited...
  // idea 2. use a queue data structure Q ...

for each v ∈ V do unmark v
insert s to Q, mark s
while Q ≠ ∅ {
      remove a vertex u from Q          (head)
      for each out-neighbor v of u do {   v ∈ Adj[u]
            if v is unmarked
                insert v to Q, mark v,
```

6.

7. insert v to Q, mark v,
↳ at tail parent[v] = u, level[v] = level(u) + 1

}

Runtime: lines 5-7 $O(\text{out-deg}(u))$ time

total time $O\Big(n + \underbrace{\sum_{u \in V} \text{out-deg}(u)}_{\text{each vertex } u \text{ considered once}}\Big)$

$= \boxed{O(n+m)}$

DFS(G, s):
// similar, but use a diff. data structure: stack, or recursion

1. mark s, discovered[s] = time ++ ← pre-numbering
2. for each out-neighbor v of s
3. if v is unmarked
4. DFS(G, v), parent(v) = s
5. finished[s] = time ++ ← post-numbering

DFS-All(G):
0. for each $v \in V$ do unmark v
1. for each $v \in V$ do
2. if v is unmarked, DFS(G, v)

Runtime: $\boxed{O(n+m)}$ again

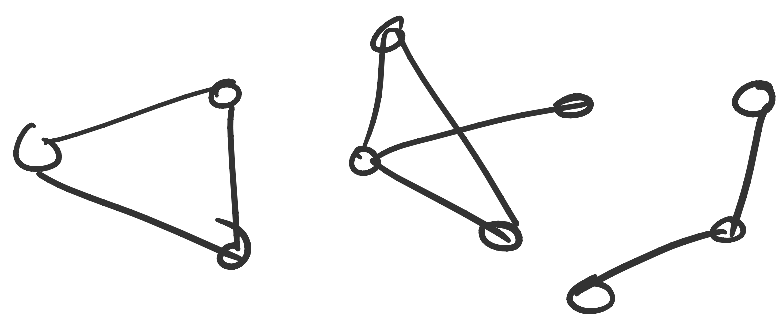
Applications

Applications

Ex 1 Given dir/undir graph $G=(V,E)$, $s,t \in V$,
find unweighted shortest path from s to t

1. run BFS(G,s)
2. return path in BFS tree from s to t

Ex 2 Given undir graph G,
find all connected components



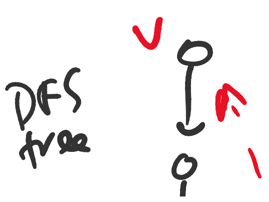
1. run BFS-All(G) or DFS-All(G)
2. return BFS/DFS trees as components

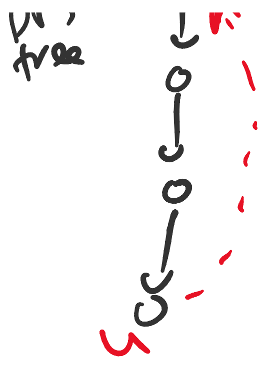
Ex 3 Given undir graph G,
decide whether G has a cycle

1. run BFS-All(G) or DFS-All(G)
2. check for non-tree edges

Ex 4 Given dir graph G,
decide whether G has a cycle

1. run DFS-All(G)
2. check for back edges $(u,v) \in E$
   (when [discovered(u), finished(u)]





(when $[\text{discovered}(u), \text{finished}(u)] \subset [\text{discovered}(v), \text{finished}(v)]$)

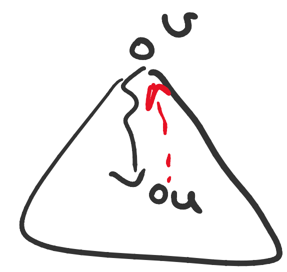
Correctness:

$\exists$ back edge $\Rightarrow$ $\exists$ cycle

$\exists$ cycle $\Rightarrow$ $\exists$ back edge:

Hint: let v be first vertex discovered in cycle.

let u be vertex before v in cycle



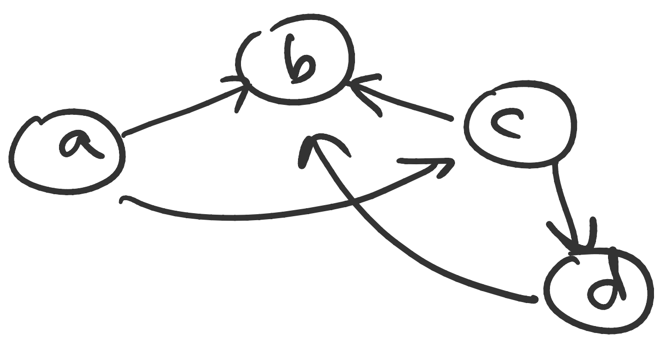
Ex 5 Topological Sort

Given dir. graph $G=(V,E)$,

find a vertex ordering s.t.

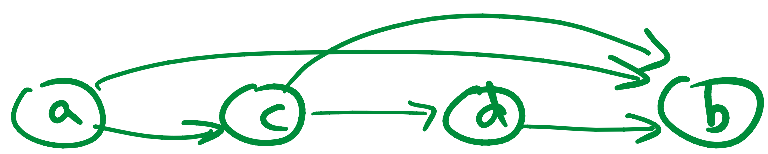
$\forall (u,v) \in E \Rightarrow u$ appears before v

e.g.

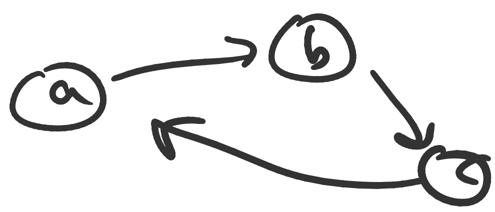


answer:

a, c, d, b



e.g.



no answer.

(because $\exists$ cycle)