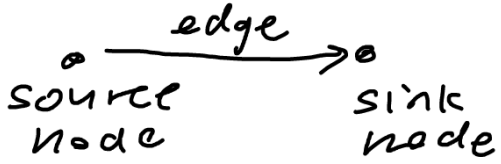
# Graph Traversals

Exam: Tuesday

Last time: Graph is set of vertices (nodes) and set of connecting edges.

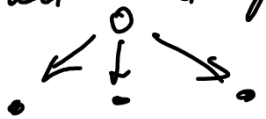
Directed graph




How to search graphs?

- find reachable nodes
- find "best" node, "best" path
- find cycles
- toposort (edges = dependencies)

Reachability: what nodes reached via path from root(s).



# Tricolor algorithm (Dijkstra)

Each node has 1 of three colors at time $t$

- white: undiscovered (no path yet from root)
- gray: discovered (frontier)
- black: discovered & finished


## Tricolor invariant

No edge from black to white

⇒ can only discover new nodes from gray

## Pseudocode (underspecified)


1. color all nodes white
2. color root(s) gray
3. while (some gray node g) {
   - color some white successors of g gray (add to frontier)
   - if no more white successors, may color g black

   }

Underspecified: Can refine by making choices:

1. pick any g
2. pick which white successors to color gray
3. pick whether to color g black

Regardless of choices, can run this in parallel.

# Refinement 1: breadth-first search (BFS)


Gray frontier = FIFO queue. (frontier)

first in, first out

v.distance = length of shortest path to v (or $\infty$ if v is white)

v is black if: v.distance $\neq \infty$ & v $\notin$ queue

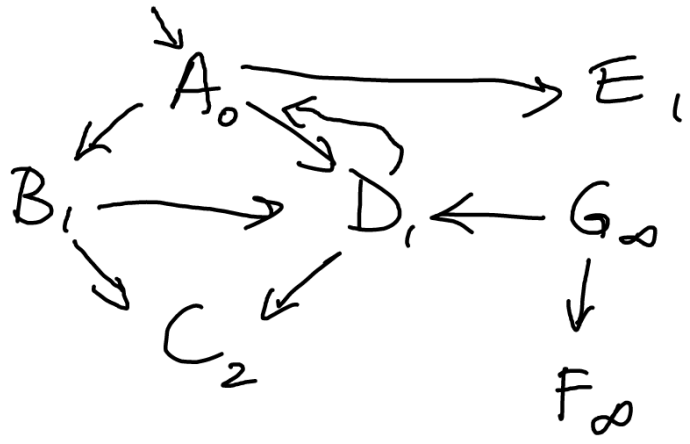
1. set all distances to $\infty$ (white)
2. set root distance to 0, push root onto queue.
3. while (!frontier.empty()) {
   Vertex g = frontier.pop();
   → for each (g → v) {
   if (v.distance == $\infty$) {
   frontier.push(v);
   v.distance = g.distance + 1;
   } }

g is black → }

outer loop O(V)

inner loop O(E)

total: O(V+E)




Frontier Queue + distance

| A : 0 |
|---|
| B : 1 |
| D : 1 |
| E : 1 |
| C : 2 |

↑ not in queue

G : ∞

F : ∞ (unreachable)

BFS invariant: queue only contains distances $d$, $d+1$ (for some $d$, sorted by distance)

⇒ nodes popped in distance order

⇒ learn final distance when popped. (& pushed)

(Don't have to keep track of distances)

BFS

Time: $O(V+E)$

Space: $O(V)$ queue + distance

# Refinement #2: Depth-First Search (DFS)

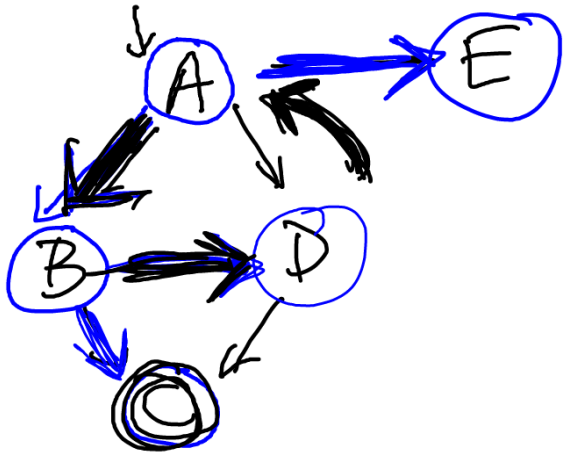
- use LIFO stack
- or: recursion: stack = call stack

Effect: frontier is a simple path from root to current node


Frontier of gray nodes = call stack's g

```
/** Effect: mark as black all
    nodes reachable from g on
    all-white paths.
    Requires: g is gray. */
void visit(Vertex g) {
    foreach (g → v) {
        if (v.color == white) {
            v.color = gray;
            visit(v);
        }
    }

    v.color = black;  // no white successors
```




visit(A)

```
void visit(Vertex g) {
    foreach (g → v) {
        if (v.color == white) {
            v.color = gray;
            visit(v);
        }
    }
    v.color = black;  // no white successors
```

Run time?

O(V) calls

O(E) time in innerloop

O(V+E)

Space

color: O(V)

+stack: O(V)

O(V)

Finishing order:

C D B E A

All reachable nodes are black

at end: no gray nodes

no black→white edges

gray → gray edge : cycle in graph.

gray loop.

DFS ⇒ linear-time cycle detection.