CS3000
6/4-weds

Admin

- HW4 due tmw 9pm
- APP7 due 11:30AM tmw
- Exam 2 6/12

Agenda

1. Topological sort
2. Depth-First Search
3. Edge classification
4. APP7

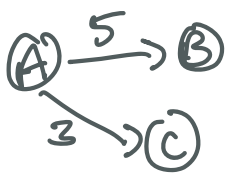
0. BFS Review

BFS
- unweighted
- undirected

? would we need to change the algo:
- weighted → lines 5
- directed → no changes

what is run-time?
$\Theta(V+E)$

```
BFS(G, s)
 1  for each vertex u ∈ G.V − {s}
 2       u.color = WHITE
 3       u.d = ∞
 4       u.π = NIL
 5  s.color = GRAY
 6  s.d = 0
 7  s.π = NIL
 8  Q = ∅
 9  ENQUEUE(Q, s)
10  while Q ≠ ∅
11       u = DEQUEUE(Q)
12       for each v ∈ G.Adj[u]
13            if v.color == WHITE
14                 v.color = GRAY
15                 v.d = u.d + 1
16                 v.π = u
17                 ENQUEUE(Q, v)
18       u.color = BLACK
```



Colors:
- white
- gray
- black

- Adj list
- assumes every edge has weight 1

11. Topological Sort

BFS - every reachable vertex from s
shortest path (optimization)

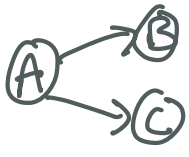
DFS - every reachable vertex from s
not optimizing!
- topo sort
- cycles

topo sort: ordering of vertices that respects dependencies

{directed, unweighted}

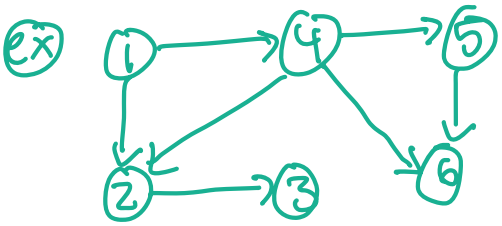
- A must appear in ordering before B

ex



valid: A, B, C
A, C, B

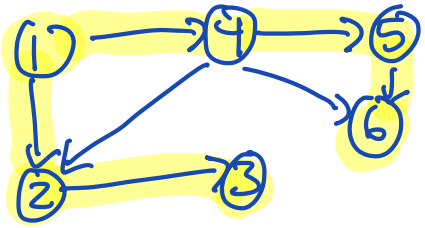
not valid: B, C, A
C, B, A
C, A, B
...



valid topo sort: 1, 4, 2, 3, 5, 6
1, 4, 5, 6, 2, 3 ✓
1, 4, 2, 5, 6, 3

2. DFS

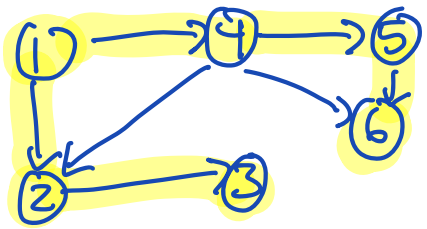
- start at some vertex
- go as far down the path as we can
- hit a deadend, backtrack until we can go forward again



			time	Finish times in rev. order
start at 1			1.s = 1	
• visit 2			2.s = 2	
• visit 3	dead end :(	done with 3	3.s = 3	
backtrack			3.f = 4	
• done with 2 backtrack	dead end :(	done with 2	2.f = 5	
Back at 1			4.s = 6	
• visit 4			5.s = 7	
• visit 5			6.s = 8	
• visit 6	dead end :(	done with 6	6.f = 9	
backtrack			5.f = 10	
Back to 5 backtrack	dead end :(	done with 5	4.f = 11	
Back to 4	dead end :(	done with 4	1.f = 12	
Back to 1	dead end :(	done with 1		

Finish times in rev. order: 1, 4, 5, 6, 2, 3

Done with ... 1, 4, 5, 6, 2, 3 topological sort.

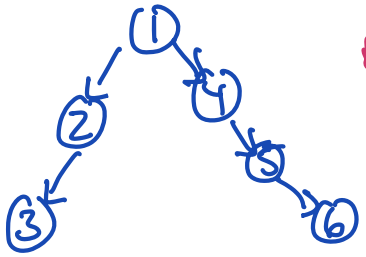


to keep track of:
- v.color (white, gray, black)
- v.s start time of v (int)
- v.f finish time of v
- v.π predecessor vertex

time 0, 1, ...

recursive DFS

Produce in DFS
- v.d values give us topo order
- DFS Tree
  ↳ same graph, dropping edges



DFS Tree

White path theorem
- in DFS tree, v is a descendent of u iff when we discover u, u ⇝ v is all white vertices

ex) 2 is a desc. of 1
3 is a desc of 1
when we discover 1, paths from 1→2, 1→3 are white

ex) 2 is not a desc of 4
when we discovered 4, we were done with 2

```
DFS(G)
1  for each vertex u ∈ G.V
2       u.color = WHITE
3       u.π = NIL
4  time = 0
5  for each vertex u ∈ G.V
6       if u.color == WHITE
7            DFS-VISIT(G, u)
```

DFS-Visit
- recursive
- time is a global variable
- vertices: w → g → b
- v.s, v.f, v.π
- adjacency list rep

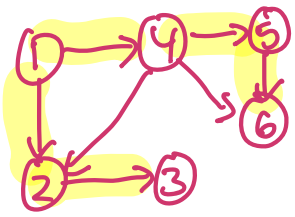
```
DFS-VISIT(G, u)
1  time = time + 1
2  u.s = time                  > vertex u starts
3  u.color = GRAY
4  for each v ∈ G.Adj[u]       → u's neighbors
5       if v.color == WHITE    ⇝ explore vertex if new
6            v.π = u
7            DFS-VISIT(G, v)   → recursive call to make it depth first
8  u.color = BLACK
9  time = time + 1             > vertex u ends
10 u.f = time
```

12:50

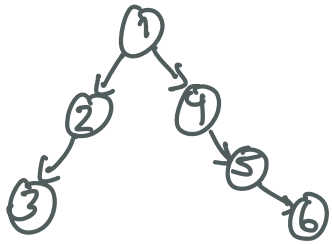
time=0
=1
=2
=3
=4

s=1 s=2
f=4 f=3

3. Cycles



graph G



DFS Tree

edge classification:

1. tree edge (in DFS tree)
2. forward edge (non-tree edge u, v where v is proper descendent of u)
3. back edge (edge u, v where v is an ancestor of u)
4. cross edge (none of the above)

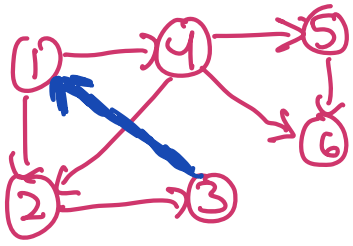
For G...

- tree edges (1,2) (2,3) (1,4) (4,5) (5,6)
- fwd edges (4,6)
- back edges ⇝ no back edges!
- cross edge (4,2)

DFS is good for:

- topo sort (finish times)
- graph is cyclic? (coloring)

What if we had a back edge?



- (3,1) would become
- then cyclic!