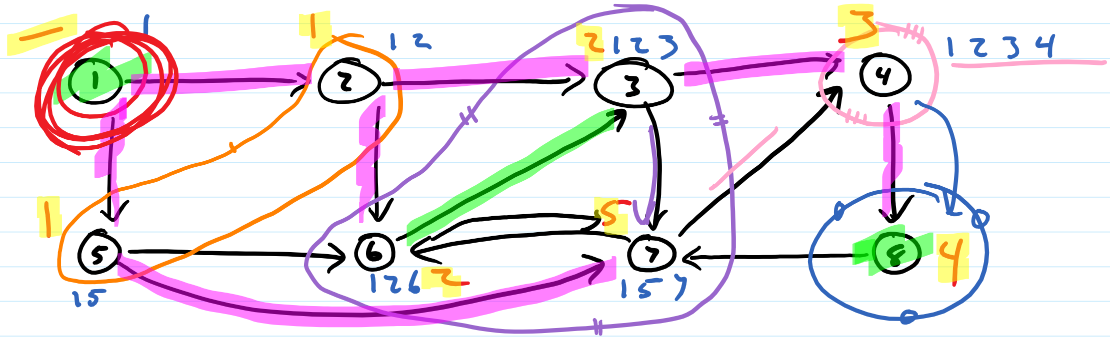
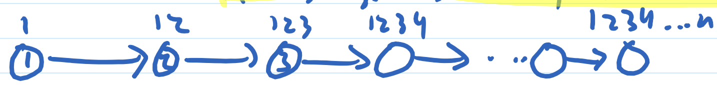
Breadth-First Search



starting from s
find verts 1 edge from s
find verts 2 edges from s
⋮
3 edges
4 edges

1 2 3 4 8

O(n²) storage if store entire path at each node...



~~1~~ ~~2~~ ~~5~~ ~~3~~ ~~6~~ 7 4 8

d=0 1 2 3 4

keep track of status unseen/processing/done
distance
parent

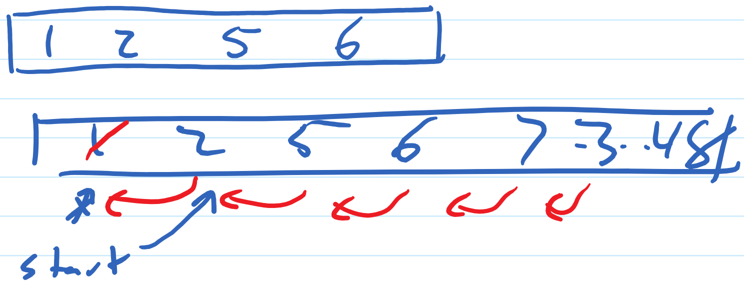
add unseen verts to queue as we see them

```
BFS(V,E,from)

for each vertex u in V
   color[u] <- UNSEEN
   d[u] <- infinity
   parent[u] <- NULL

color[from] <- PROCESSING
d[from] <- 0
parent[from] <- NULL
Q <- [from]

while not Q.isEmpty()
   u <- Q.dequeue()
   for each v adjacent to u
      if color[v] == UNSEEN
         parent[v] <- u
         d[v] <- d[u] + 1
         color[v] <- PROCESSING
         Q.enqueue(v)
   color[u] = DONE
```

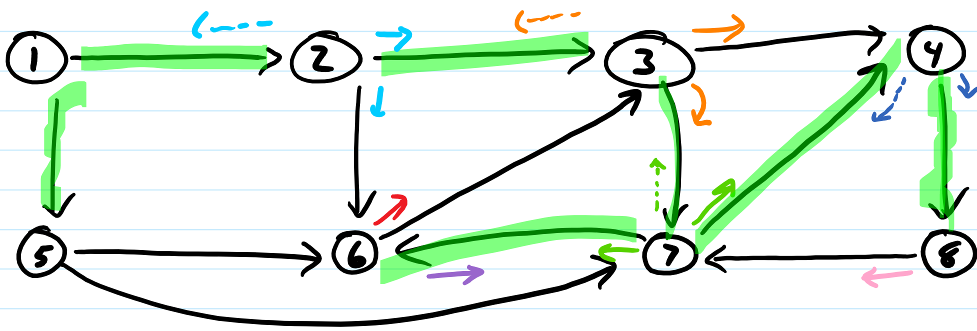


O(n) iterations

O(1) (using linked list or array w/ changing start index)

O(n+m) total for adj list (another form of for each vertex v
for each edge from v)

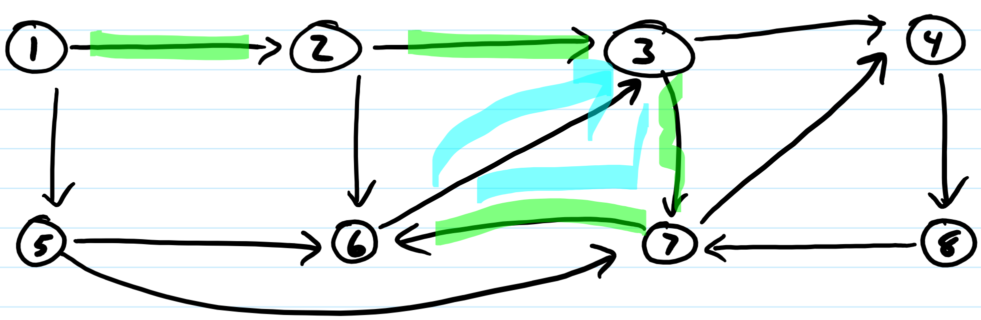
Depth-First Search



Depth-first search: Keep following edges from current vertex
Backtrack when no edges to unvisited vertices

finds cycles (when sees edge to processing vertex)

Cycles

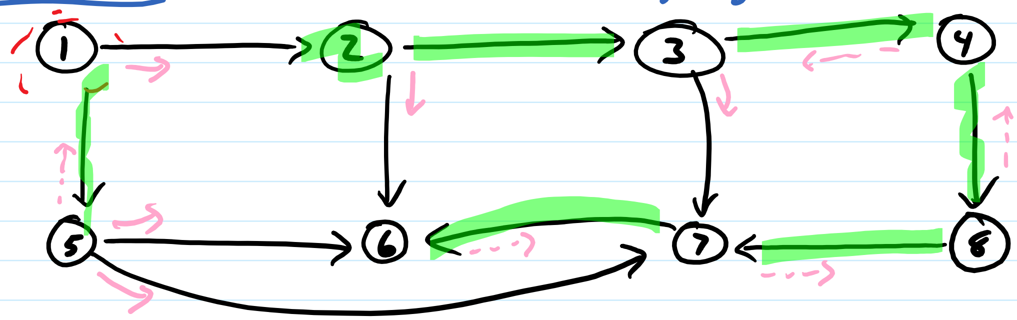


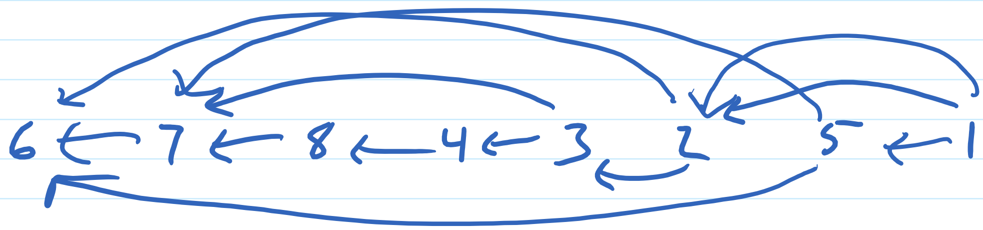
when examining $6 \to 3$, see that 3 is processing (recursive call not done)
↓
cycle!

Topological Sort – for a directed acyclic graph
orders verts so edges go in one direction

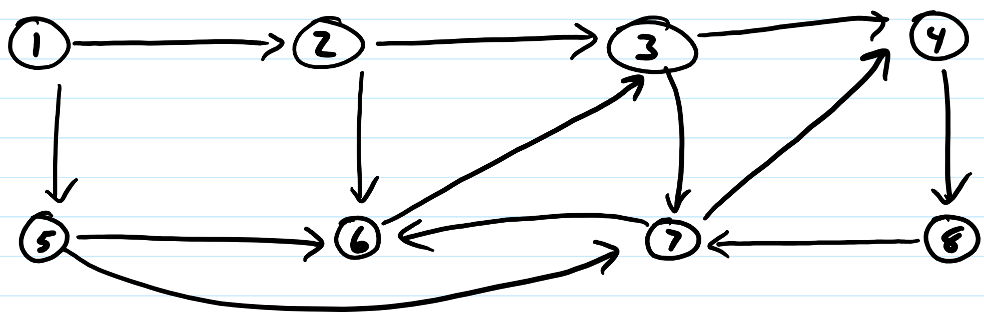
Topological Sort – for a directed acyclic graph

orders verts so edges go in one direction





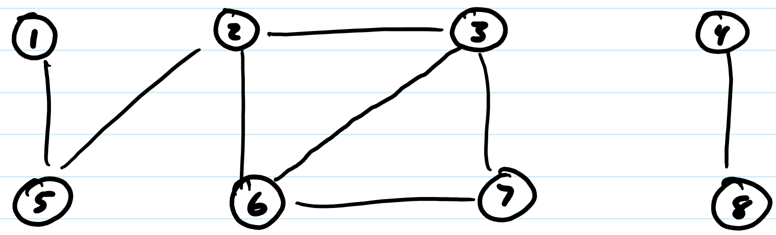
Longest Path



```
DFS-VISIT(G,u)
   color[u] <- PROCESSING
   discovery[u] <- time++

   for each v adjacent to u
      if (color[v] == UNSEEN)
         pred[v] <- u
         DFS-VISIT(G,v)

   finish[u] <- time++
   color[u] <- DONE
```



more useful in DFS-based algos
than distance

```
DFS(G)
   for each u in G.V
      color[u] <- UNSEEN

   time <- 0

   for each u in G.V
      if color[u] = UNSEEN
         DFS-VISIT(G,u)
```

to restart search
at new point when
completely stuck
(useful for some applications
of DFS, including topo sort)