Graphs: 2 Basic Algs

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

Extension to graphs

main diff: don’t revisit a vertex if we have seen it before!
no forward edges in BFS
for undir graphs, no cross edges in DFS

Implementation:

BFS(G,s):
    // idea 1: mark vertices when visited...
    // idea 2: use a data structure Q...
1. for each u ∈ V do unmark v
2. insert s to Q, mark s
3. while Q ≠ ∅ {
   4. remove a vertex u from Q
   5. for each out-neighbor v of u do {
      6. if v is unmarked
         insert v to Q. mark v,
6. \( n \rightarrow u \rightarrow \)
7. \( \text{insert } u \text{ to Q, mark } u, \text{ at tail } \)
   \( \text{parent}[u] = u, \text{ level}[u] = \text{level}(u) + 1 \)

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

+ total time \( O(n + \sum_{u\in V} \text{out-deg}(u)) \)

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

**DFS(G,s):**

// similar, but use a diff. data structure: stack, or recursion

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

**DFS-All(G):**

0. for each \( v \in V \) do unmark \( v \)
1. for each \( v \in V \) do
2. if \( v \) is unmarked, \( \text{DFS}(G,v) \)

**Runtime:** \( O(n + m) \) again

**Applications**
Applications

**Ex 1** Given dir/undir graph \( G = (V, E) \), \( s, t \in V \), find 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 \( \text{discovered}(u) \), \( \text{finished}(u) \))
Correctness:

\[ \exists \text{ back edge} \Rightarrow \exists \text{ cycle} \]
\[ \exists \text{ cycle} \Rightarrow \exists \text{ back edge:} \]

Hint: let \( u \) be first vertex discovered in cycle.
let \( u \) be vertex before \( u \) in cycle.

**Ex 5 Topological Sort**

Given dir. graph \( G=(V,E) \),
find a vertex ordering s.t.
\[ \forall (u,v) \in E \Rightarrow u \text{ appears before } v \]

e.g.

\[ \begin{array}{c}
\text{answer: } a, c, d, b \\
\end{array} \]

\[ \begin{array}{c}
\text{no answer. (because } \exists \text{ cycle) } \\
\end{array} \]