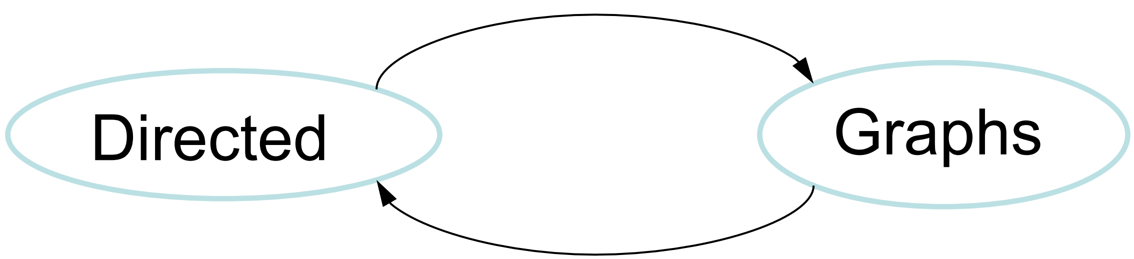
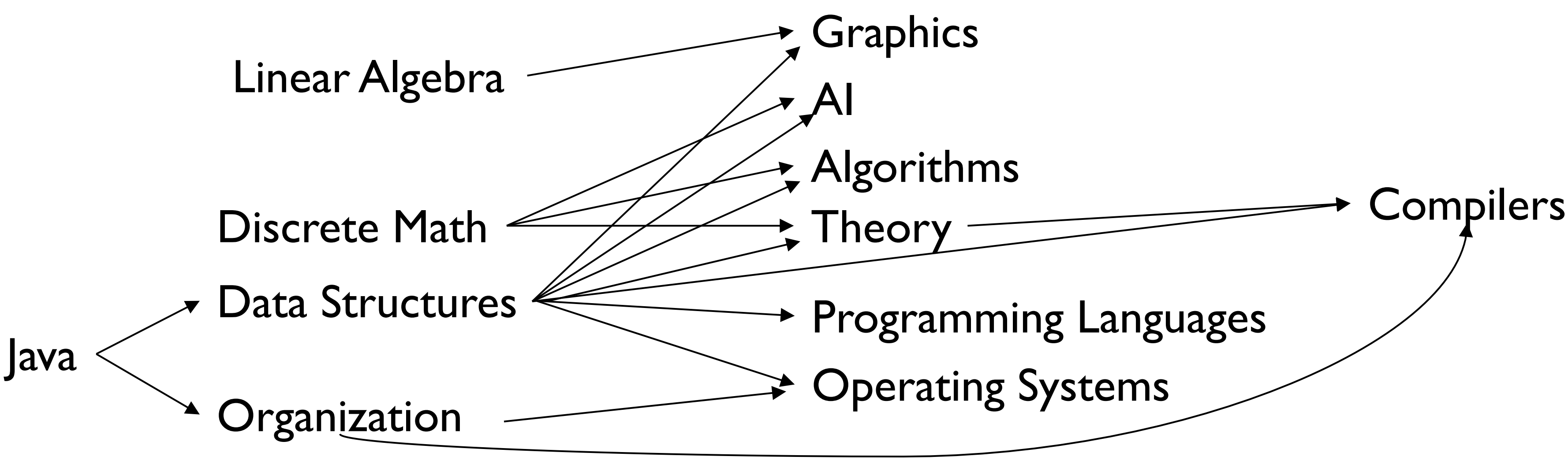
CSCI 136
Data Structures &
Advanced Programming



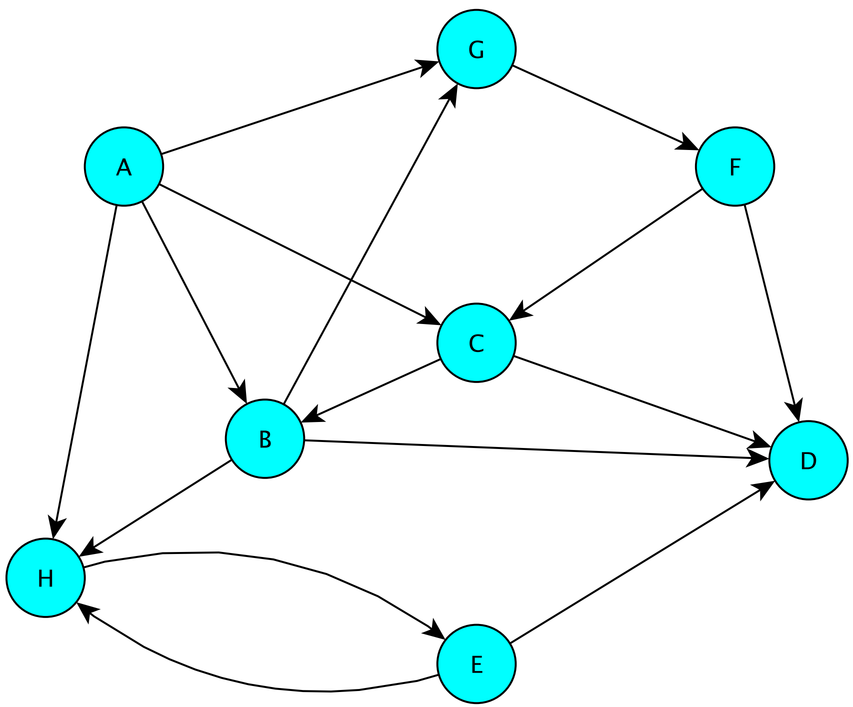
Directed Graphs



Def'n: In a *directed graph G = (V,E)*, each edge e in E is an *ordered* pair: e = (u,v) of vertices: its *incident vertices*.

- The *source* of e is u; the *destination/target* is v.
- Edge e = (u,v) goes *from* u *to* v
- Note: $(u,v) \neq (v,u)$

Directed Graphs



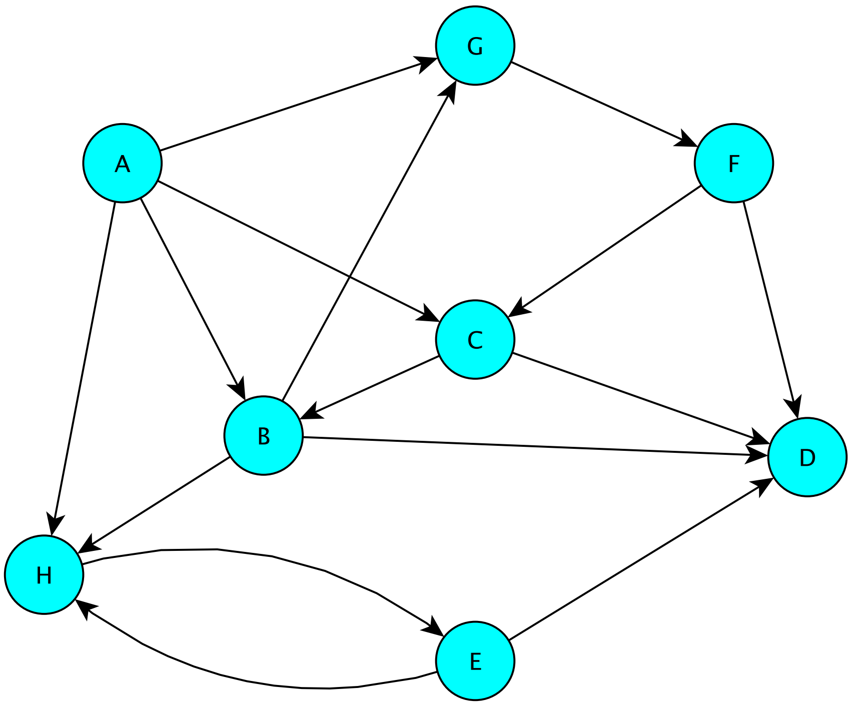
- The *out-neighbors* of B are D, G, H: B has *out-degree* 3
- The *in-neighbors* of B are A, C: B has *in-degree 2*
- A has in-degree 0: it is a *source* in G
- D has out-degree 0: it is a *sink* in G
- Not all graphs have sources/sinks

A *walk* is still an alternating sequence of vertices and edges

$$u = v_0, e_1, v_1, e_2, v_2, \ldots, v_{k-1}, e_k, v_k = v$$

but now $e_i = (v_{i-1}, v_i)$: all edges *point along direction* of walk

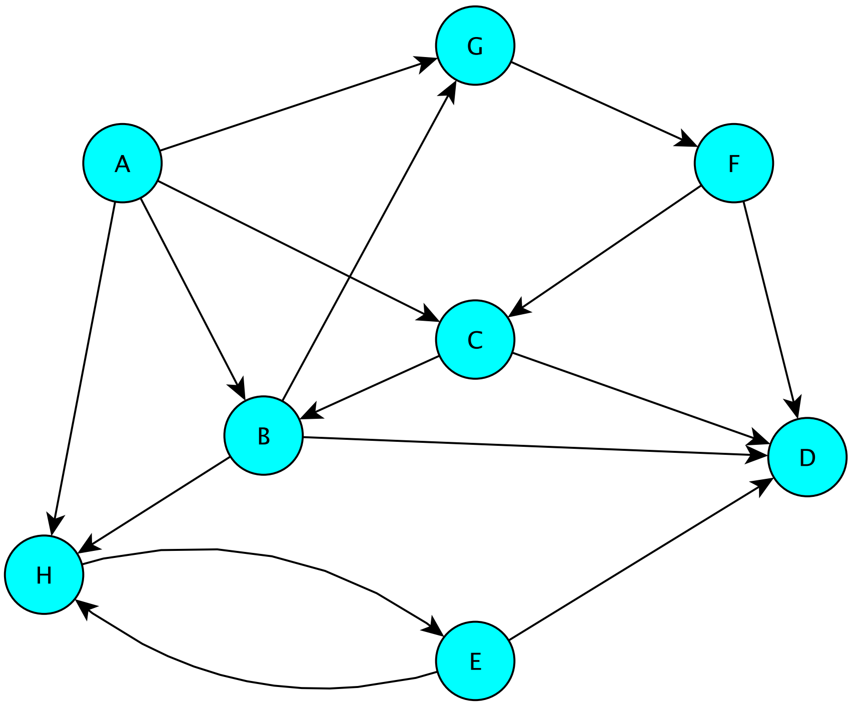
Directed Graphs



- A, B, H, E, D is a *directed walk* from A to D
- It's also a *(simple) path*
- D, E, H, B, A is *not* a walk from D to A
- B, G, F, C, B is a *directed cycle* (it's a *4-cycle*)
- So is H, E, H (a *2-cycle*)

- D is *reachable* from A (via directed path A, B, D), but A is *not reachable* from D
- In fact, *every* vertex is reachable from A

Directed Graphs



- A BFS of *G* from A visits every vertex
- A BFS of *G* from F visits all vertices but A
- A BFS of *G* from E visits only E, H, D
- Same is true for DFS

- BFS and DFS still find vertices reachable from start vertex
- But connectivity in directed graphs is more subtle than in undirected graphs!

Mutual Reachability

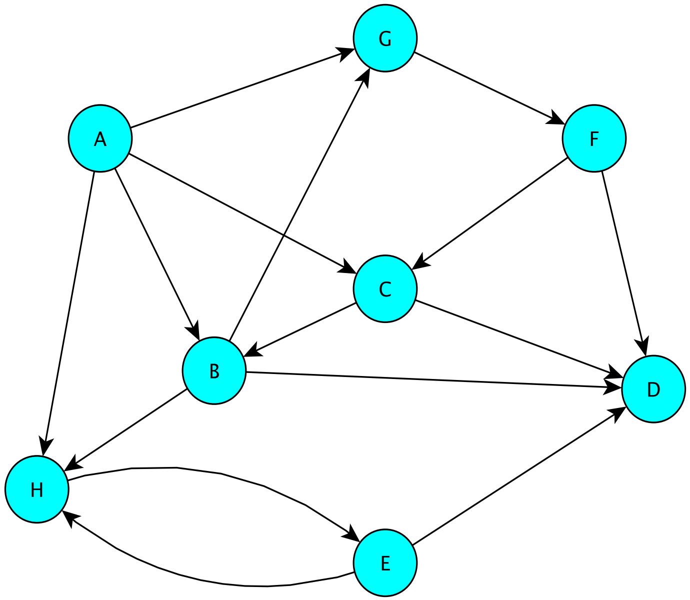
Vertices u and v in a directed graph G are *mutually reachable* if there are paths from u to v and from v to u

Note

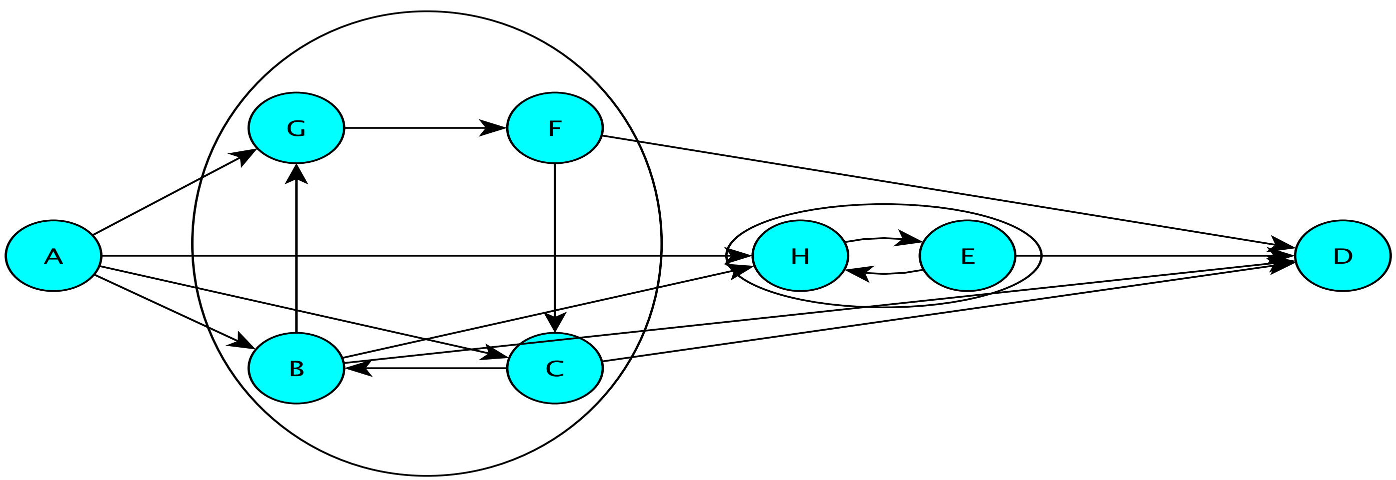
- For every vertex v, v and v are mutually reachable
  - Mutual reachability is a *reflexive* relation
- For every pair of vertices u,v: if u and v are mutually reachable, then v and u are mutually reachable
  - Mutual reachability is a *symmetric* relation
- For every triple u,v,w of vertices: If u and v are mutually reachable and v and w are mutually reachable, then u and w are mutually reachable
  - Mutual reachability is a *transitive* relation

Mutual reachability is an *equivalence relation*

Strong Components



- Vertices u and v are *mutually reachable* vertices if there are paths from u to v and v to u
- *Maximal* sets of mutually reachable vertices form *the strongly connected components* of G



Test For Strong Connectivity

A directed graph G is *strongly connected* if it has only one strong component.

- G is strongly connected *iff* for *every pair* of vertices u,v in G, u and v are mutually reachable

An "easy" test for mutual reachability

- Pick any vertex v_0 in G and find all u reachable from v_0
- If some u is *not* reachable from v_0, G is not S.C.
- Otherwise, build the *reverse graph* G_{rev} *of G*
  - (u,v) is in G_{rev} iff (v,u) is in G
- Now check to see if all u in G_{rev} are reachable from v_0
  - Note: u is reachable from v_0 in G_{rev} iff v_0 is reachable from u in G
- Can be used to find all strongly connected components

Summary & Observations

- Every edge in a directed graph has an *orientation*: The edge has a *source* vertex and a *destination* vertex
- This allows for modeling more complex, asymmetric relations between pairs of objects
- All of the concepts introduced for undirected graphs have analogs
  - degree(v) → inDegree(v), outDegree(v)
  - neighbor of v → in-neighbor/out-neighbor of v
  - walks/paths/cycles → directed walks/paths/cycles
  - connected component → strongly connected component
  - BFS and DFS still work to find all vertices reachable from a given vertex
    - As long as "neighbor" is replaced by "out-neighbor" in the search
- In later lectures we'll use these features to model important problems and develop efficient algorithms for solving them