CMPT307: Depth-First-Search

Week 11-1

Xian Qiu

Simon Fraser University

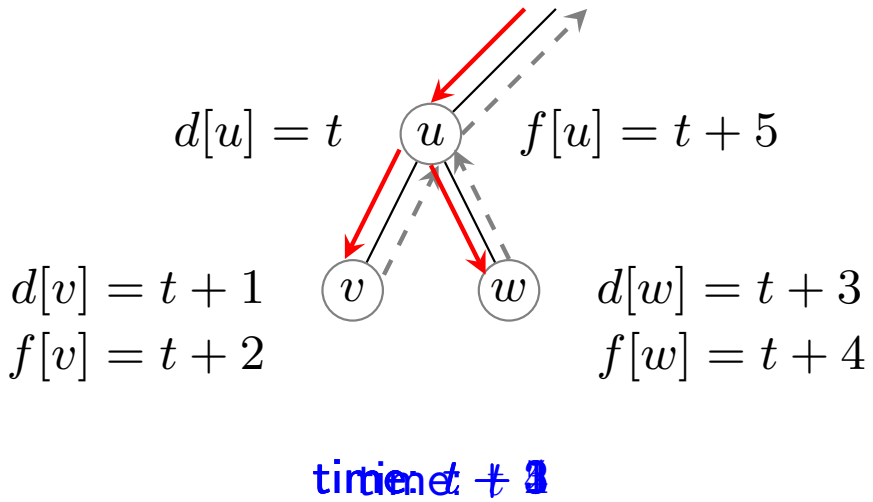
xianq@sfu.ca

Depth-First-Search

- BFS yields (unweighted) shortest path tree
  - shortest distances
  - searching paths
- DFS has applications in
  - topological sort
  - strongly connected components
- information saved in DFS
  - discovering times and finishing times
  - colors
  - searching paths

Discovering/Finishing Time

- $d[u]$: discovering time of u
- $f[u]$: finishing time of searching u's neighbors



Parenthesis properties

- $d[u] < f[u]$
- $[d[u], f[u]]$ entirely contains $[d[v], f[v]]$ — v is a descendant of u
- $[d[v], f[v]]$ and $[d[w], f[w]]$ are disjoint

Colors

coloring of vertices:

- white: undiscovered
- gray: discovered, but not yet scanned all of its neighbors
- black: discovered and all adjacent vertices have been scanned

Observations

- u is colored white before $d[u]$
- u is colored gray between $d[u]$ and $f[u]$
- u is colored black after $f[u]$

Depth-First Search

DFS(G, u)

```
   // initialization
1  for u ∈ V do
2      u.color = WHITE;
3      u.p = nil;
4  time = 0;
   // apply DFS to all connected components
5  for u ∈ V do
6      if u.color = WHITE then
7          DFS-VISIT(G, u);
```

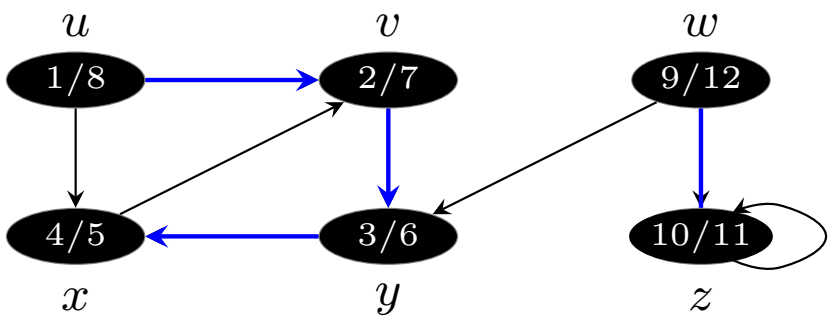
- ▷ DFS output a depth-first forest

Depth-First Search

DFS-VISIT(G, u)

```
1  time = time +1;
2  u.d = time;                                   // discovering time
3  u.color = GRAY;
4  for each v that is adjacent to u do
5      if v.color == WHITE then
6          v.p = u;                              // record searching path
7          DFS-VISIT(G, v);
8  u.color = BLACK;
9  time = time + 1;
10 u.f = time;                                   // finishing time
```

Example



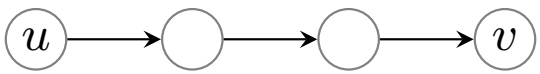
discovering time/finishing time

what if start from w?

White-Path Theorem

Theorem

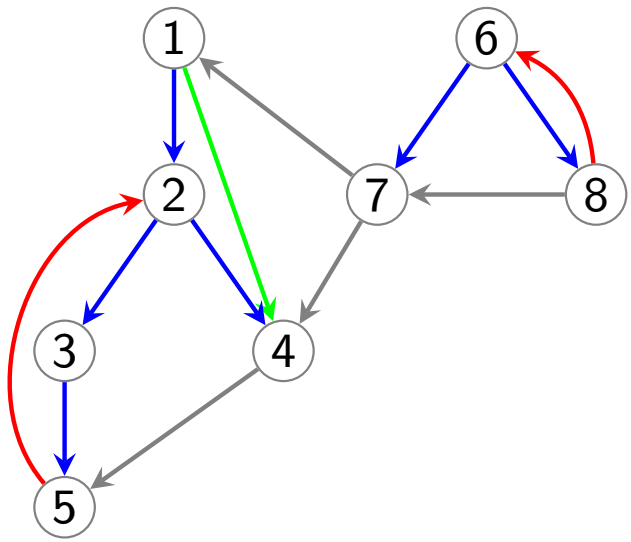
In a depth-first forest of G, v is a descendant of u iff there is a path from u to v consisting entirely of white vertices at time u.d.



- $(\Rightarrow)$ u is discovered before its descendants, at time u.d (before coloring it gray), u is white, so are the descendants
- $(\Leftarrow)$ u.d $<$ v.d $<$ v.f $<$ u.f and by parenthesis properties

Edge Classification

- tree edge: edges of DFS forest
- forward edge: tree ancestor to tree descendant
- back edge: tree descendant to tree ancestor
- cross edge: either vertex is not an ancestor of another, or edges between two trees



DFS for directed graph

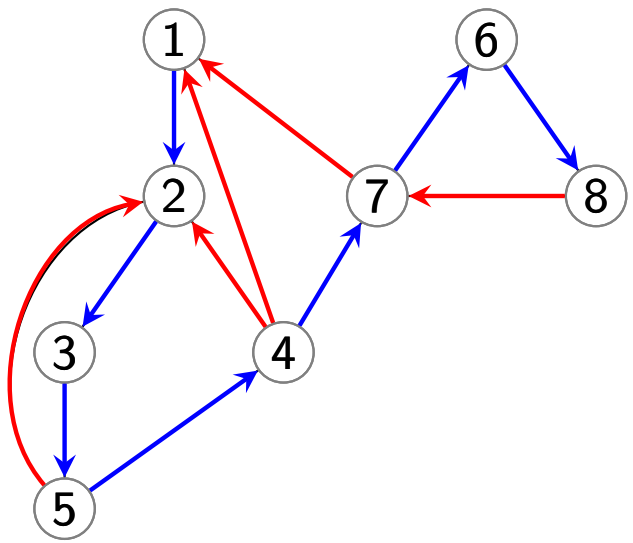
DFS in Edge Classification

when exploring an edge (u, v), examine the color of v:

- ▷ WHITE indicates a tree edge
- ▷ GRAY indicates a back edge
- ▷ BLACK indicates a forward or cross edge

Theorem

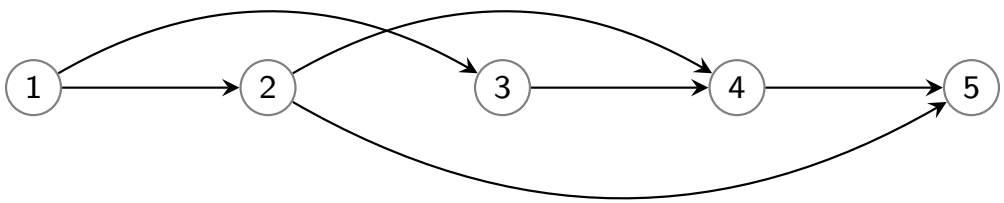
In a DFS of undirected G, every edge is either a tree edge or a back edge.



Directed Acyclic Graphs

Definition

A directed graph $G = (V, E)$ has a linearization if there is a bijection $\pi : V \to \{1, \ldots, |V|\}$ such that $\pi(u) < \pi(v)$, $\forall (u, v) \in E$.



all arcs from left to right

Directed Acyclic Graphs

Theorem

A digraph $G = (V, E)$ has a linearization iff it has no directed cycle.

Proof

($\Rightarrow$) a cycle $(u, v, \ldots, u)$ yields $\pi(u) < \pi(v) < \ldots < \pi(u)$

($\Leftarrow$) acyclic graphs have a node without incoming arcs

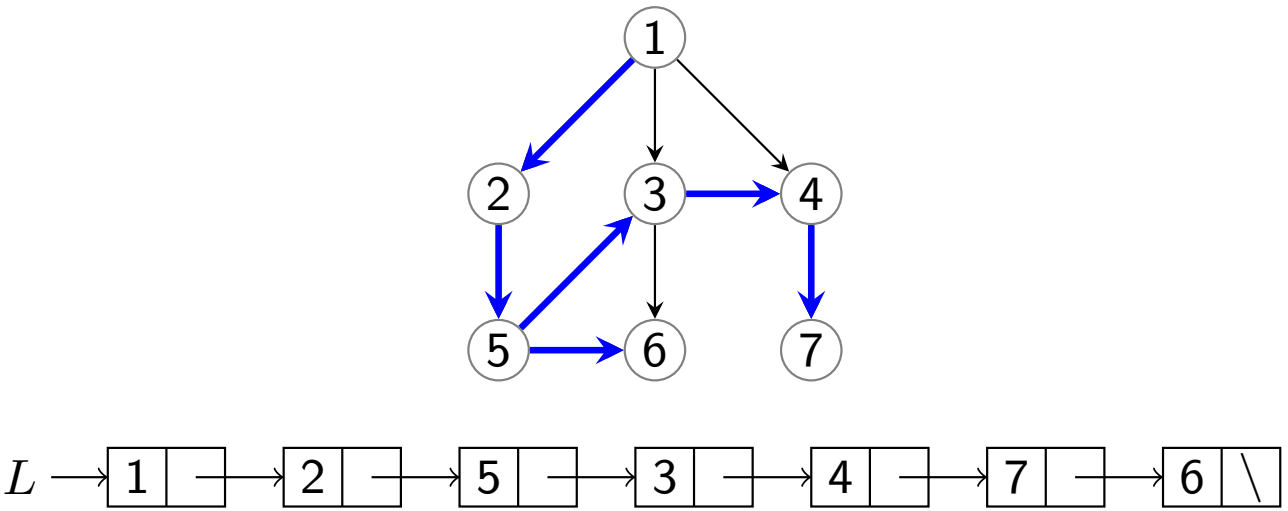
- ▷ assume $u \in V$ has no incoming arcs and let $\pi(u) = 1$
- ▷ delete all arcs $(u, v) \in E$ and consider $G - u$ (which is acyclic)
- ▷ by induction on $G - u$, we get a linearization

topological sort: find a linearization of a DAG

- ▷ from the proof
- ▷ apply DFS

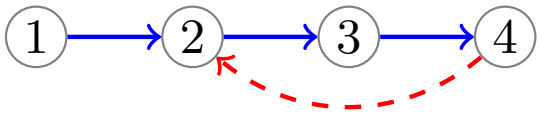
Topological Sort by DFS

- $G = (V, E)$ is a DAG and $v \in V$
- call $\mathrm{DFS}(G)$ to compute finishing time $v.f$ for each v
- as each v is finished, insert it onto the front of a linked list



Correctness

- no back edges in each tree



- no "backward" edges in between trees