Recitation 8 - DFS Edge Classification

Sebastian Laudenschlager

sebastian.laudenschlager@colorado.edu

March 16, 2018

DFS

- Recall how DFS works:
  - Explore a vertex, then explore one undiscovered adjacent vertex, and keep going until no undiscovered vertices remain.
  - DFS then backtracks to explore adjacent vertices that might have been missed at its predecessor.
  - Keep going until all paths from the source node have been explored.

DFS Pseudocode

```
def DFS(G):
    for each vertex u in G.V:
        u.color = white
        u.pred = NIL
    time = 0
    for each vertex u in G.V:
        if u.color == white:
            DFS-Visit(G, u)
```

DFS Pseudocode, cont.

```
def DFS-Visit(G, u):
    time += 1
    u.d = time
    u.color = gray
    for each v in G.adj[u]:
        if v.color == white:
            v.pred = u
            DFS-Visit(G, v)
    u.color = black
    time += 1
    u.f = time
```

How DFS works

- DFS simply initializes all vertices in G to be white and have no predecessor.
- Also initializes the global time value, used to give each vertex a discovery time, $u.d$, and finishing time, $u.f$.
- Then it goes through each vertex u and does a depth first search if u is colored white.

How `DFS-Visit` works

- First set discovery time and color vertex u gray.
- Go through the adjacency list of u and explore the first one that is colored white via another `DFS-Visit` (recursion).
- Once the adjacency list of vertex u has been fully explored, we color it black and set its finishing time $d.f$.

DFS Properties

- Just like BFS, DFS may not explore all edges.
- The results of DFS depend on the order of how the adjacency list is explored, and on the order in which the vertices themselves are explored.

Running Time

- The two loops in `DFS` take $\mathcal{O}(V)$ each, without looking at the `DFS-Visit` calls.
- `DFS-Visit` is called once for each vertex v.
- The loop in `DFS-Visit` walks through the adjacency list of v, for every $v \in V$.
- Since $\sum_{v \in V} |adj(v)| = \Theta(E)$, we can see that the running time of `DFS` is $\mathcal{O}(V + E)$.

Edge classification

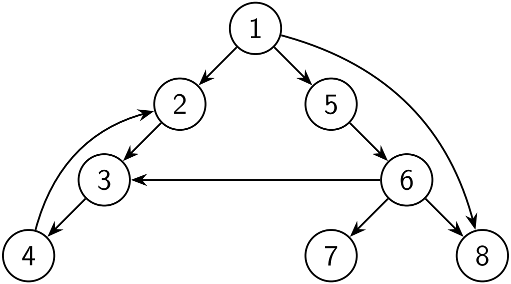
- DFS can classify edges into four categories:
  - **Tree edge:** If v is visited for the first time as we traverse the edge (u, v), then the edge (u, v) is a tree edge.
  - **Back edge:** If v is an ancestor of u, then (u, v) is a back edge.
  - **Forward edge:** If v is a descendant of u, then (u, v) is a forward edge.
  - **Cross edge:** If v is neither an ancestor or descendant of u, then (u, v) is a cross edge.

DFS edge classification

- If we are exploring an edge (u, v) and v is currently marked as white, (u, v) gets classified as a tree edge.
- If v is marked gray, (u, v) is a back edge.
- If v is marked black and $u.d < v.d$, (u, v) is a forward edge.
- If v is marked black and $u.d > v.d$, (u, v) is a cross edge.

Example

- Consider the following directed graph:



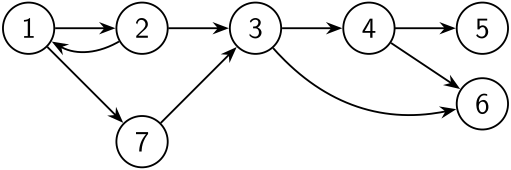
- Let's run a DFS on this graph and see if we can classify all edges.

Example, cont.

- Running DFS starting at 1, we find that:
  - **Tree edges:** $(1, 2)$, $(2, 3)$, $(3, 4)$, $(1, 5)$, $(5, 6)$, $(6, 7)$, $(6, 8)$.
  - **Back edges:** $(4, 2)$
  - **Forward edges:** $(1, 8)$
  - **Cross edges:** $(6, 3)$

Practice

- Consider the following directed graph:



- Perform a DFS on this graph to classify all edges in this graph.

Edge classification theorem

- In a DFS of an undirected graph G, every edge in G is either a tree edge or a back edge.
- **Proof:**
  - Let (u, v) be some edge in G, and assume that $u.d < v.d$.
  - This means that v must finish before u since v is on u's adjacency list.
  - If we first explore the edge (u, v) in the direction of $u \to v$, then v is undiscovered at that time (white), meaning that (u, v) is a tree edge.
  - If we first explore (u, v) in the direction of $v \to u$, then (u, v) is a back edge since u would be colored gray at that point.