Lecture 11 DFS and BFS

Thursday, February 21, 2019 2:45 PM

- Recap: Pre/post order

DFS_visit(1) → DFS_visit(2) → DFS_visit(4) → DFS_visit(3) → DFS_visit(5)

1 2 4 3 4 2 5 2 1

Pre-order: 1 2 4 3 5
Post-order: 3 4 5 2 1

- Types of edges

DFS-tree

1. Tree edges: edges in the DFS tree.
2. Forward edges: edge connecting a vertex to one of its descendants in the tree.
3. Backward edges: edge connecting a vertex to one of its ancestors in the tree.
4. Cross edges: edges that connect a vertex to another vertex in a different branch.

- Similar to DFS tree/pre-post order, edge types also depend on the choice in DFS algorithm.

- Forward edge (u, v)
- **Visit U** $\rightarrow$ **Visit U** $\rightarrow$ **DFS-visit(U)** returns $\rightarrow$ **Consider (u,v)** $\rightarrow$ **DFS-visit(u)** return

- **back edge (u,v)**

- **Visit V** $\rightarrow$ **Visit U** $\rightarrow$ **Consider (u,v)** $\rightarrow$ **DFS-visit(U)** return $\rightarrow$ **DFS-visit (V)** returns

- **Cross edge (u,v)**

- **Visit V** $\rightarrow$ **DFS-visit(U)** returns $\rightarrow$ **Visit U** $\rightarrow$ **Consider (u,v)** $\rightarrow$ **DFS-visit(u)** returns

- **Cycle-Finding**

  - **Proof of correctness.**

  - **Lemma:** For a DFS algorithm, if when U is visited there is a path from U to V, and U is the only visited vertex on the path, then U is on the stack when V is visited. (U is later than V in post-order)

  - **Proof:** we prove this by induction on the length of the path.

  - **Induction Hypothesis:** For a DFS algorithm, if when U is visited, there is a path of length ≤ l from U to V such that U is the only visited vertex on the path, then U is on the stack when V is visited.

  - **Base case:** l=1

    in this case, (u,v) is an edge and V is not visited when U is visited. We know the sequence of following 3 events, by the DFS-visit algorithm.

    $$\text{Visit(U)} \rightarrow \text{Consider edge (u,v)} \rightarrow \text{DFS-visit(u)} \text{ returns}$$

    u must be visited between events (1) and(2) or right after event (2)
so when $U$ is visited we always know $U$ is on the stack.

- Induction step: Assume IH is true for $l=K$. Consider the case when the path has length $K+1$. Consider $W$, the vertex before $V$ on the path. We know there is a path from $U$ to $W$ of length $K$, and $U$ is the only visited vertex.

By IH, we know $U$ is on stack when $W$ is visited, by design of algorithm, we know the sequence of the following events

- $\text{Visit } U$ 
- $\text{Visit } W$ 
- $\text{edge}(W, V)$ considered
- $\text{DFS-visit}(W)$ returns
- if $V$ was visited before edge $(U, V)$ is considered, then it is between $O$ and $2$, so $V$ is visited while $U$ is on stack.
- otherwise, $V$ is going to be visited right after $O$, at that time $U$ is still on top stack. This finishes the induction.

- Counter-example if we do not require $U$ to be the only visited vertex.

Let $U = 3$, $V = 4$

We run DFS on $O$, and choose edge $(1, 2)$ first.

When $3$ is visited, $4$ is not visited, and there is a path $(3, 1, 4)$ (except $3$ has been visited)

So the version of the lemma that only requires $V$ to be visited after $U$ will predict $3$ is on stack when $4$ is visited. However, this is not true. Only $W$ is on stack when $2$ is visited.

- Proof for cycle finding:
  - Assume there is a cycle $(U_1, U_2, \ldots, U_k)$
- Proof for cycle finding:
  - assume there is a cycle \((V_1, V_2, \ldots, V_k)\)
  - assume (w.l.o.g.) that \(V_1\) is the first vertex visited in the cycle.
  - by lemma: we know \(V_1\) is still on stack when DFS visits \(V_k\).
  - now when edge \((V_k, V_1)\) is considered, it must be a back edge.

- BFS
  - Lemma: BFS finds shortest path from \(U\) to any vertex \(V\) reachable from \(U\).
  - Proof: IH: BFS finds shortest path for all vertices \(V\) at a distance \(\leq L\) to \(U\).
    - Base case: \(L=1\) (\(U,V\)) is an edge.
      - Since BFS first considers all neighbors of \(U\), (\(U,V\)) will be considered and BFS finds the shortest path.
    - Induction step: assume IH is true for \(L=K\), consider a vertex \(V\) at distance \(K+1\) to \(U\.
      - the shortest path from \(U\) to \(V\) has length \(K+1\)
      - consider \(W\): vertex before \(V\) on the shortest path.
        - distance from \(U\) to \(W\) is \(K\).
        - using IH, BFS finds shortest path to \(W\).
      - now consider the time \(W\) is processed in BFS:
        - 1) \(V\) is already in the queue
          - in this case, \(V\) is added to the queue by a vertex \(W\) that is processed before \(W\).
          - by design \(\text{dist}(U,W) \leq \text{dist}(U,V) = K\)
          - so BFS finds a path of length \(\leq K+1\) \(\checkmark\)
        - 2) \(V\) is not in the queue
          - BFS will add \(V\) to the queue and find a path of length \(K+1\) \(\checkmark\)