Mini-Lecture
Loops

I. Definition of Natural Loops
II. Algorithm to find loops

Reading: Chapter 9.6
What is a Loop?

• **Goals:**
  – Define a loop in graph-theoretic terms (control flow graph)
  – Not sensitive to input syntax
  – A uniform treatment for all loops: DO, while, goto’s

• **Programs typically have only “natural loops”**

• **A “natural” loop has**
  – edges that form at least a cycle
  – a single entry point

![Diagram of a loop with nodes labeled a, b, c, d and an entry point labeled start.](image)
Dominators

- Node $d$ dominates node $n$ in a graph ($d \text{ dom } n$):
  - if every path from the start node to $n$ goes through $d$
    - a node dominates itself

- Immediate dominance:
  $d \text{idom } n : d \text{ dom } n, d \neq n, \neg \exists m \text{ s.t. } d \text{ dom } m \text{ and } m \text{ dom } n$

- Immediate dominance relationships form a tree
Finding Dominators

• **Definition**
  - Node \( d \) dominates node \( n \) in a graph \( (d \text{ dom } n) \)
  if every path from the start node to \( n \) goes through \( d \)

• **Formulated as MOP problem:**
  - node \( d \) lies on all possible paths reaching node \( n \) \( \Rightarrow d \text{ dom } n \)
    - Direction:
    - Values:
    - Meet operator:
    - Top:
    - Bottom:
    - Boundary condition: start/exit node =
    - Finite descending chain?
    - Transfer function:

• **Speed:**
  - With reverse postorder, solution to most flow graphs
    (reducible flow graphs) found in 1 pass
Definition of Natural Loops

- Single entry-point: **header** ($d'$)
  - a header dominates all nodes in the loop

- A **back edge** ($n \rightarrow d'$) in a flow graph is
  - an edge whose destination dominates its source ($d \text{ dom } n$)

- The **natural loop of a back edge** ($n \rightarrow d'$) is
  
  $d' + \{ \text{nodes that can reach } n \text{ without going through } d' \}$
Constructing Natural Loops

• The natural loop of a back edge \((n \rightarrow d')\) is
  \[ d' + \{\text{nodes that can reach } n \text{ without going through } d'\} \]

• Remove \(d'\) from the flow graph, find all predecessors of \(n\)

• Example:

\[ \begin{array}{c}
1 \text{ (start)} \\
\downarrow \\
2 \\
\downarrow \\
3 \\
\downarrow \\
4 \\
\downarrow \\
5 \\
\downarrow \\
6 \\
\downarrow \\
7 \\
\downarrow \\
8 \\
\end{array} \]
Inner Loops

• If two loops do not have the same header:
  – they are either disjoint, or
  – one is entirely contained (nested within) the other
    • inner loop: one that contains no other loop.

• If two loops share the same header:
  – Hard to tell which is the inner loop
  – Combine as one
Graph Edges

- **Depth-first spanning tree**
  - Edges traversed in a depth-first search of the flow graph form a depth-first spanning tree

- **Categorizing edges in graph**
  - **Advancing** edges: from ancestor to proper descendant (incl. spanning tree)
  - **Retreating** edges: from descendant to ancestor (not necessarily proper)
  - **Cross** edges: all other edges
Back Edges

• Definition
  – **Back edge**: \( n \rightarrow d, d \text{ dom } n \)

• Relationships between graph edges and back edges
  – a back edge must be a retreating edge
    dominator \( \Rightarrow \) visit \( d \) before \( n \), \( n \) must be a descendant of \( d \)
  – a retreating edge is not necessarily a back edge

• **Most programs (all structured code, and most GOTO programs):**
  – retreating edges = back edges
Summary

• Define loops in graph theoretic terms

• Definitions and algorithms for
  • Dominators
  • Back edges
  • Natural loops