CS536

Loop Optimizations

A Sahu

CSE, IIT Guwahati

Outline

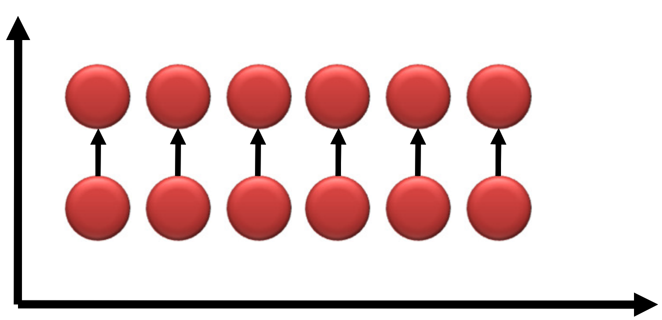
- Basic Loop Optimization
- Basic Data Flow Analysis

Transformations:Permutation

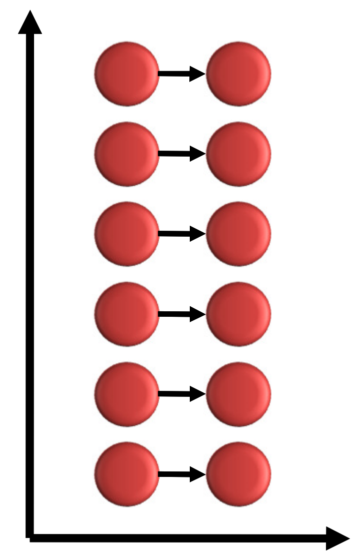
```
for(i=1;i<=N;i++){
   for(j=1;j<=M;j++){
      Z[i][j]=Z[i-1][j];
   }
}
```

Permutation

```
for(p=1;p<=M;p++) {
   for(q=1;q<=N;q++) {
      Z[q][p]=Z[q-1][p];
   }
}
```



$$\begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix}$$



//confusion in class, it was Z[p][q] instead of Z[q][p]

MidSem Paper Discussion

Loop Optimizations

Loop Optimizations

- Most important set of optimizations
  - Programs are likely to spend more time in loops
- Presumption: Loop has been identified
- Optimizations:
  - Loop invariant code removal
  - Induction variable strength reduction
  - Induction variable reduction

Loops in Flow Graph

- **Dominators**: A node *d* of a flow graph *G* dominates a node *n*, if every path in *G* from the initial node to *n* goes through *d*.

  Represented as: ***d dom n***

- Corollaries:
  - Every node dominates itself.
  - The initial node dominates all nodes in *G.*
  - The entry node of a loop dominates all nodes in the loop.

Loops in Flow Graph

- Each node ***n*** has a unique ***immediate dominator m***, which is the last dominator of *n* on any path in *G* from the initial node to *n*.

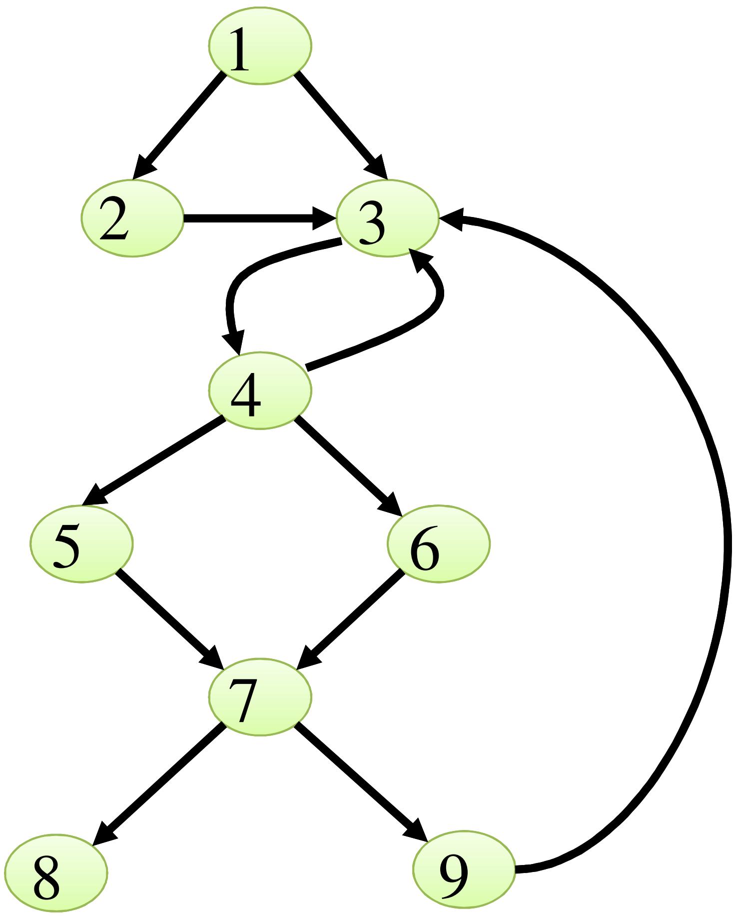
  (d ≠ n) && (d dom n) → d dom m

- Dominator tree (*T*):

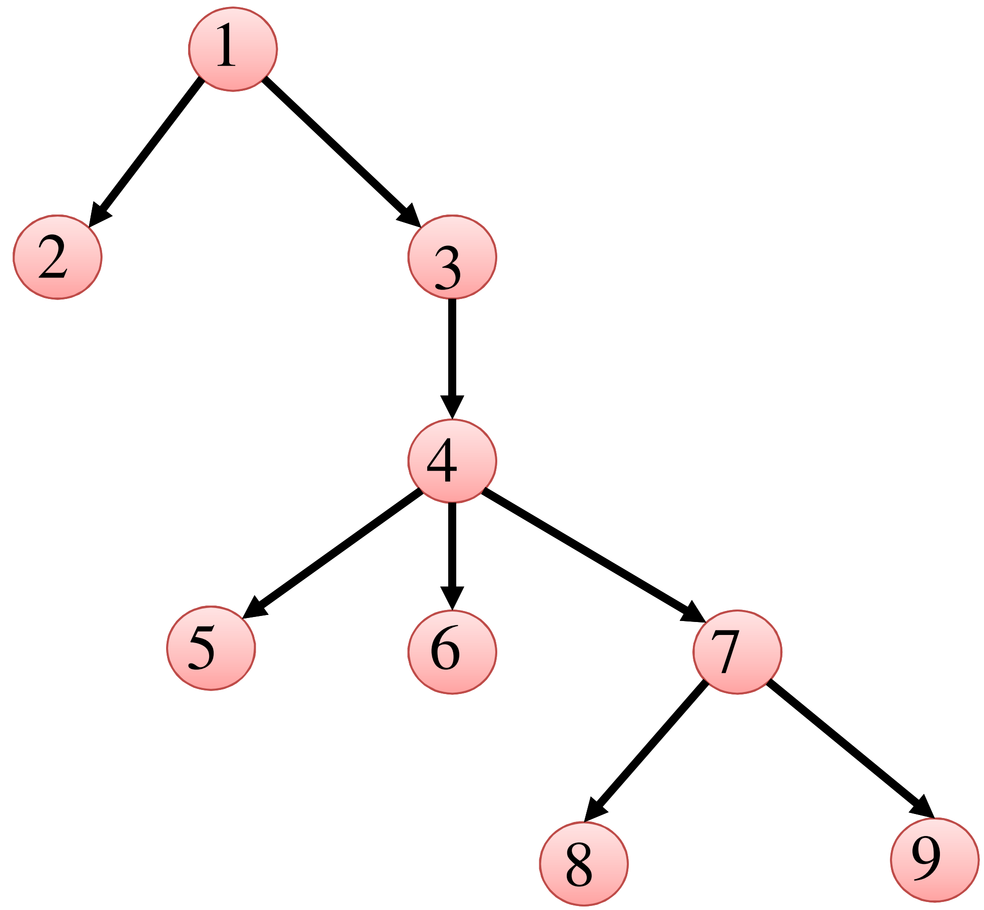
  A representation of dominator information of flow graph *G*.

  - The root node of *T* is the initial node of *G*
  - A node *d* in *T* dominates all node in its sub-tree

Example: Loops in Flow Graph



Flow Graph



Dominator Tree

Loops in Flow Graph

- **Natural loops:**
  1. A loop has a single entry point, called the "**header**". Header dominates all node in the loop
  2. There is **at least one path back to the header from the loop nodes** (i.e. there is at least one way to iterate the loop)
- Natural loops can be detected by *back edges*.
  - ***Back edges***: edges where the sink node (head) dominates the source node (tail) in *G*

Natural loop construction

- Construction of natural loop for a back edge

  Input: A flow graph G, **A back edge $n \rightarrow d$**

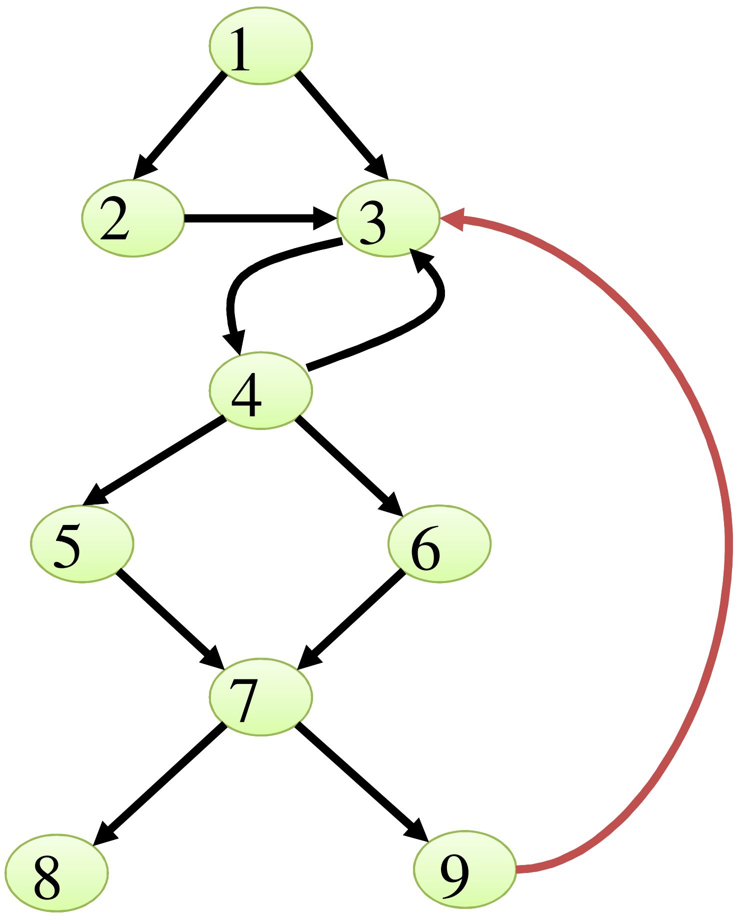
  Output: The set *loop* consisting of all nodes in the natural loop of $n \rightarrow d$

  Method:

  stack := ϵ ; *loop* := {*d*};
  insert(*n*);
  while (*stack* not empty)
  m := *stack*.pop();
  for each predecessor *p* of m do
  insert(*p*)

```
Fun: insert (m) {
  if !(m ϵ loop) {
    loop = loop ∪ {m}
    stack.push(m)
  }
}
```

Natural loop construction



Back edge n=9, d= 3
Stack: ϵ, Loop={3}

insert(9):
stack={9}, loop={3, 9}

pop(9): insert (7)
stack={7}, loop={3, 9,7}

pop(7): insert (5,6)
stack={5,6}, loop={3, 9,7, 5, 6}

pop(5,6): insert (4)
stack={4}, loop={3, 9,7, 5, 6,4}

pop(4): insert (3) , ***!(m ϵ loop)***
stack={}, loop={3, 9,7, 5, 6,4}

Inner loops

- Property of natural loops:
  - If two loops l_1 and l_2, do not have the same header,
    - l_1 and l_2 are disjoint.
      - *l2 can be compressed to one Loop Stmt*
    - **One is an inner loop of the other.**
- Inner loop: loop that contains no other loop.
    - Loops which do not have the same header.

Thanks