Program Analysis

Path Profiling (Part 2)

Prof. Dr. Michael Pradel

Software Lab, University of Stuttgart

Winter 2020/2021

Outline

1. **Motivation and Challenges**
2. **Ball-Larus algorithm for DAGs** 
3. **Generalization and Applications**

Mostly based on this paper:

- *Efficient path profiling*, Ball and Larus, MICRO 1996

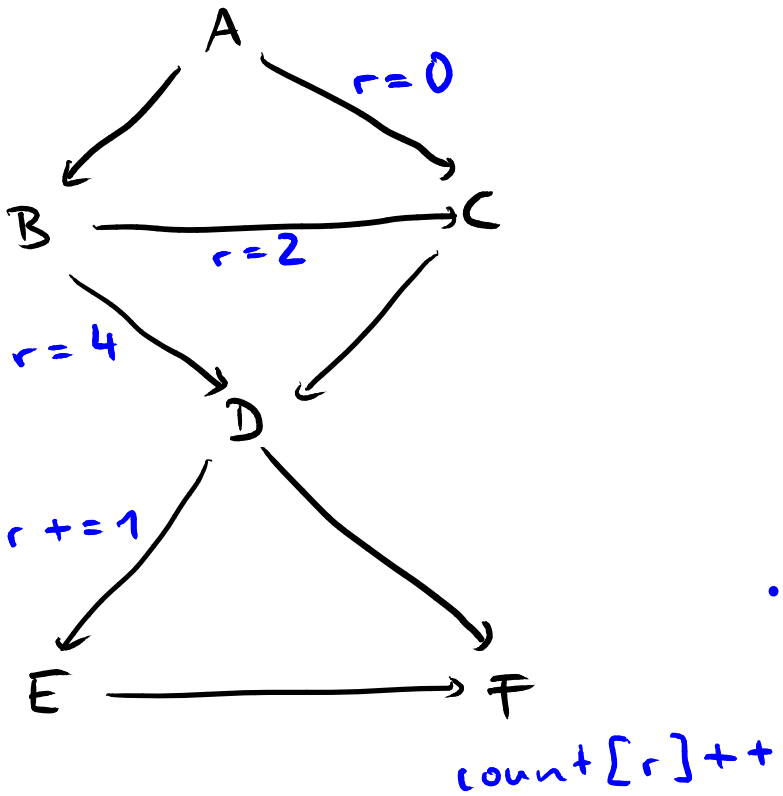
Other reading material:

- *Whole program paths* , Larus, PLDI 1999
- *HOLMES: Effective statistical debugging via efficient path profiling*, Chilimbi et al., ICSE 2009

Ball-Larus Algorithm

- **Assign a number to each path**
- **Compute path number by incrementing a counter at branching points**
- **Properties of path encoding**
  - Precise: A single unique encoding for each path
  - Minimal: Instruments subset of edges with minimal cost

Example: Path Encoding



Instrumentation:

- state / counter : r
- array of counts : counts

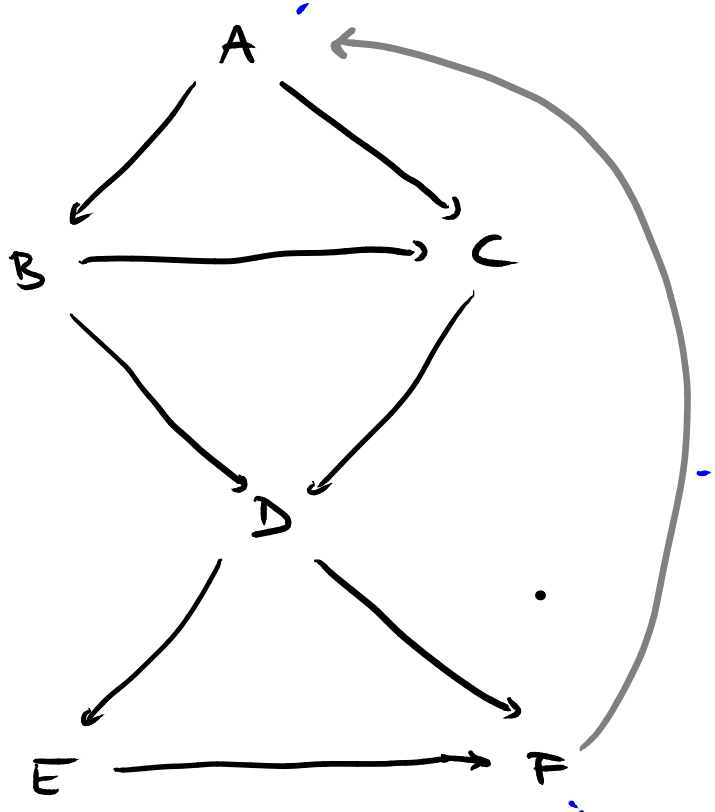
Path	Encoding
ACDF	0
ACDEF	1
ABCDF	2
ABCDEF	3
ABDF	4
ABDEF	5

Algorithm for DAGs

Assumptions

- Control flow graph is a directed acyclic graph (DAG)
- n paths (numbered 0 to $n-1$)
- Graph has unique entry and exit nodes
- Artificial back edge from exit to entry

Assumptions



... unique entry node

Graph is acyclic

... artificial back edge

... unique exit node

Algorithm: Overview

- **Step 1: Assign integers to edges**
  - Goal: Sum along a path yields unique number for path
  - Enough to achieve ”precise” goal
- **Step 2: Assign increment operations to edges**
  - Goal: Minimize additions along edges
  - Instrument subset of all edges
  - Assumes to know/estimate how frequent edges are executed

Representing Paths with Sums

- **Associate with each node a value:**
  $NumPaths(n)$ **= number of paths from** n **to exit**
- **Computing** $NumPaths$
  - Visit nodes in reverse topological order
  - If n is leaf node:
    $NumPaths(n) = 1$
  - Else:
    $NumPaths(n) =$ sum of $NumPaths$ of destination of outgoing edges

Representing Paths with Sums (2)

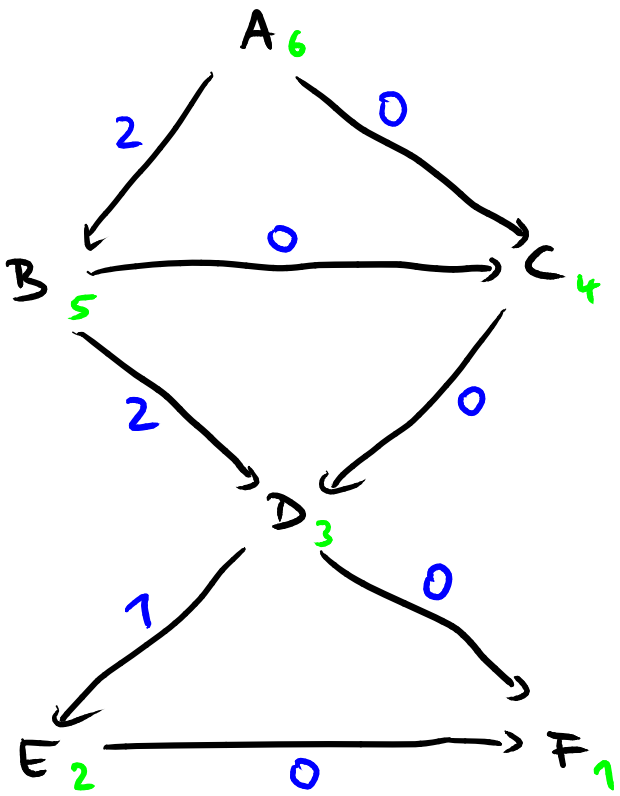
For each node in reverse topological order:

- **If n is leaf node:**

  $NumPaths(n) = 1$

- **Else:**
  - $NumPaths(n) = 0$
  - For each edge $n \rightarrow m$:
    - $Val(n \rightarrow m) = NumPaths(n)$
    - $NumPaths(n) \mathrel{+}= NumPaths(m)$

Example: NumPaths & Edge Values



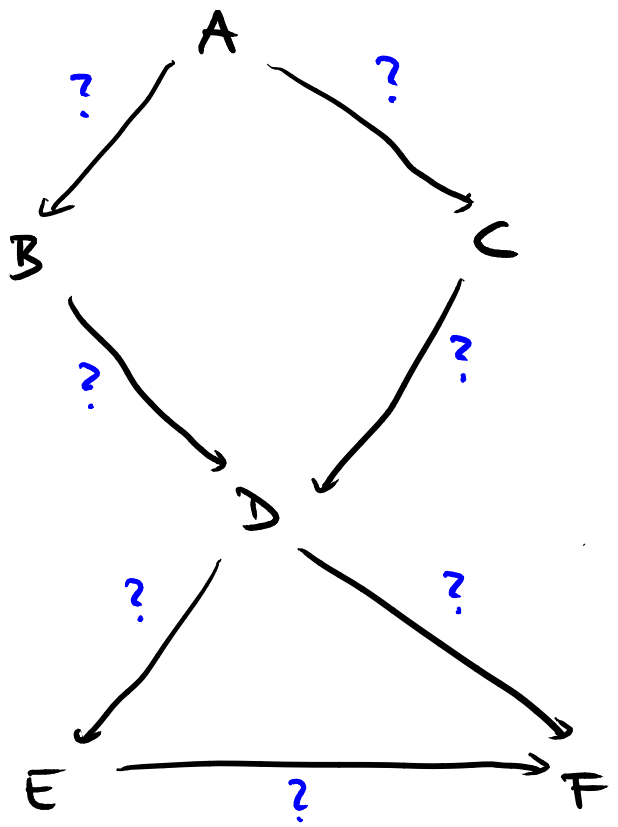
Values of edges

Reverse topological order:

Successor of n visited before n

Node n	NumPaths(n)
F	1
E	~~0~~ 1
D	~~0~~ ~~1~~ 2
C	~~0~~ 2
B	~~0~~ ~~2~~ 4
A	~~0~~ ~~2~~ 6

Quiz: Values for Edges



Values for edges

n	NumPaths(n)
0	0

Hint: $\sum_n \text{NumPaths}(n) = 12$

Algorithm: Overview

- **Step 1: Assign integers to edges**
  - Goal: Sum along a path yields unique number for path
  - Enough to achieve ”precise” goal
- **Step 2: Assign increment operations to edges**
  - Goal: Minimize additions along edges
  - Instrument subset of all edges
  - Assumes to know/estimate how frequent edges are executed

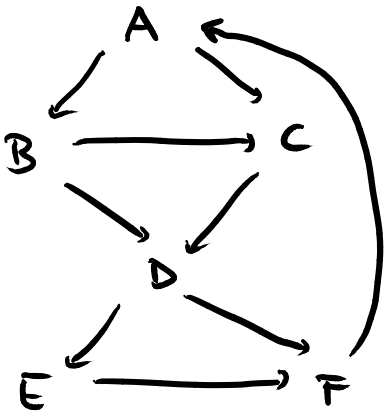
Spanning Tree

- **Given: Graph G**
- **Spanning tree T:**
  Undirected subgraph of G that is a tree and that contains all nodes of G
- **Chord edges: Edges in G but not in T**

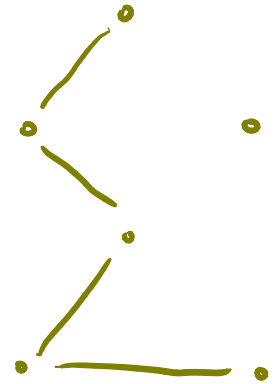
Example: Spanning Tree

Which of these is a spanning tree of G?

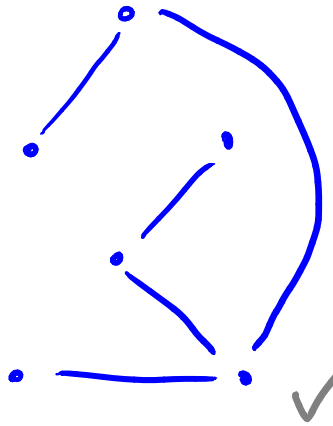
G:



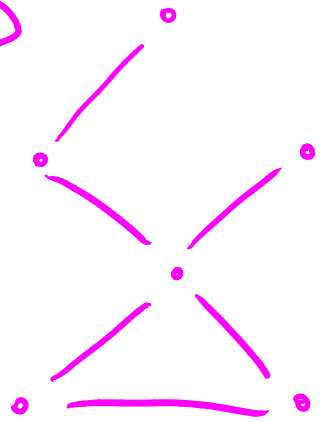
1.



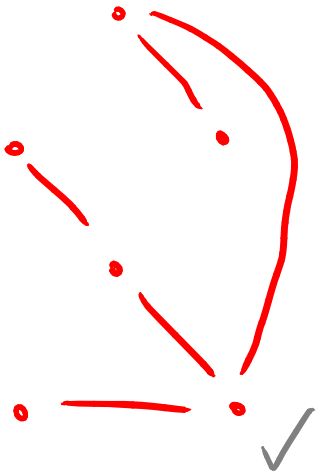
2.



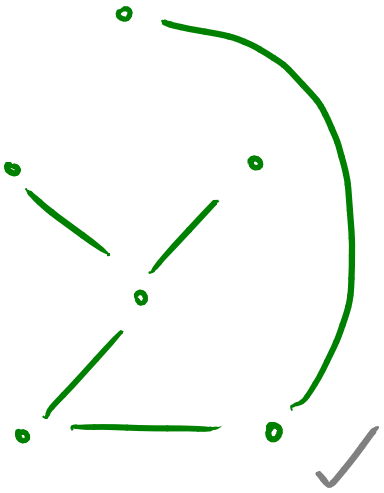
3.



4.



5.

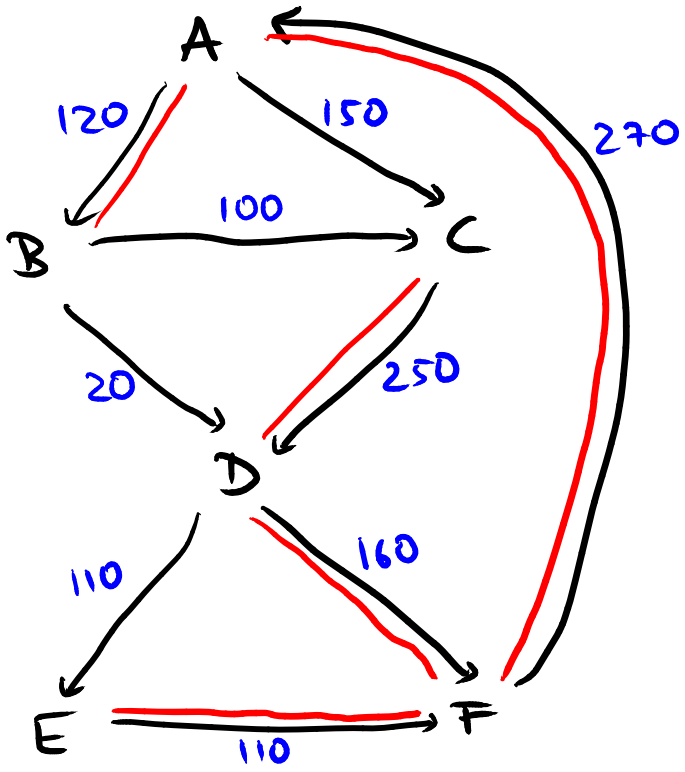


Increments for Edges

Goal: Increment sum at subset of edges

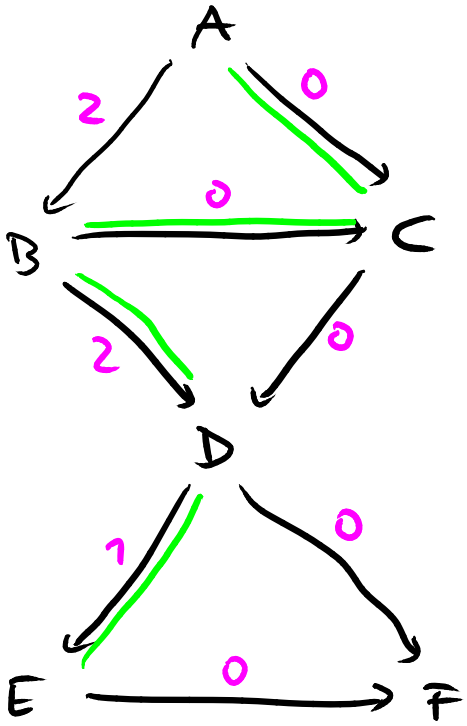
- **Choose spanning tree with maximum edge cost**
  - Cost of individual edges is assumed to be known
- **Compute increments at the chords of the spanning tree**
  - Based on existing event counting algorithm

Example: Increments for Edges



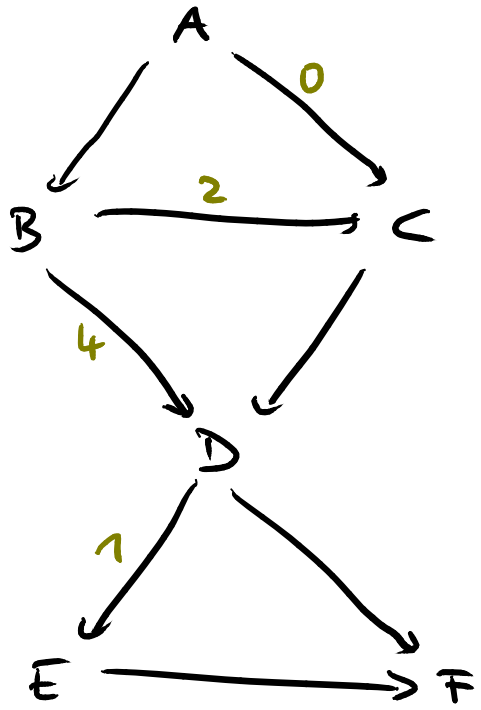
edge cost

most expensive spanning tree



non-minimal increments

chord edges



minimal increments = path encoding

Instrumentation

- **Basic idea**
  - Initialize sum at entry: `r=0`
  - Increment at edges: `r+=..`
  - At exit, increment counter for path:

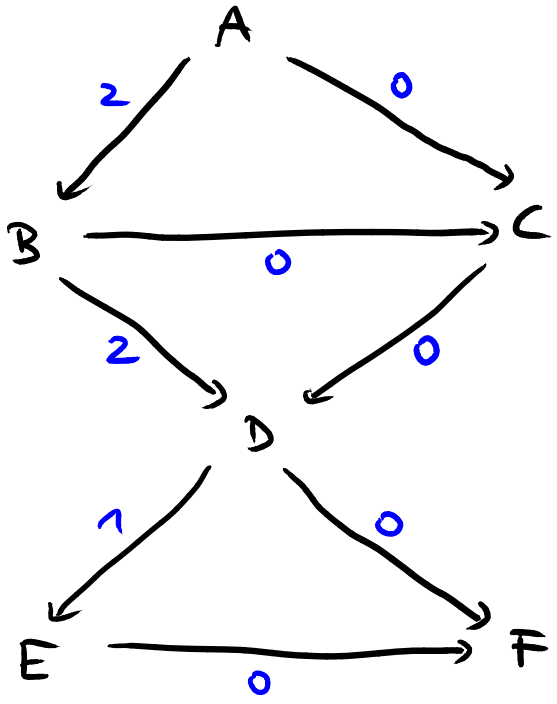
    `count[r]++`
- **Optimization**
  - Initialize with incremented value, if first chord on path: `r=..`
  - Increment sum and counter for path, if last chord on path: `count[r+..]++`

Regenerating the Path

Knowing the sum r, how to determine the path?

- Use edge values from step 1 (”non-minimal increments”)
- Start at entry with $R = r$
- At branches, use edge with largest value v that is smaller than or equal to R and set $R -= v$

Example: Regenerate the Path



r = 4 :

A B D F

R = ~~A~~ ~~X~~ O

r = 1 :

A C D E F

R = ~~A~~ O