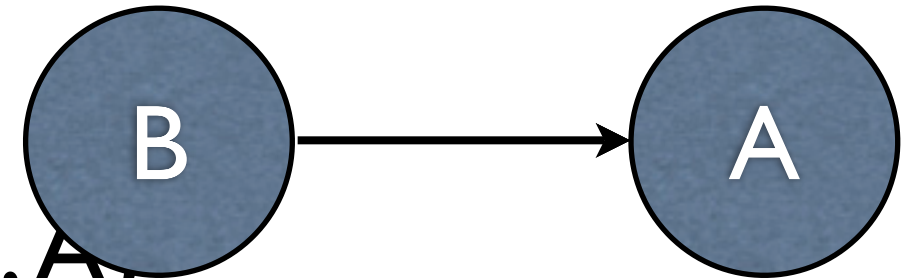
Colocation Explained

Heartbeat 2.1.2-4 Onwards

abeekhof@suse.de

Terminology



- Collocate(B, A)
- <rsc_colocation from=B to=A/>
- Decide where to put A, then put B there too
- Include B's preferences when deciding where to put A
- If A cannot run anywhere, B can't run either
- If B cannot run anywhere, A will be unaffected

Adding Scores

- number > INFINITY =   INFINITY
- number < -INFINITY = - INFINITY
- number + INFINITY =   INFINITY
- number - INFINITY = - INFINITY
- INFINITY - INFINITY = - INFINITY

- INFINITY ::= 1,000,000

Simple Example

Setup

- resource(A, priority=5)
- resource(B, priority=50)
- location(A, node1, 100)
- location(A, node2, 10)
- location(B, node2, 1000)
- collocate(B, A)

Simple Example

What Happens

- Start at highest priority resource (**B**)
- Defer and process **A** instead (collocation rule)
- Incorporate **B**'s preferences
  - A.node1.score += B.node1.score (100)
  - A.node2.score += B.node2.score (1010)
- Choose a node (**node2**)

Simple Example

Actually I Lied

- Incorporate **B**'s preferences
  - A.node[x].score += ***factor*** * B.node[x].score
- What is ***factor***?
  - *factor ::= constraint.score / INFINITY*
- *For most people it will be 1 or -1*
- *So really its: colocate(B, A, **score**)*

Choosing a Node for **B**

Simple Example

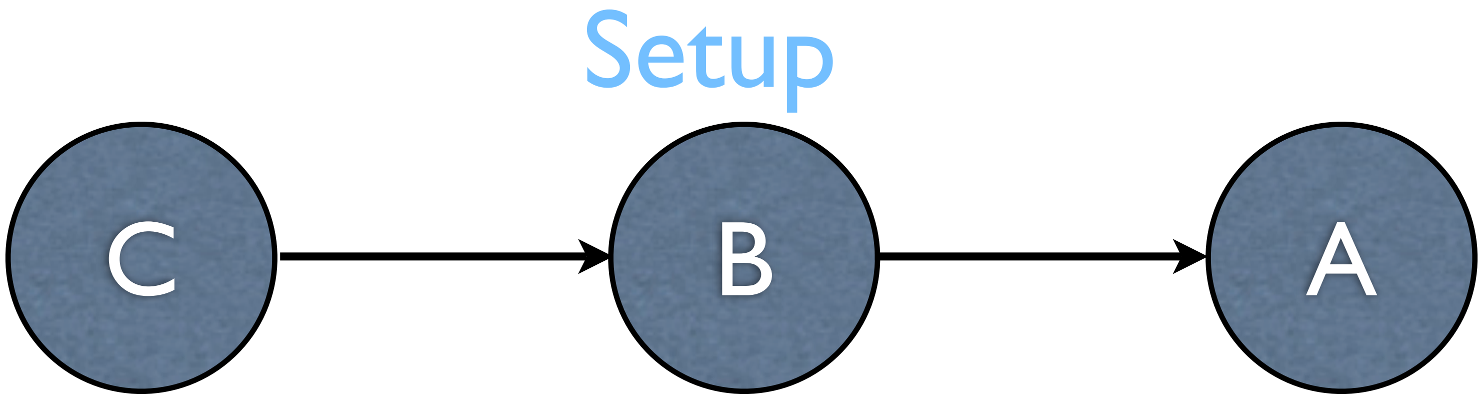
- Process collocation constraint
  - Matching node: node.score = INFINITY
  - Everything else: node.score = -INFINITY
- Scores do **not** include **A**'s preferences
- Final scores for **B**
  - node1 = -INFINITY
  - node2 = INFINITY

Choosing a Node for **B**

Suggested Colocation

- When the collocation score != INFINITY
  - Matching node: node.score += collocation.score
  - Everything else: unchanged
- Scores do **not** include **A**'s preferences
- Final scores for **B** (collocation.score = 500)
  - node1 = 0
  - node2 = 1500

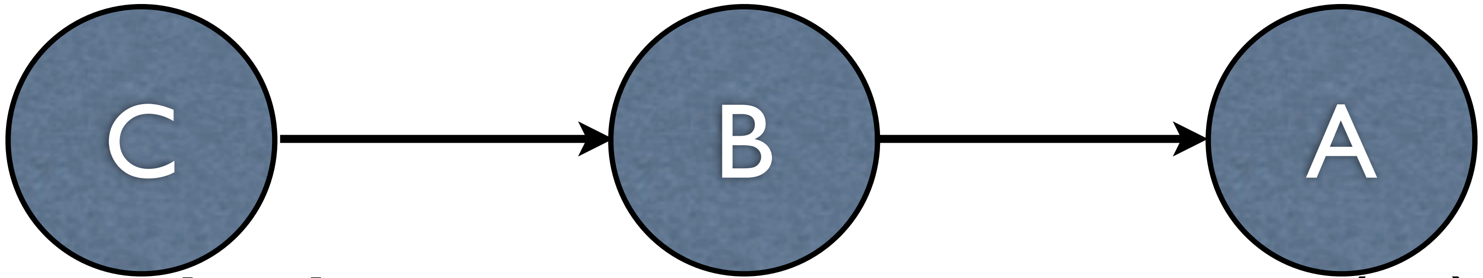
Chained Example



- resource(A, p=5)
- resource(B, p=500)
- resource(C, p=50)
- location(A, node1, 100)
- location(A, node2, 10)
- location(B, node2, 1000)
- location(C, node1,10000)
- collocate(B,A)
- collocate(C, B)

Chained Example

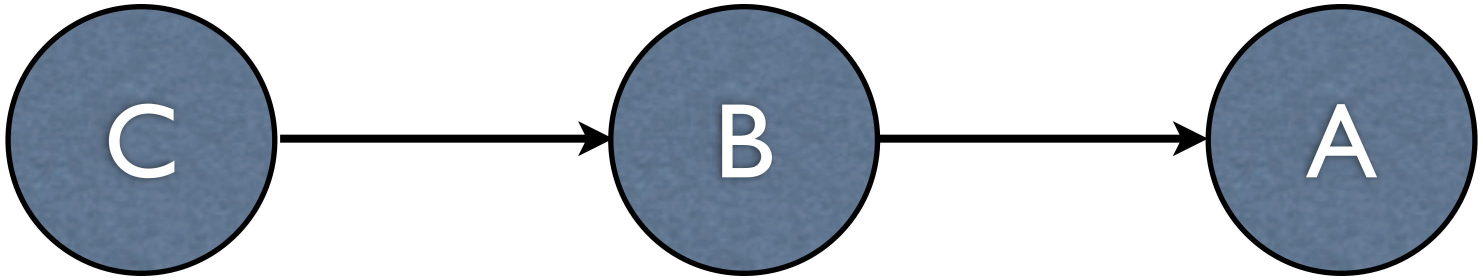
What Happens



- Start at highest priority resource (**B**)
- Defer and process **A** instead (collocation rule)
- Incorporate **B**'s preferences
    - A.node[x].score += B.node[x].score
- So far nothing is different

Chained Example

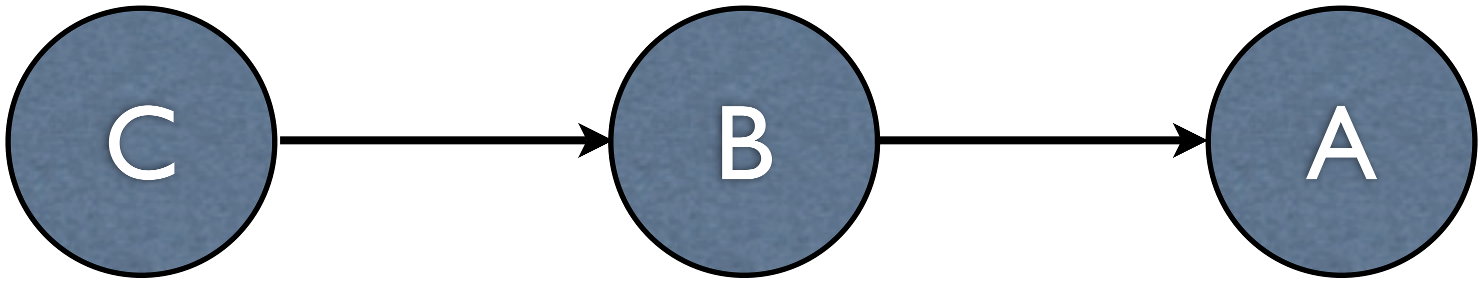
What Happens (Continued)



- Incorporate **C**'s preferences too!
  - A.node[x].score += C.node[x].score
- Final scores (when choosing a node for **A**)
  - node1 = 10100
  - node2 = 1010

Chained Example

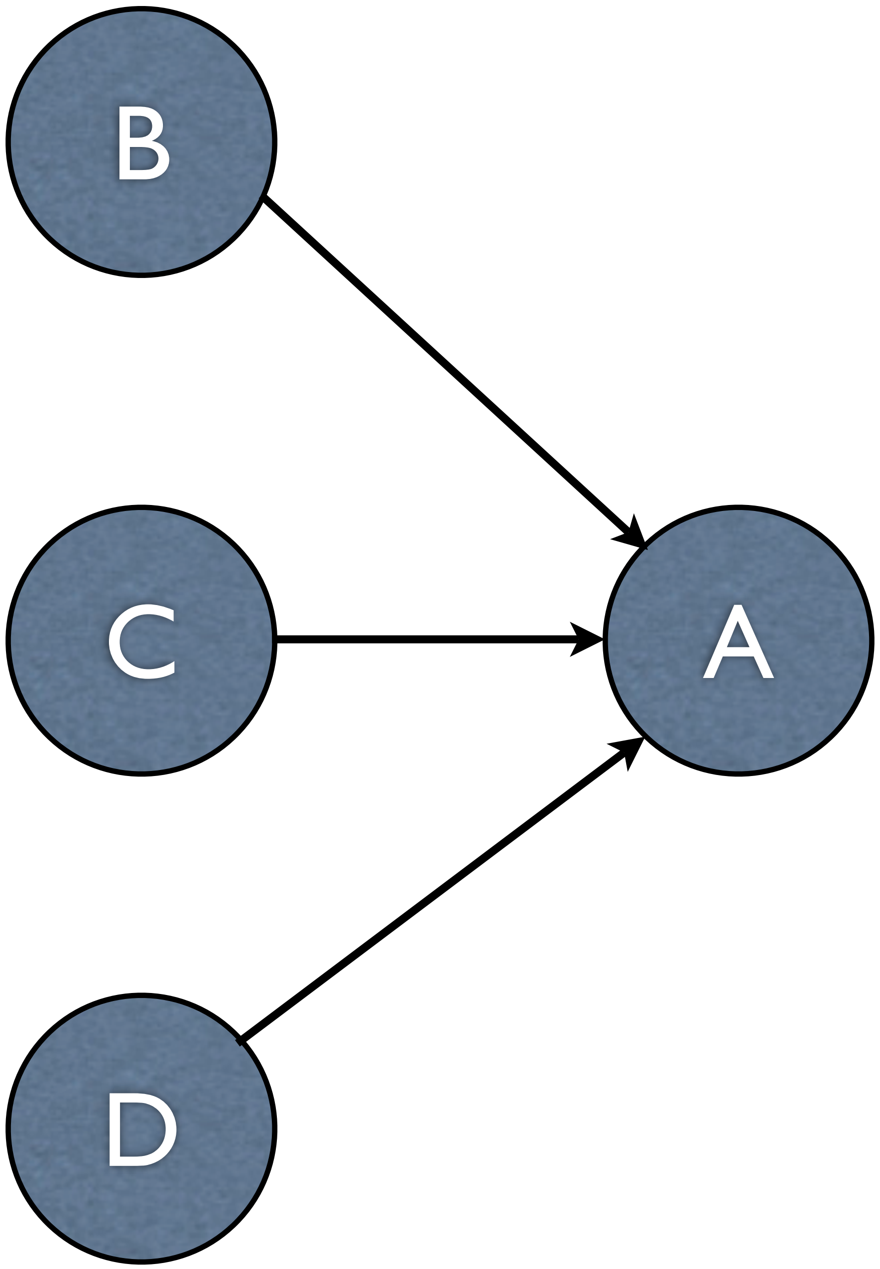
Final Scores: **B** and **C**



- Resource **B**
  - node1 = INFINITY
  - node2 = -INFINITY
- Resource **C**
  - node1 = INFINITY
  - node2 = -INFINITY

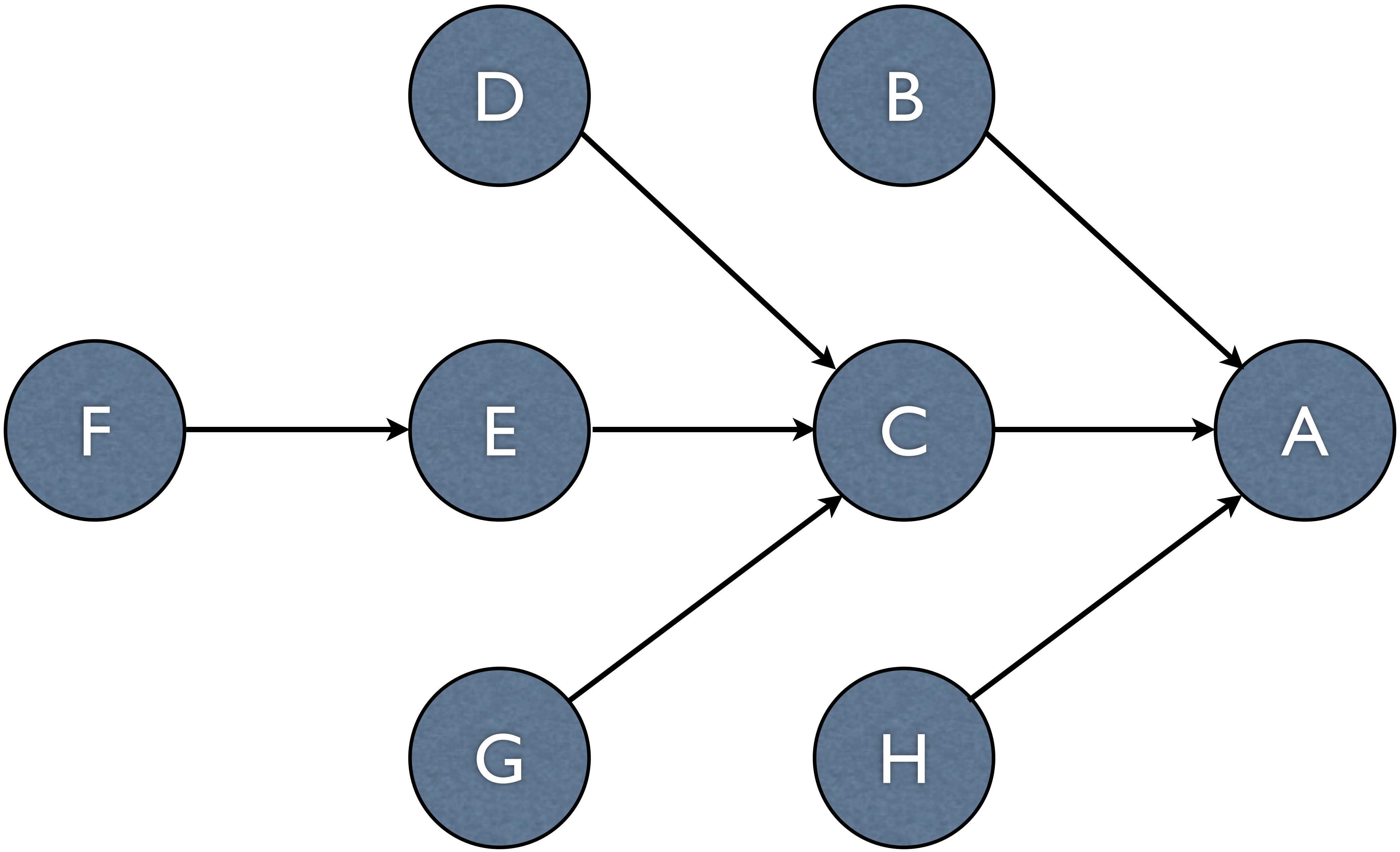
Multiple Dependencies

- Include scores from B, C <u>and</u> D when choosing a node for A
- Order is defined by priority of dependent resources (or name if priority is equal)
- In this example:
  - B.priority > C.priority
  - C.priority > D.priority



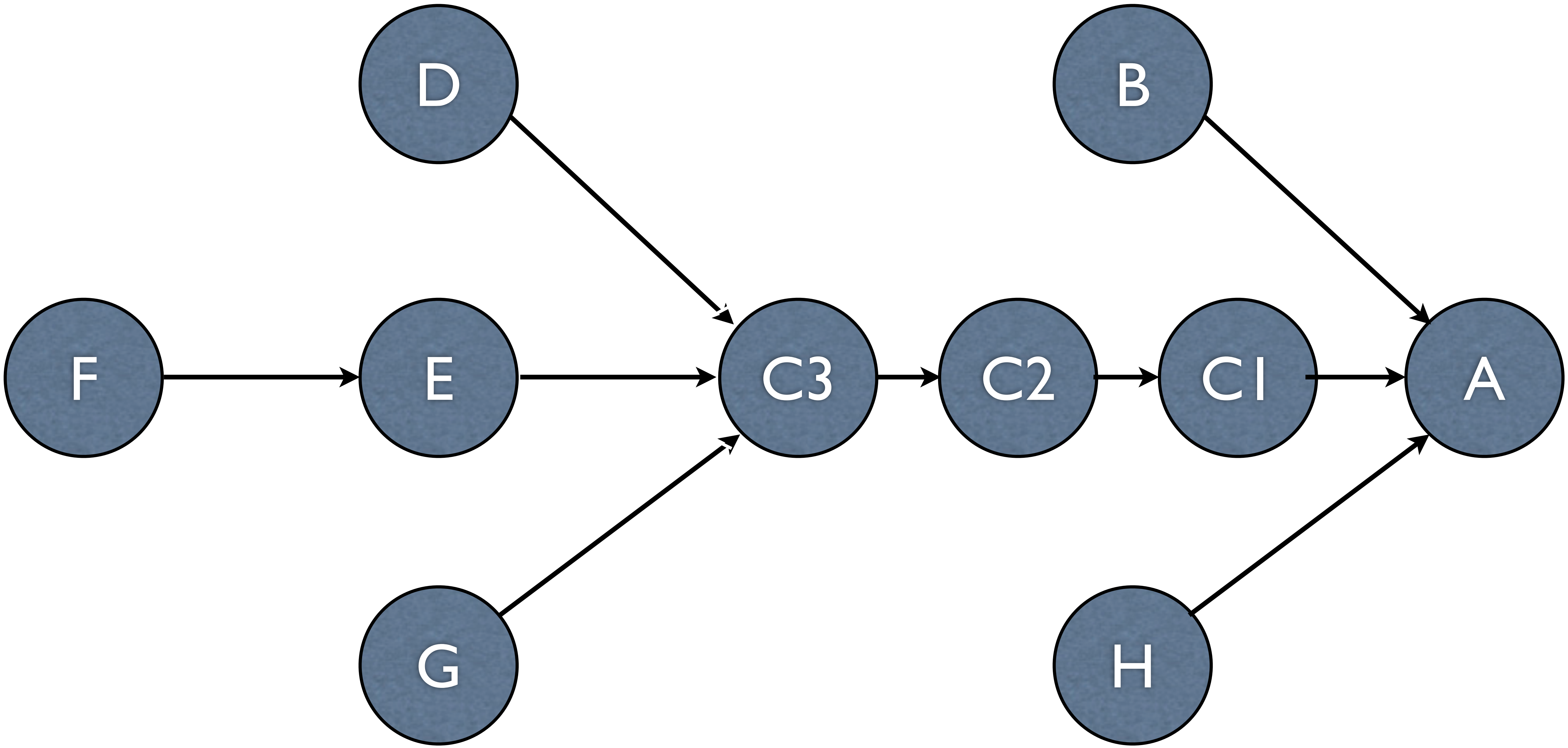
Dependancy Tree

Order in Which Preferences are Applied (A-H)



More Complex

C is a Group

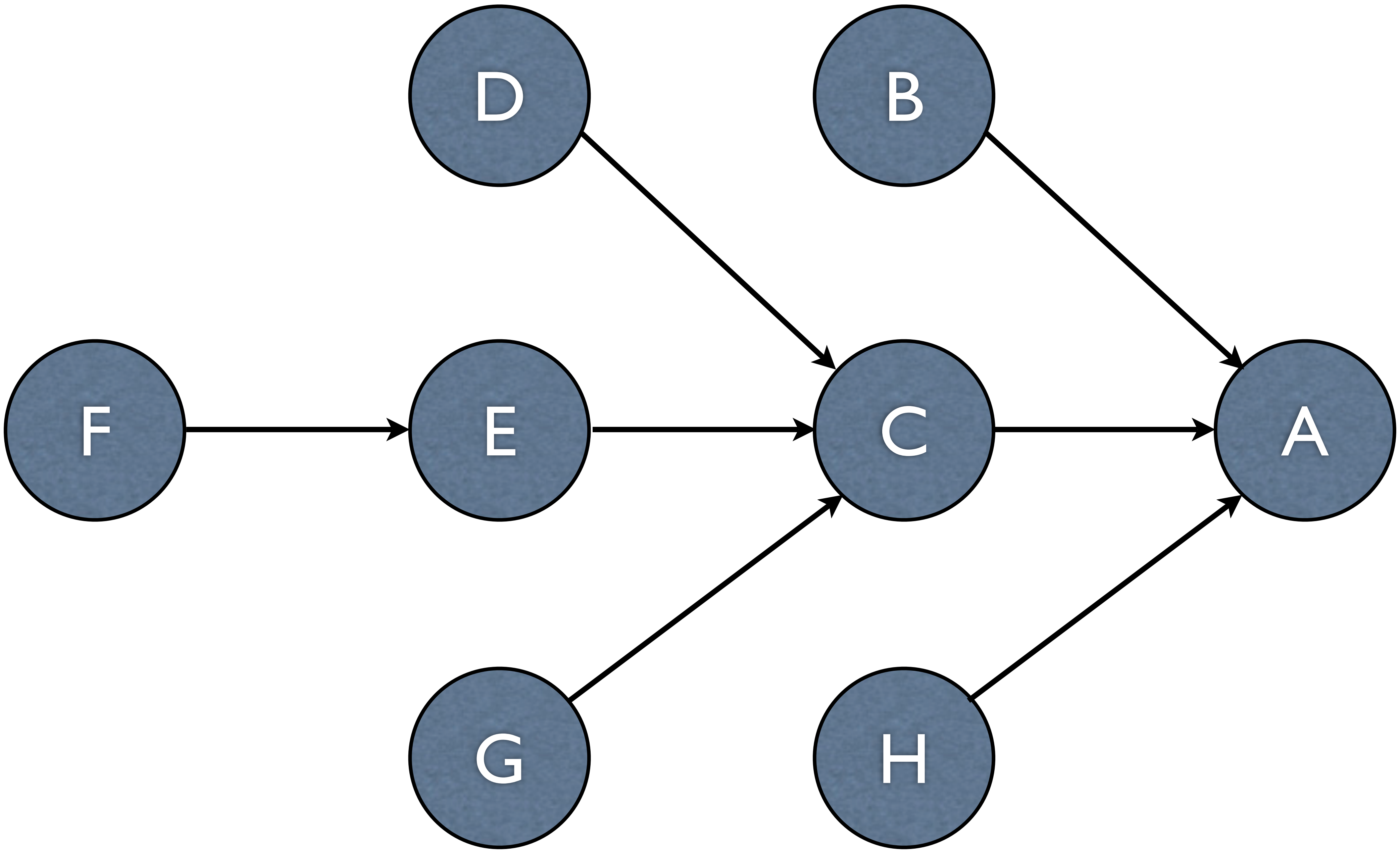


Getting Smart

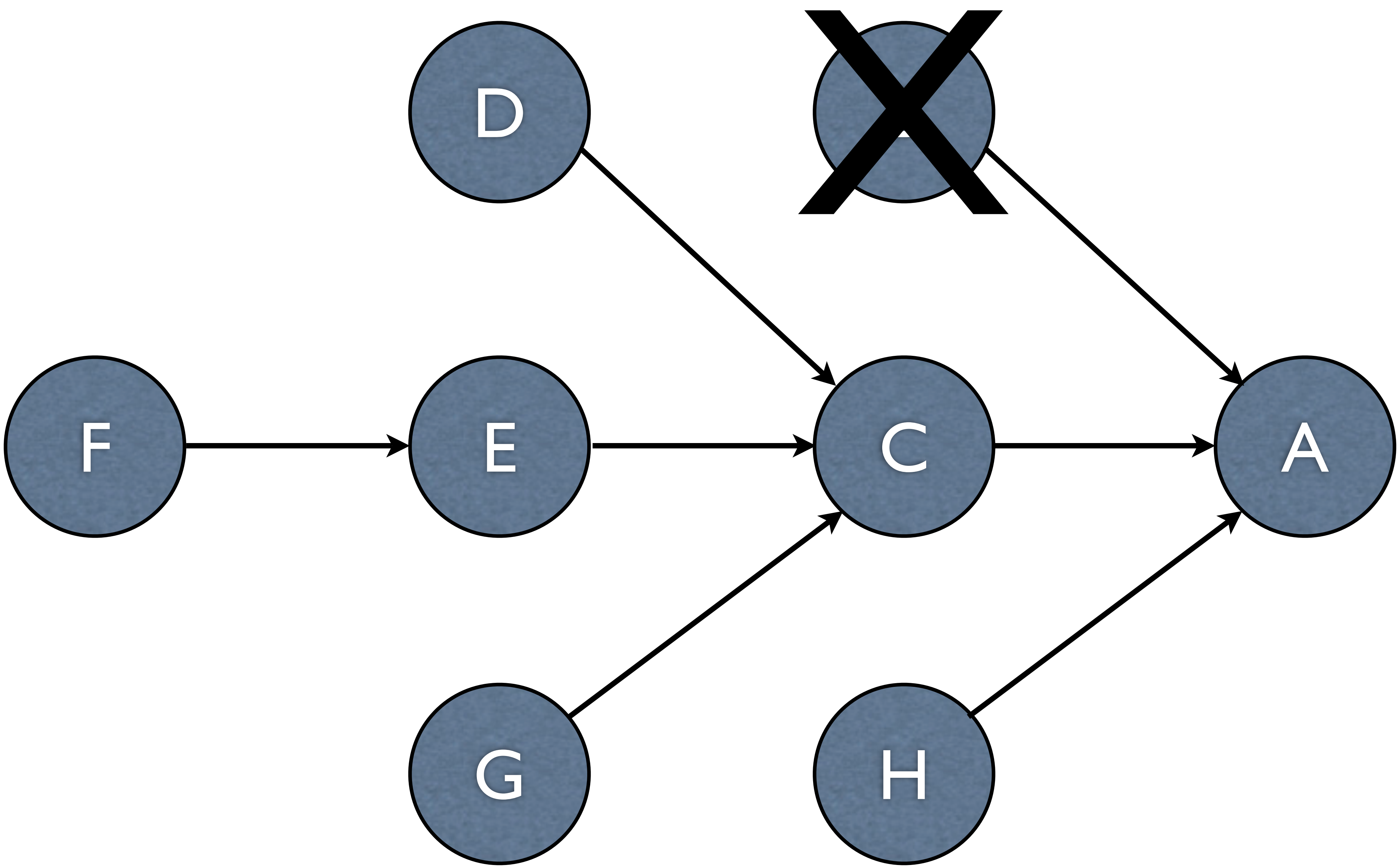
When not Everything can Run

- If applying a resource's preference, means that all nodes would be unavailable...
  - Undo the current resource's preference
  - Skip any resources that need to be collocated with the current resource
  - Process the next peer

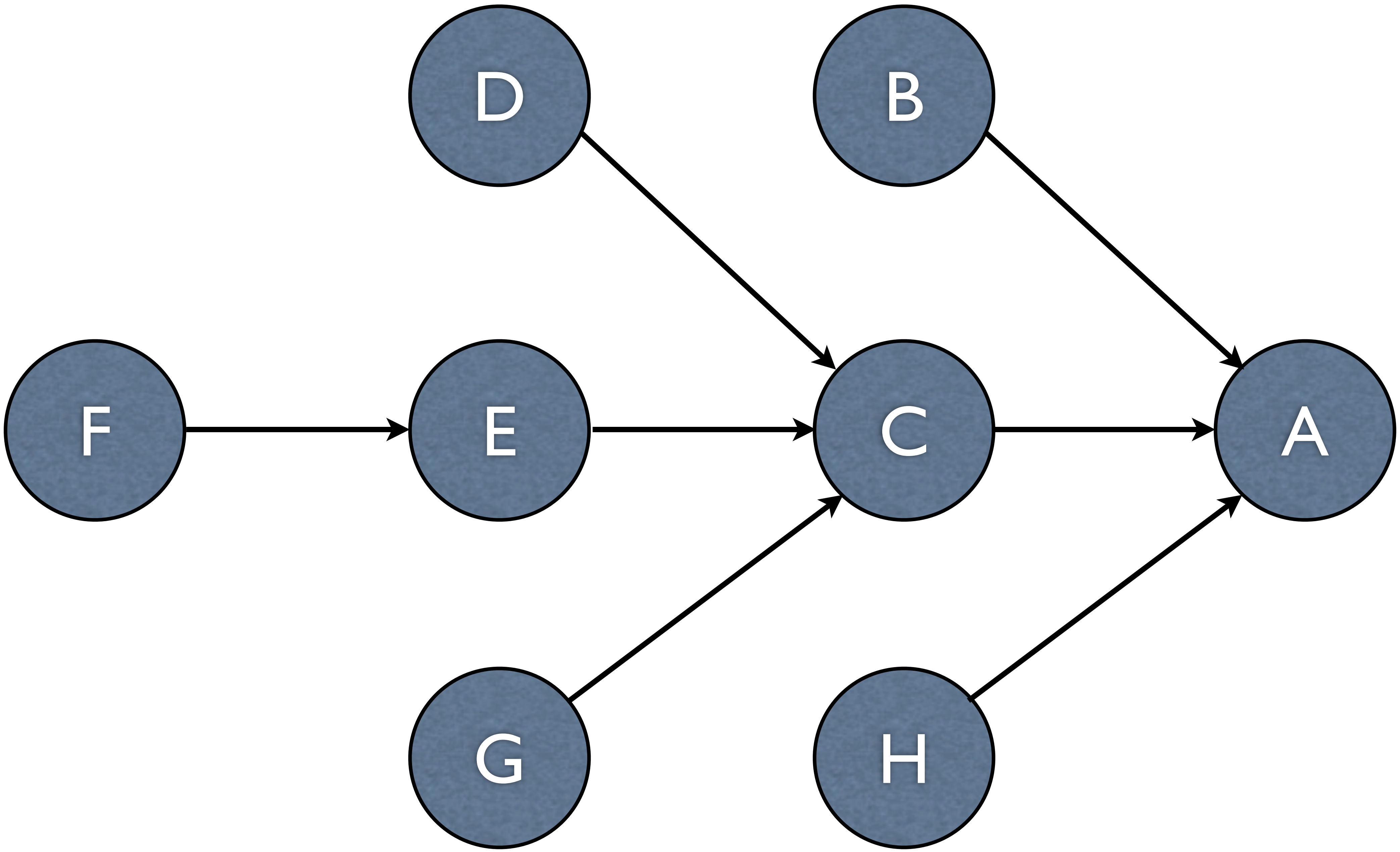
Un-runnable: B



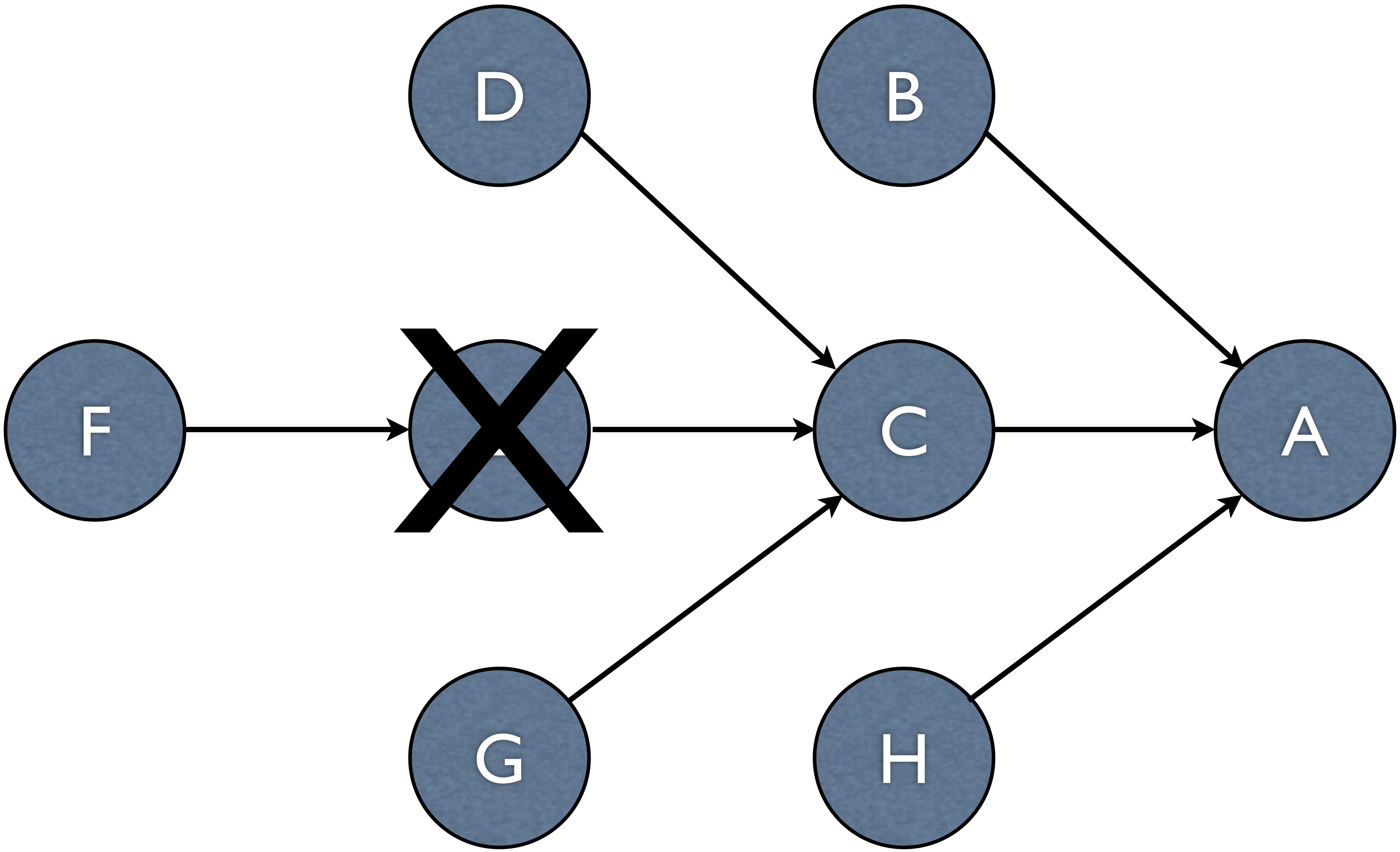
Un-runnable: B



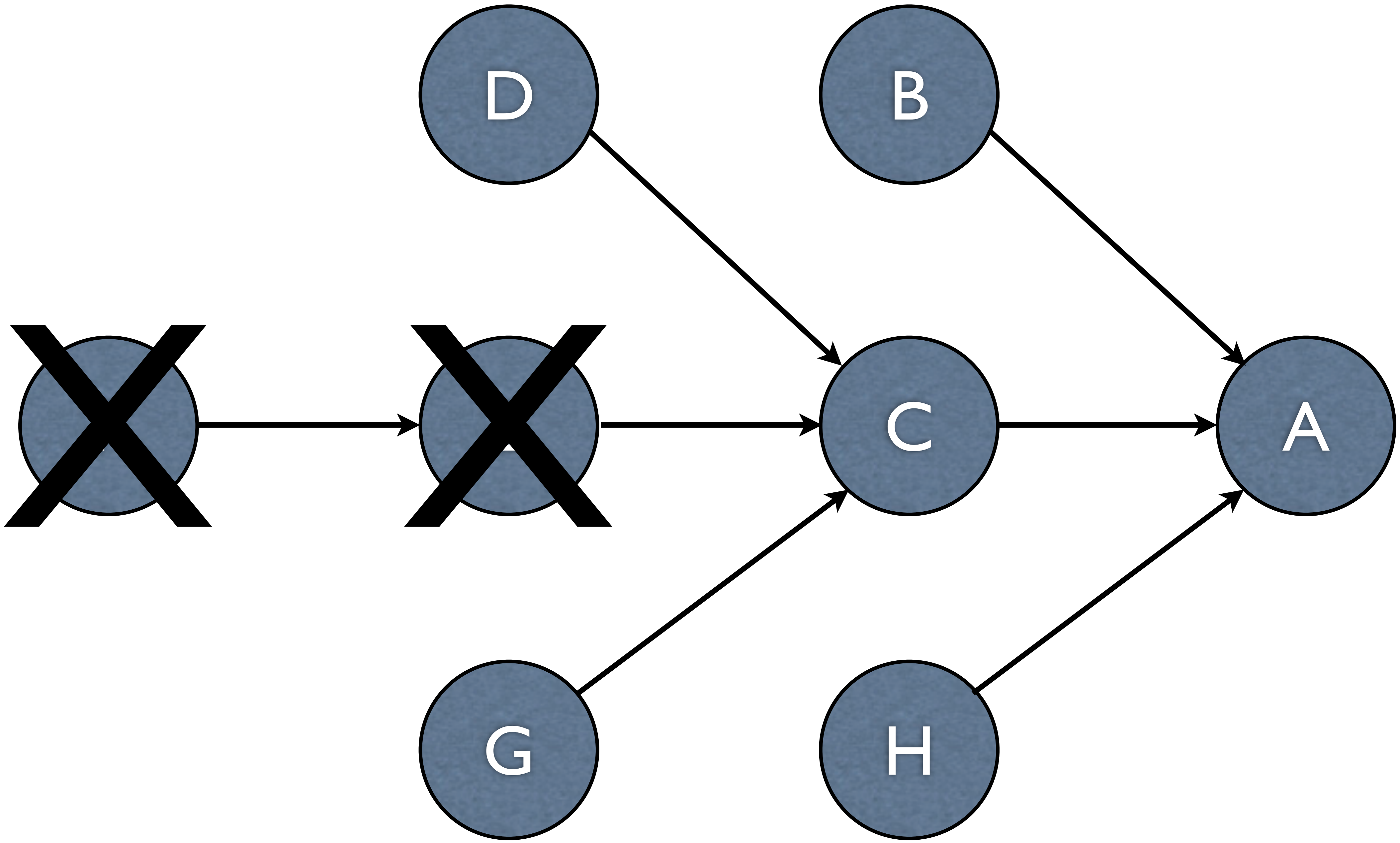
Un-runnable: E



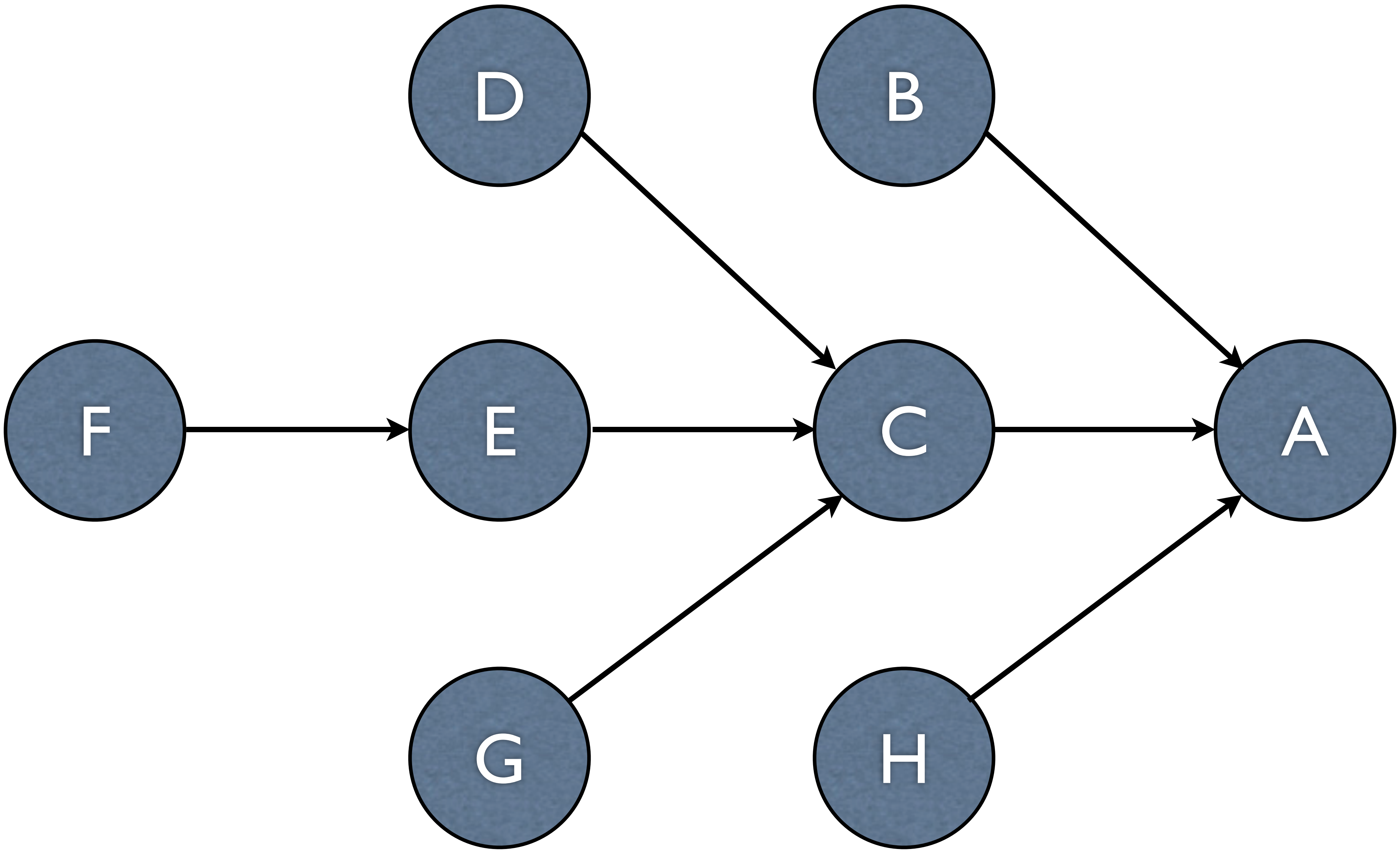
Un-runnable: E



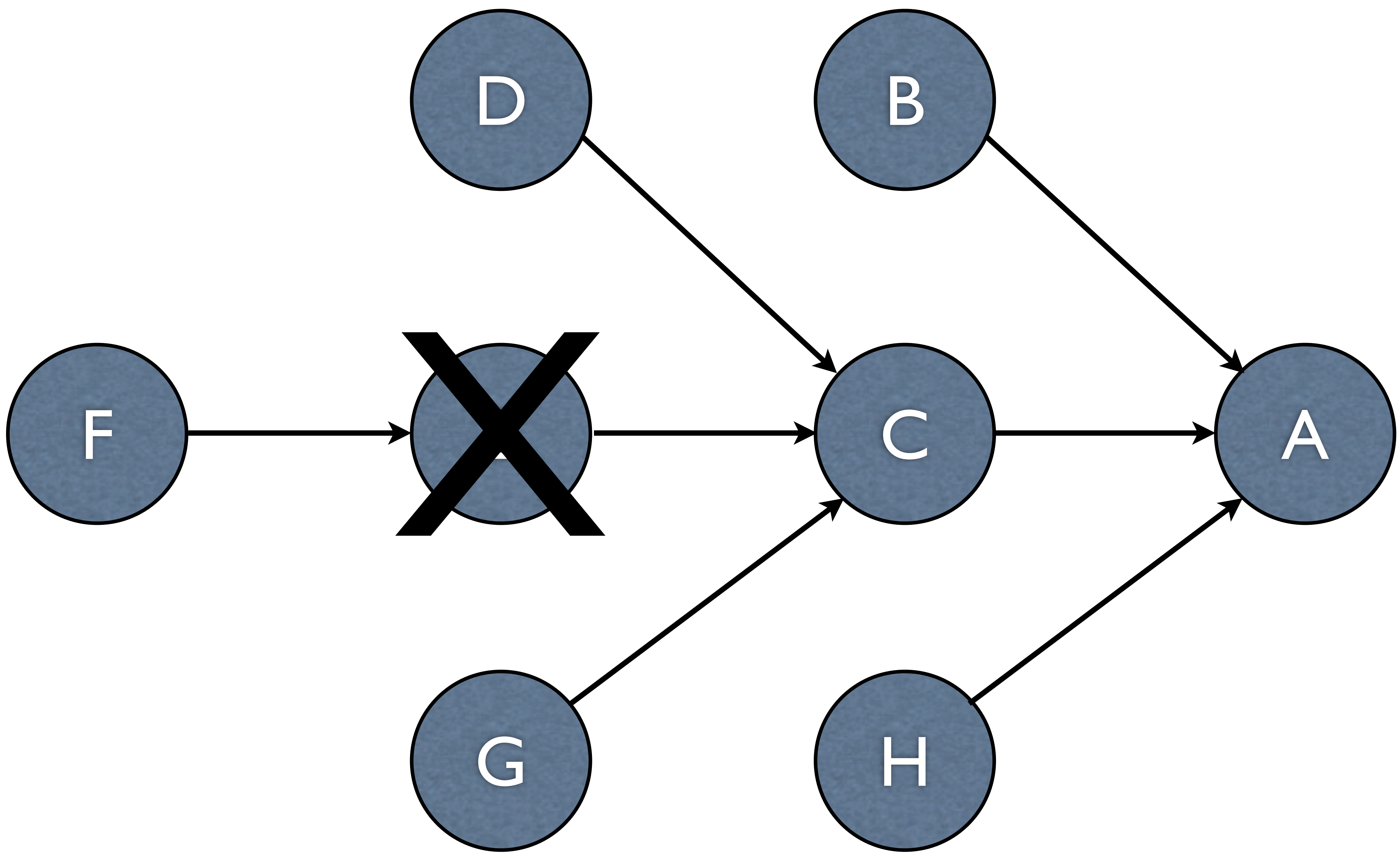
Un-runnable: E



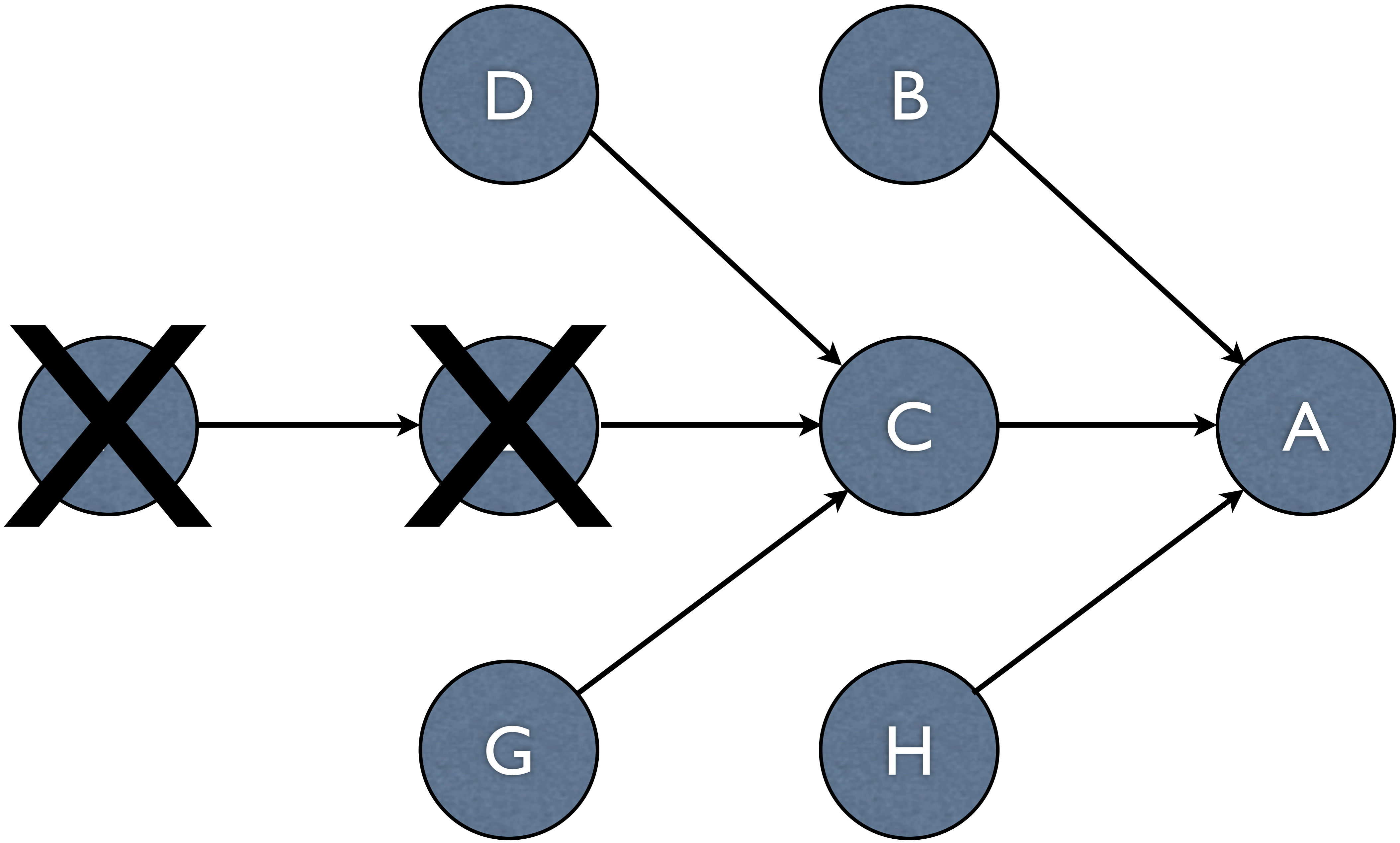
Un-runnable: C



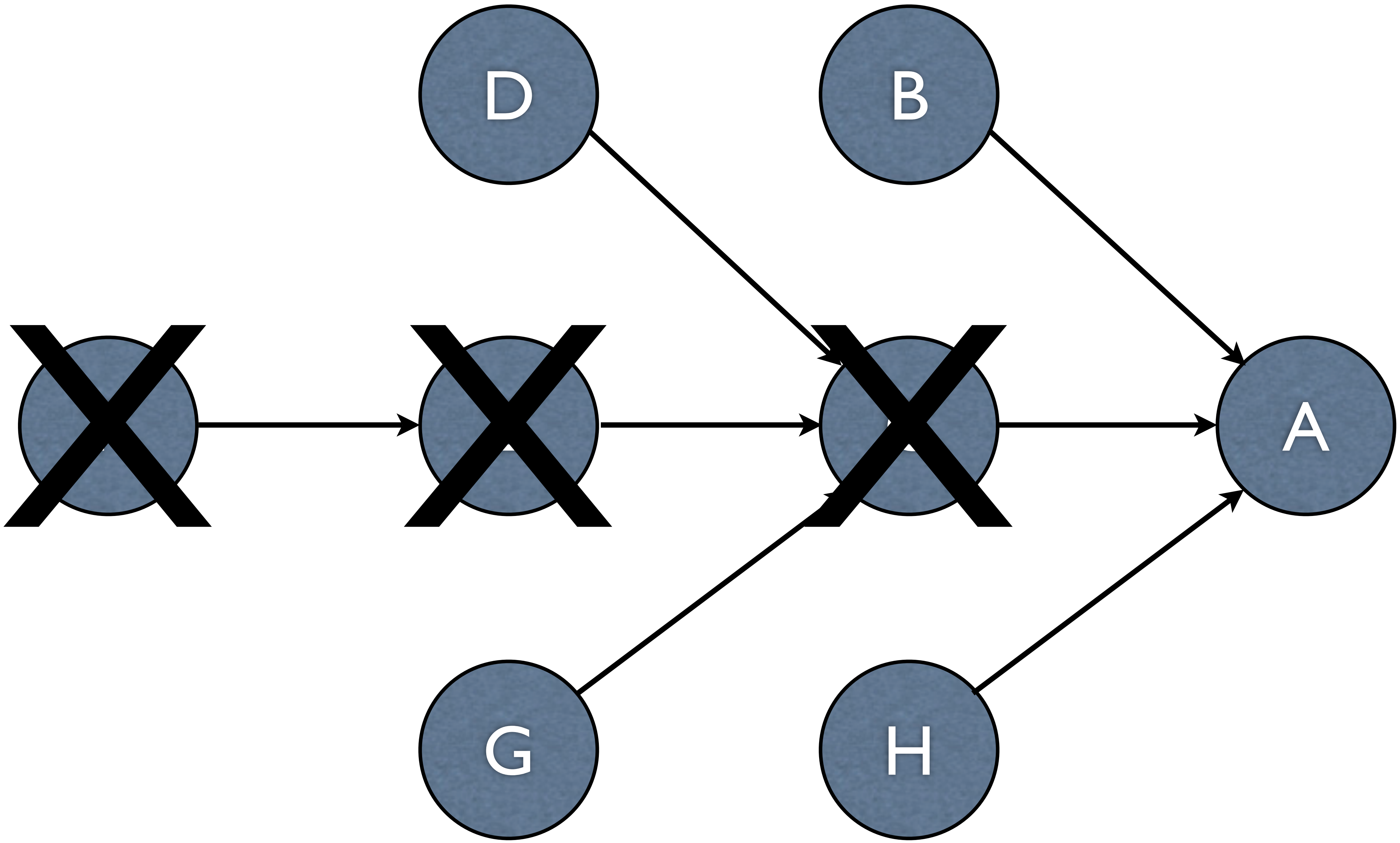
Un-runnable: C



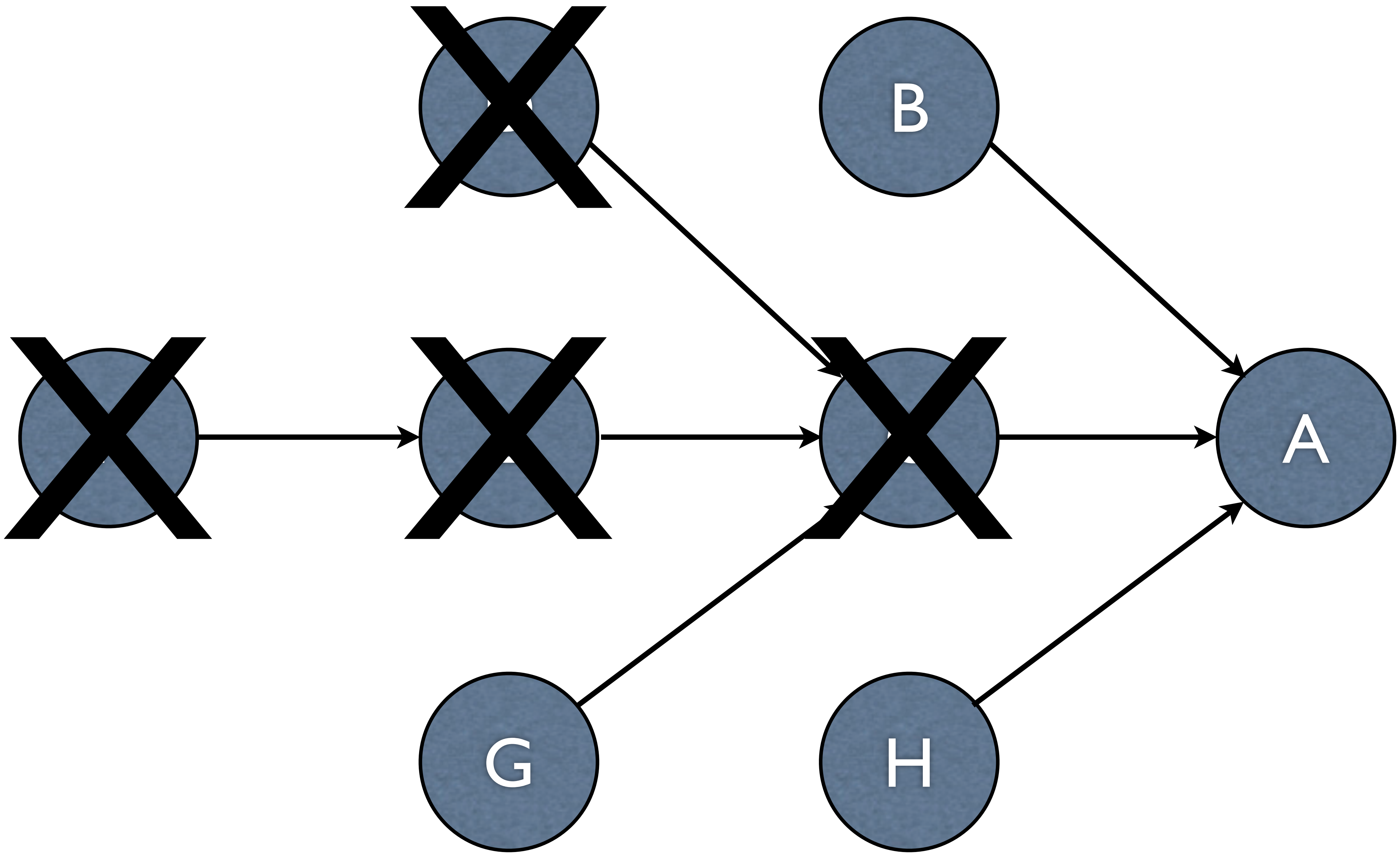
Un-runnable: C



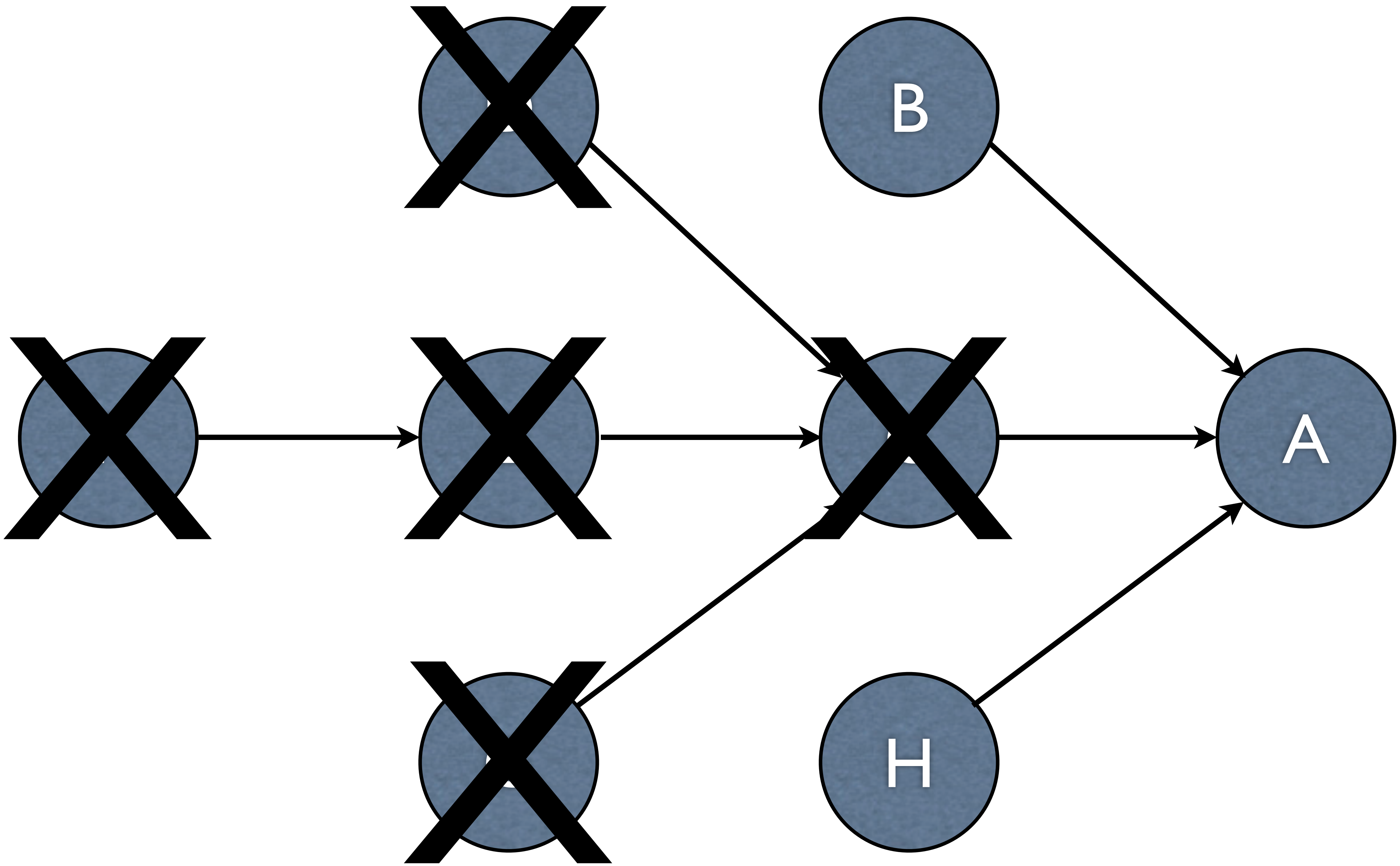
Un-runnable: C



Un-runnable: C



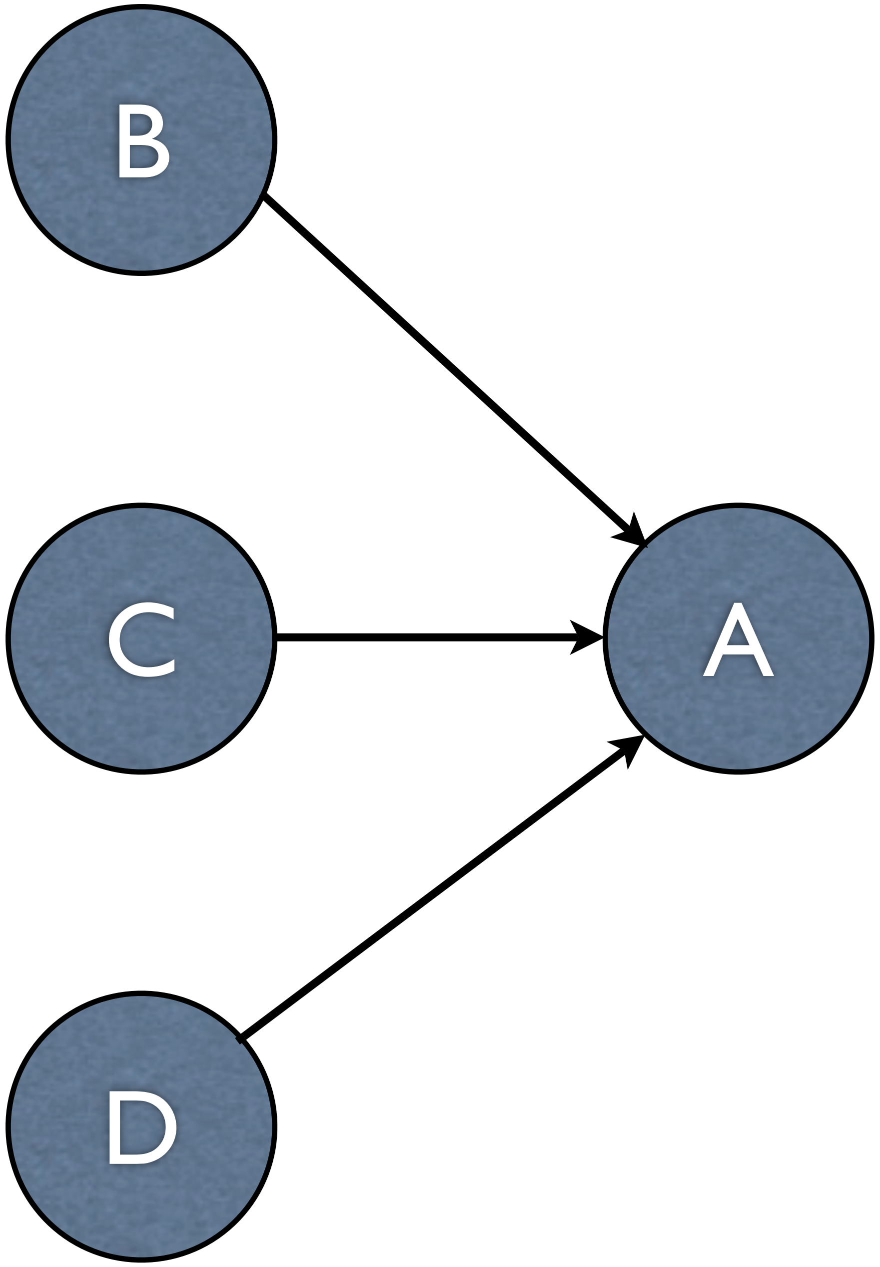
Un-runnable: C



Un-runnable

Worked Example

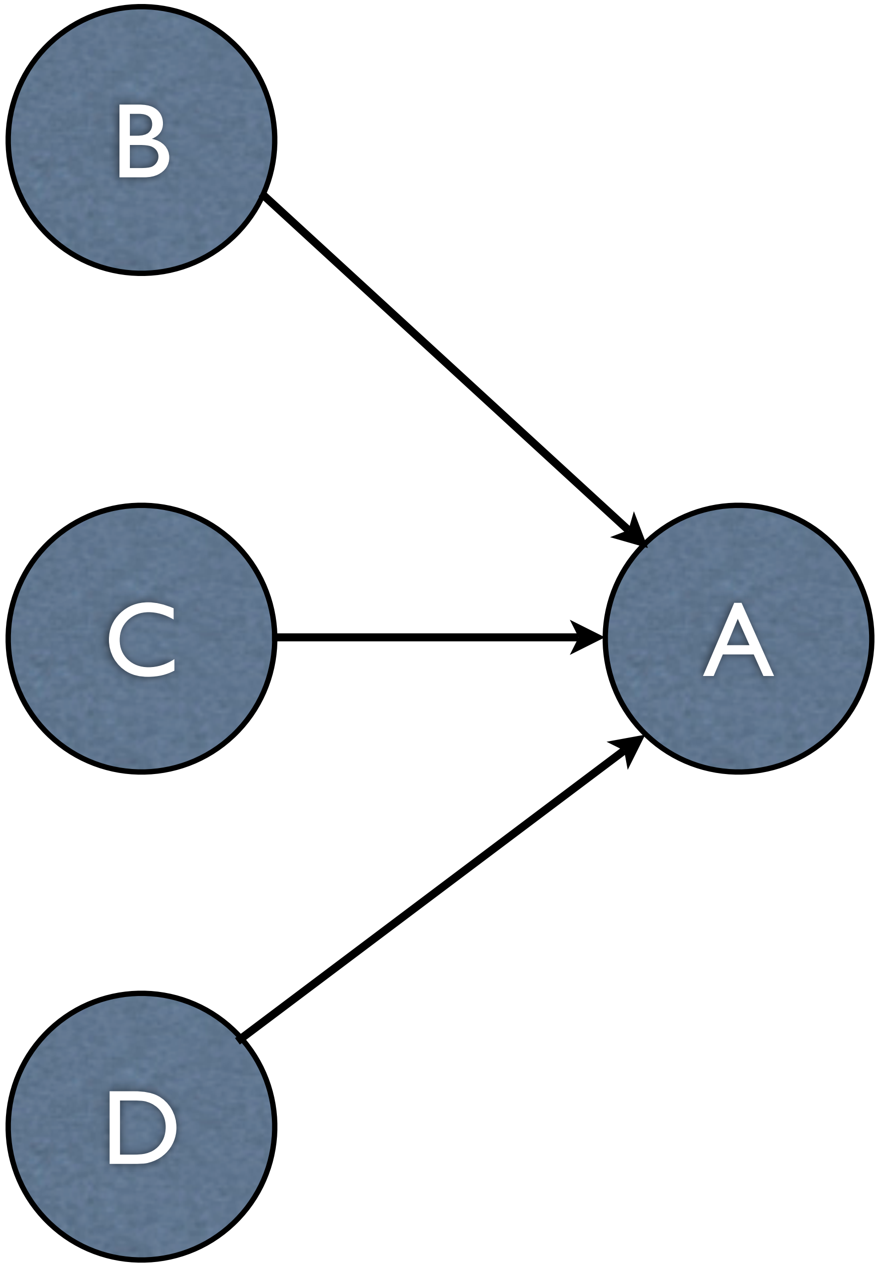
Rsc	Node	Score
A	node1	50
A	node2	5
B	node1	1
B	node2	10
C	node1	-INFINITY
C	node2	-INFINITY
D	node1	100



Un-runnable

Worked Example (continued)

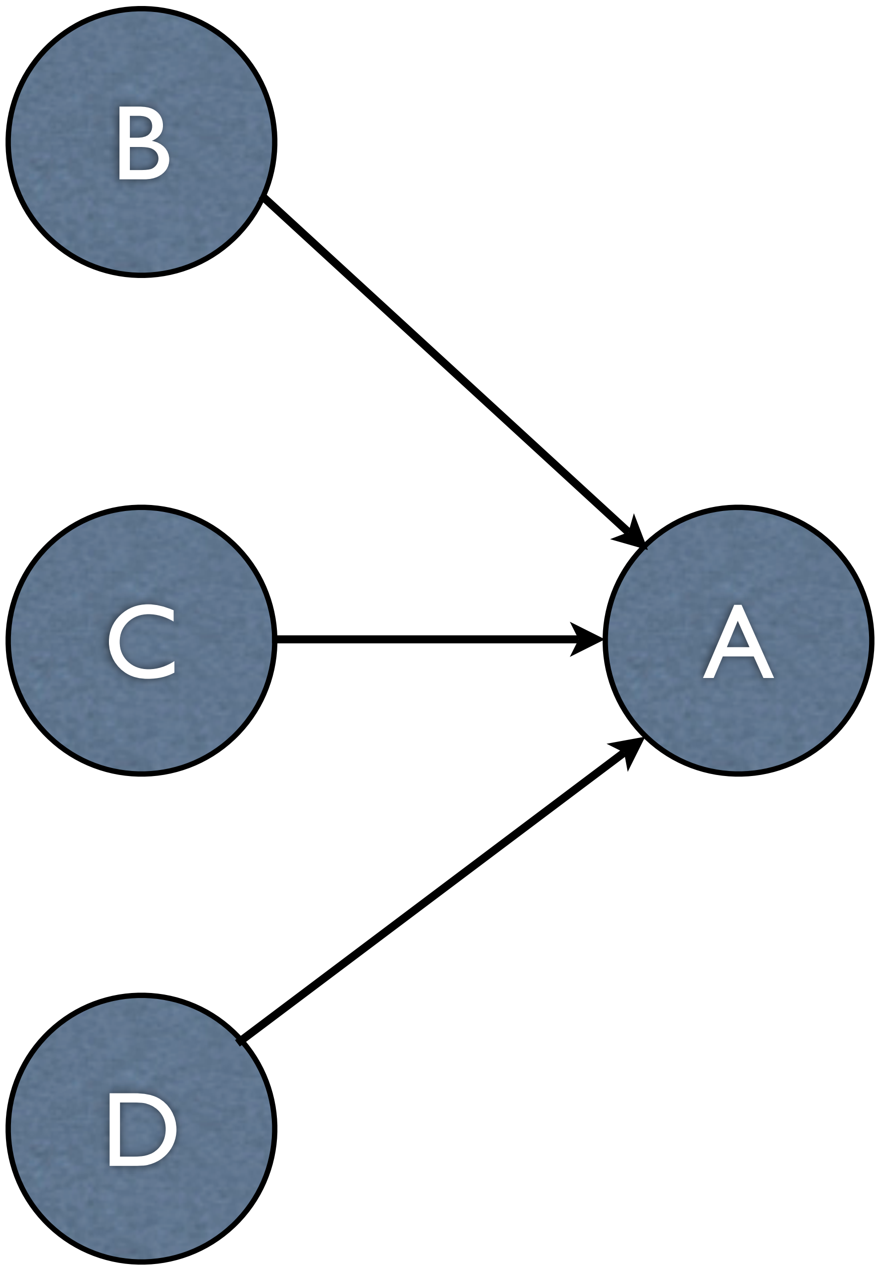
- Consider **B**
  - A.node1.score = 50 + 1
  - A.node2.score = 5 + 10



Un-runnable

Worked Example (continued)

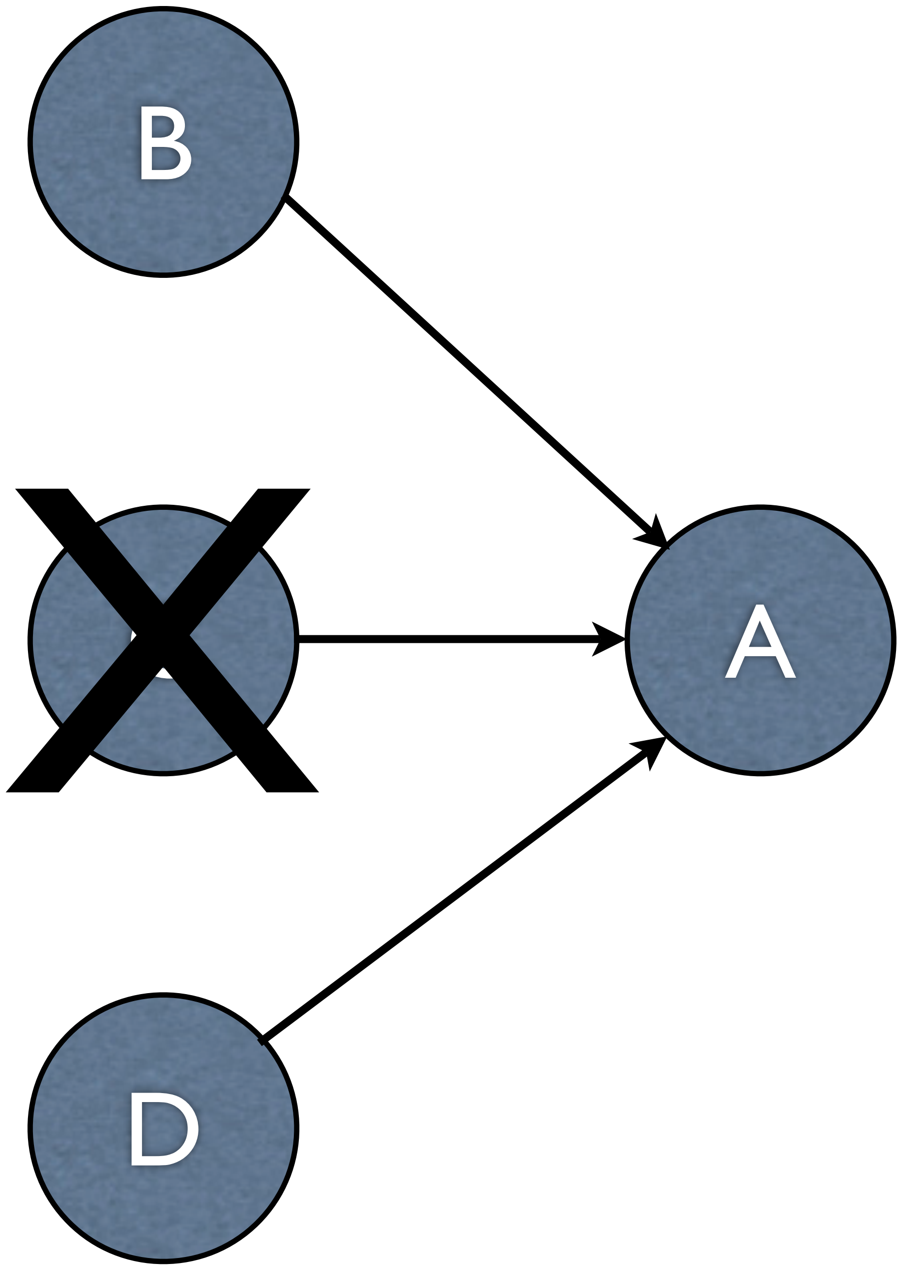
- Consider **C**
  - A.node1.score = 51 -INFINITY
  - A.node2.score = 15 -INFINITY
- Rollback Scores
  - A.node1.score = 51
  - A.node2.score = 15



Un-runnable

Worked Example (continued)

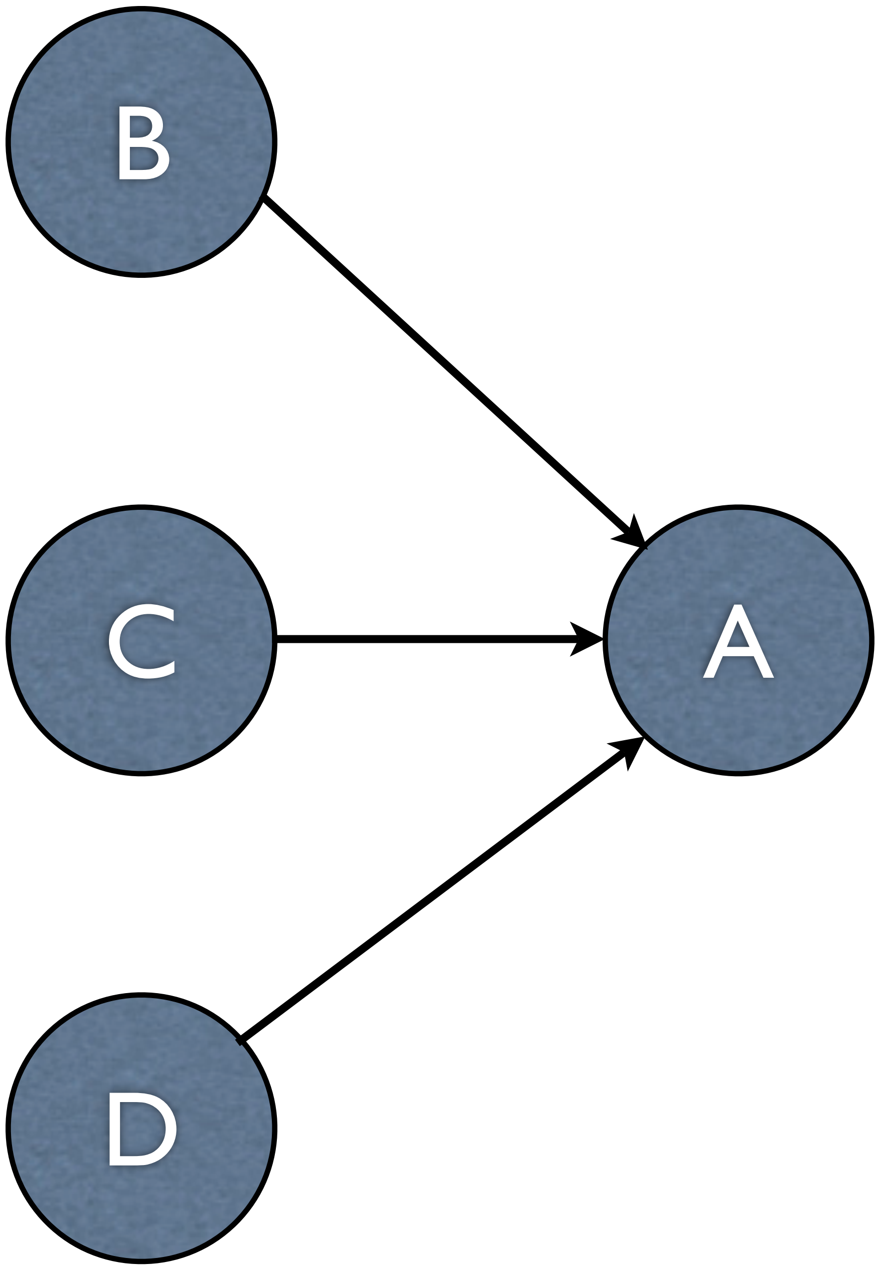
- Consider **C**
  - A.node1.score = 51 -INFINITY
  - A.node2.score = 15 -INFINITY
- Rollback Scores
  - A.node1.score = 51
  - A.node2.score = 15



Un-runnable

Worked Example (continued)

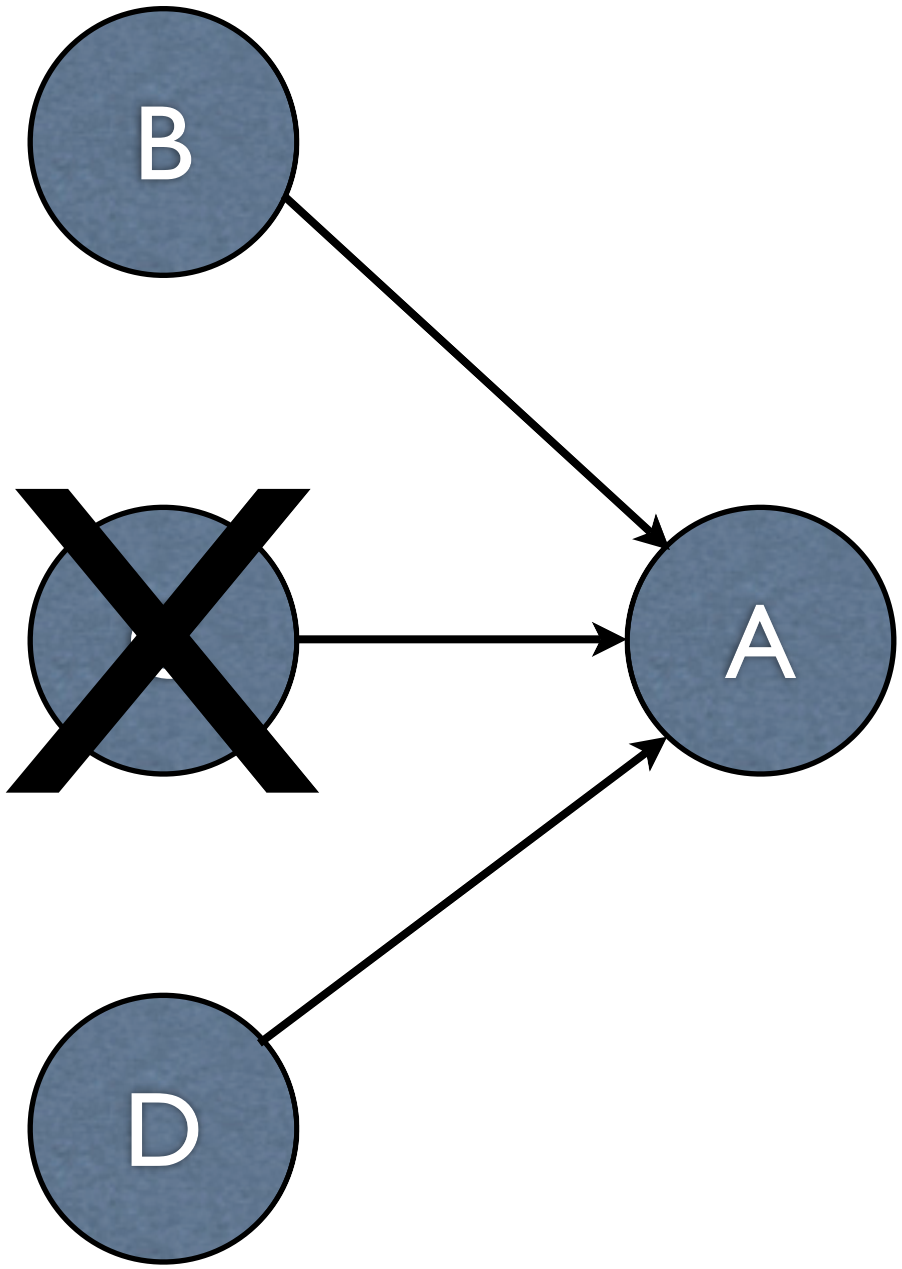
- Consider **D**
  - A.node1.score = 51 + 100
  - A.node2.score = 15 + 1000
- Final Scores
  - A.node1.score = 151
  - A.node2.score = 1015
- Choose **node2**



Un-runnable

Worked Example (continued)

- Consider **D**
  - A.node1.score = 51 + 100
  - A.node2.score = 15 + 1000
- Final Scores
  - A.node1.score = 151
  - A.node2.score = 1015
- Choose **node2**



Colocation by Role

Master/Slave - Summary

- A resource that needs to run on the master can force the master to move (rather than not be allowed to run anywhere)
- A resource that can’t run anywhere and must run with the master does not prevent the promotion of a master

Colocation by Role

Who Gets Promoted

- Allocation occurs as-per previous slides
- Decision of which instances to promote is based on
  - Preference as set by RA with crm_master
  - **Location preferences of resources that wish to be colocated with the master instance(s)**

Colocation by Role

Master/Slave Example

Child	Location	M/S Score
ms:0	node1	1,000
ms:1	node2	100
ms:2	node3	10
ms:3	node4	-INFINITY

Colocation by Role

Changes

- Under the old system, we would
  - sort the children by their m/s score
  - allocate masters in that order (ms:0, ms:1, ms:2)
- Now we include the colocation scores too

Colocation by Role

Master/Slave Example (continued)

Dependent	Location	Score
rsc1	node1	20
rsc2	node2	200
rsc3	node2	-INFINITY
rsc3	node3	2,000
rsc4	[everywhere]	-INFINITY

Colocation by Role

Master/Slave Example (continued)

Child	Location	M/S Score	Final Score
ms:0	node1	1,000	**1,020**
ms:1	node2	100	**-INFINITY**
ms:2	node3	10	**2,010**
ms:3	node4	-INFINITY	**-INFINITY**

Colocation by Role

Master/Slave Example (continued)

- “Final” weight affects sorting order only
  - Negative final score does not prevent the instance from being promoted
- Sort and allocate Masters in order (depending on the number of masters required):
  - ms:2 , ms:0, ms:1
- ms:3 can’t be promoted as it’s m/s score is less than zero

?

abeekhof@suse.de or linux-ha@lists.linux-ha.org