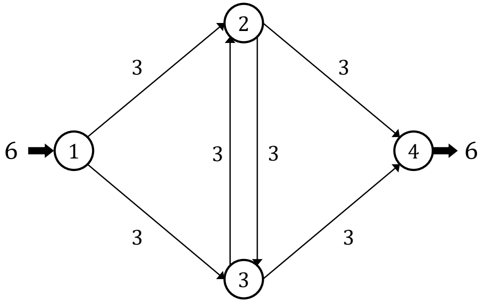
CE 272
Traffic Network Equilibrium

Lecture 22

Acyclic Graphs, Bushes, and Topological Ordering

Previously on Traffic Network Equilibrium...

Consider the following link flow solution. A feasible path flow decomposition is to have 3 travelers each on paths 1-2-3-4 and 1-3-2-4. (For e.g., this could occur in the second iteration of MSA.)



Can we have travelers on both (2,3) and (3,2) at equilibrium? Both MSA and FW will leave some residual cyclic flows.

Previously on Traffic Network Equilibrium...

Over the last two lectures we saw that the equilibrium solutions satisfy two interesting properties.

1. The equilibrium OD flow cannot be present on both sides of a two-way street.
2. The ratio of flows on any two routes between an OD pair is independent of the OD demand. Further, for a given PAS, the ratio of flows on the two segments is same across all OD pairs.

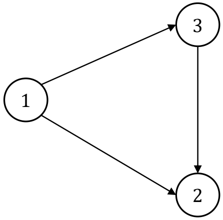
Lecture Outline

Lecture Outline

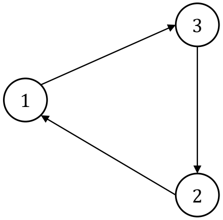
Acyclic Graphs and Bushes

Acyclic Graphs and Bushes
Definitions

An acyclic graph is a directed graph that does not contain any directed cycles. We will also refer to such graphs as Directed Acyclic Graphs (DAGs).



Acyclic Graph



Cyclic Graph

Acyclic Graphs and Bushes
Definitions

A bush rooted at a node (typically an origin in Z) is a sub-graph that satisfies the following properties:

- **Connected**
  Every other node (destination) is reachable from the origin.
- **Acyclic**
  The sub-graph does have any directed cycles.

Acyclic Graphs and Bushes
Definitions

A **tree** is a bush in which every destination is connected by exactly one path.

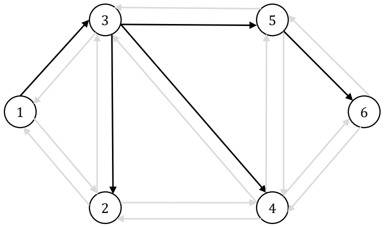
In contrast, there can be multiple paths between the origin and a destination in a bush.

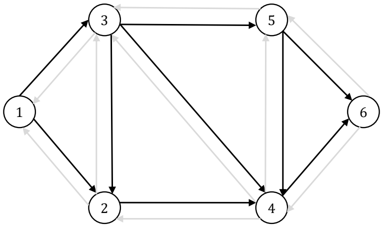
A bush can be viewed as a collection of all used paths of travelers between a given origin and all the destinations.

Acyclic Graphs and Bushes

Examples

Which of these sub-networks are (a) acyclic (b) bushes and (c) trees?

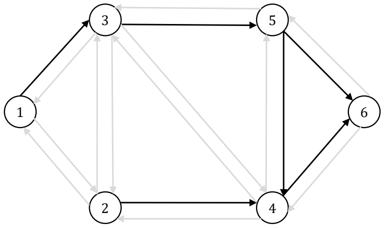


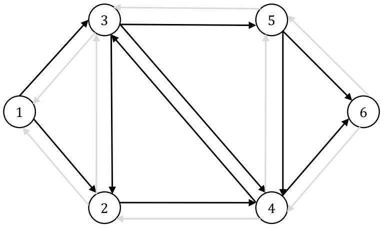


Acyclic Graphs and Bushes

Examples

Which of these sub-networks are (a) acyclic (b) bushes and (c) trees?





Lecture Outline

Topological Ordering

Topological Ordering

Introduction

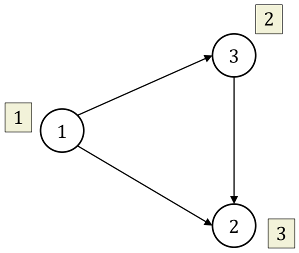
Sometimes, we can relabel the nodes in a graph such that every arc is connected from a lower-labeled node to a higher-labeled node. This procedure is called **topological ordering**.

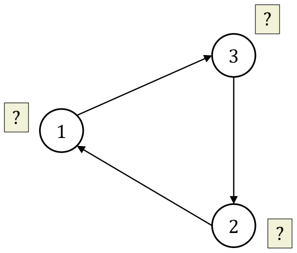
We will use ϑ_i to denote the topological order of node i. In other words, ϑ_i is the new label of node i, and if there exists an arc (i, j), then $\vartheta_i < \vartheta_j$.

Topological Ordering

Introduction

The topological ordering for the graph on the left is shown in the boxes.





Can you find the topological ordering in the figure on the right?

Proposition

A directed network can be topologically ordered iff it is acyclic.

Topological Ordering

Algorithm

To topologically order nodes, define the indegree of a node as the number of arcs coming into it. Consider a node i with zero indegree. Set $\vartheta_i = 1$.

Delete the arcs emanating from i. Pick a new node with indegree zero and set it's topological order to 2.

Repeat this procedure until there are no nodes with zero indegree. If there are leftover nodes, then the network is cyclic.

Topological Ordering

Algorithm

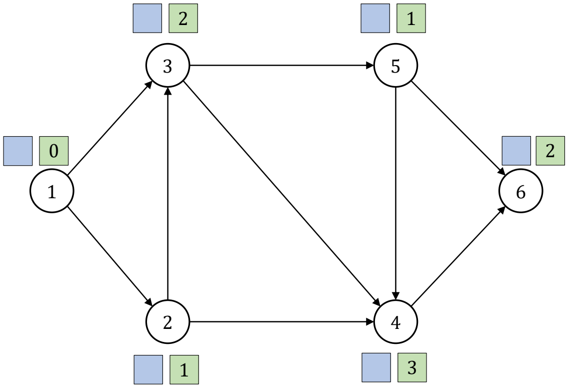
TOPOLOGICAL ORDERING(G)

```
Step 1: Initialize
$indegree(i) \leftarrow 0 \; \forall \, i \in N$
for $(i, j) \in A$ do
    $indegree(j) \leftarrow indegree(j) + 1$
end for
SEL $\leftarrow \emptyset$
$order \leftarrow 0$
Add all nodes with zero indegree to SEL

Step 2:
while SEL$\neq \emptyset$ do
    Remove $i$ from SEL
    $order \leftarrow order + 1$
    $\vartheta_i \leftarrow order$
    for $j : (i, j) \in A$ do
        $indegree(j) \leftarrow indegree(j) - 1$
        if $indegree(j) = 0$ then add $j$ to SEL
    end for
end while
if $order < n$ then the network has a cycle
```

Topological Ordering

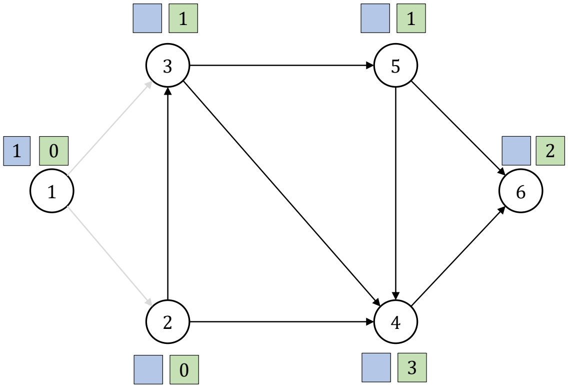
Example



order $= 0$, SEL $= \{1\}$

Topological Ordering

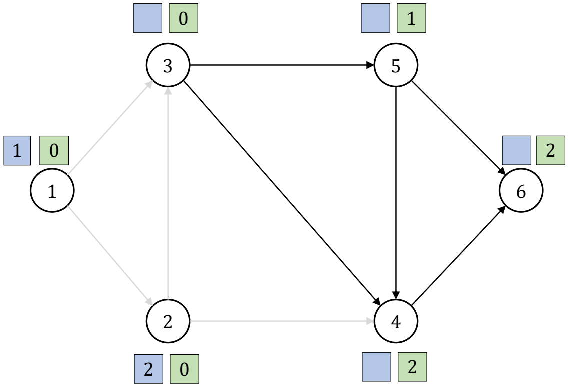
Example



order = 1, SEL = {2}

Topological Ordering

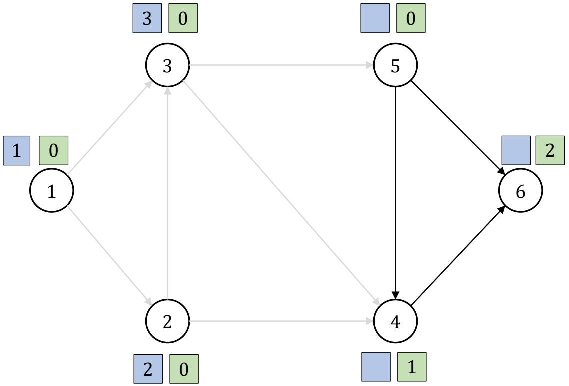
Example



order $= 2$, SEL $= \{3\}$

Topological Ordering

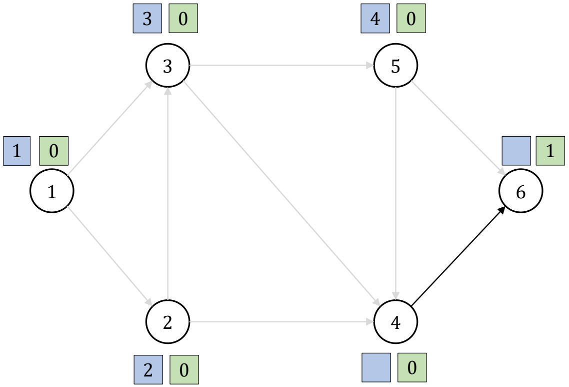
Example



order = 3, SEL = $\{5\}$

Topological Ordering

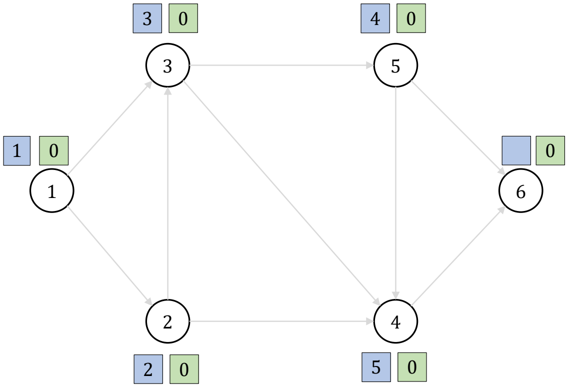
Example



order $= 4$, SEL $= \{4\}$

Topological Ordering

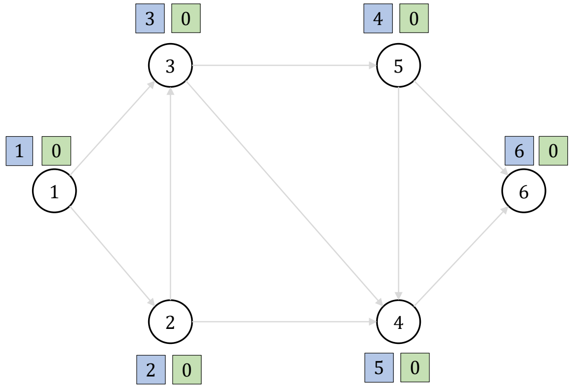
Example



order $= 5$, SEL $= \{6\}$

Topological Ordering

Example



order = 6, SEL = { }

Lecture Outline

Shortest and Longest Paths

Shortest and Longest Paths

Introduction

Earlier in the course, we saw Dijkstra's and label correcting method for finding the shortest paths.

But it turns out that for DAGs, finding these paths is very simple. We just need to update labels by scanning nodes in increasing topological order.

Shortest and Longest Paths

Complexity of Shortest Path Algorithms

Recall that in Dijkstra's algorithm, at each iteration, we move one node from a set of temporary labeled nodes to a set of permanently labeled nodes.

Finding the node with minimum label is implementation specific, but the simplest version takes

Iteration	Steps
1	n
2	$n-1$
3	$n-2$
$\vdots$	$\vdots$
n	1
Total	$n(n+1)/2$

The distance labels are updated at most m times. (Why?) Hence, the overall complexity is $O(n^2+m)$. Since $n^2 \gg m$, we simply say that Dijkstra's runs in $O(n^2)$ time.

Shortest and Longest Paths

Complexity of Shortest Path Algorithms

In topological ordering, we essentially delete arcs and update the indegrees of nodes. Hence, the complexity of topological ordering is $O(m)$.

To find the shortest path in a DAG, we simply scan nodes in increasing topological order and update the labels of the downstream nodes.

In this version, we do not find a node with minimum distance label nor do we keep track of a SEL. Each arc is scanned at most once. Hence, the complexity of this method is $O(m)$.

Shortest and Longest Paths
Shortest Paths on DAGs

Proposition

The proposed algorithm converges to the optimal distance labels.

Proof.

The proof is based on induction. Assume that after iteration k, the algorithm has *optimally* labeled nodes $i_1, i_2, \ldots, i_k$ whose topological order is $1, 2, \ldots, k$ respectively.

At iteration $k + 1$, the algorithm scans the upstream arcs of node i_{k+1} and updates it's label.

Since, this node cannot be reached from other nodes that have a higher topological order, and since all of it's upstream nodes are optimally labeled (from induction hypothesis), the label of node i_{k+1} has to be optimal. ■

Shortest and Longest Paths

Shortest Paths on DAGs

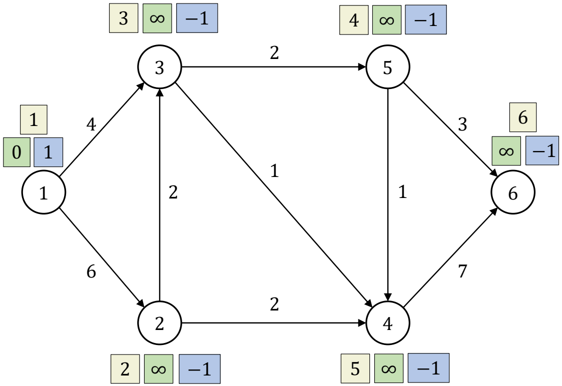
```
SHORTEST PATH($G, r$)

Step 1: Initialize
TOPOLOGICAL ORDERING($G, r$)
$\mu_r \leftarrow 0, \pi_r \leftarrow r$
$\mu_i \leftarrow \infty, \pi_i \leftarrow -1 \, \forall \, i \in N \backslash \{r\}$
$order \leftarrow 1$

Step 2:
while $order \leq n$ do
    $order \leftarrow order + 1$
    Select $j \in N : \vartheta_j = order$
    for $i : (i, j) \in A$ do
        if $\mu_j > \mu_i + t_{ij}$ then
            $\mu_j \leftarrow \mu_i + t_{ij}$
            $\pi_j \leftarrow i$
        end if
    end for
end while
```

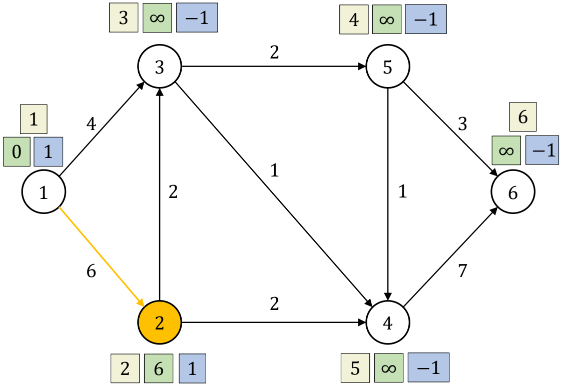
Shortest and Longest Paths

Example



Shortest and Longest Paths

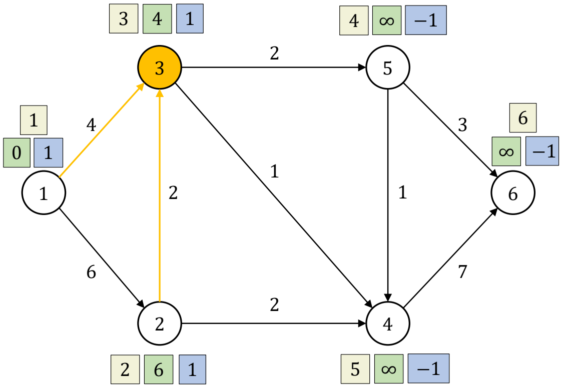
Example



order $= 2$

Shortest and Longest Paths

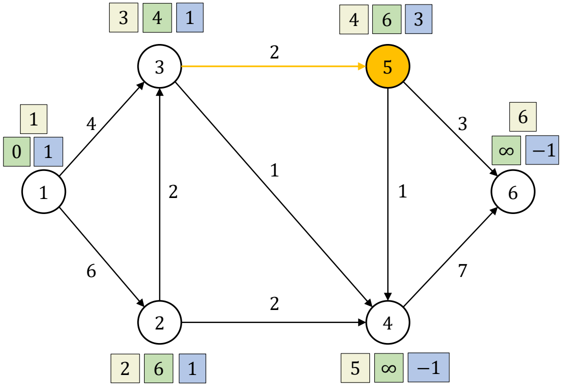
Example



order $= 3$

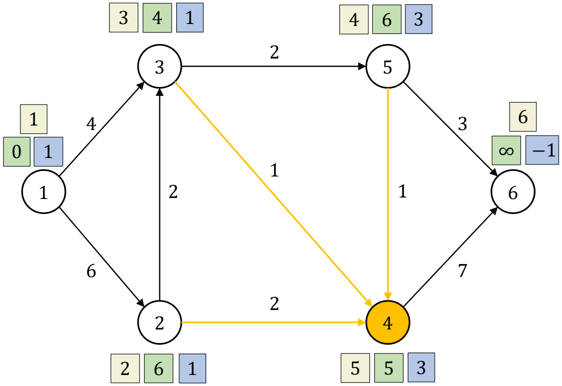
Shortest and Longest Paths

Example



order $= 4$

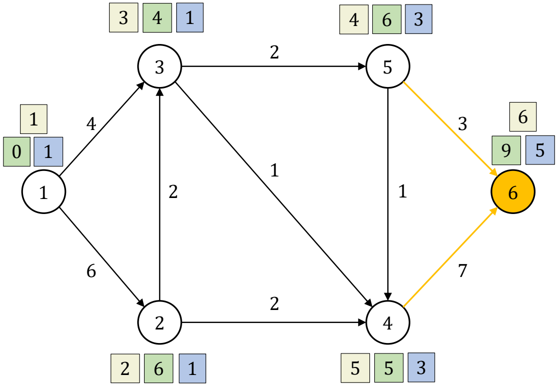
Shortest and Longest Paths
Example



order = 5

Shortest and Longest Paths

Example



order $= 6$

Shortest and Longest Paths

Longest Paths on DAGs

For some bush-based algorithms, we not only need the shortest paths but we also have to compute the longest paths for different OD pairs.

For general graphs, finding the simple/elementary longest path (one without cycles) is NP-Hard (they cannot be solved in polynomial time).

However, for DAGs, we can modify the earlier algorithm and find the one-to-all longest paths in $O(m)$ time.

Shortest and Longest Paths

Algorithm

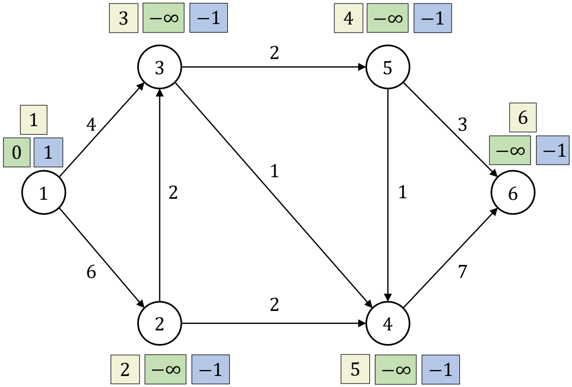
```
LONGEST PATH$(G, r)$

Step 1: Initialize
TOPOLOGICAL ORDERING$(G, r)$
$\nu_r \leftarrow 0, \pi_r \leftarrow r$
$\nu_i \leftarrow -\infty, \pi_i \leftarrow -1 \, \forall \, i \in N \backslash \{r\}$
$order \leftarrow 1$

Step 2:
while $order \leq n$ do
    $order \leftarrow order + 1$
    Select $j \in N : \vartheta_j = order$
    for $i : (i, j) \in A$ do
        if $\nu_j < \nu_i + t_{ij}$ then
            $\nu_j \leftarrow \nu_i + t_{ij}$
            $\pi_j \leftarrow i$
        end if
    end for
end while
```

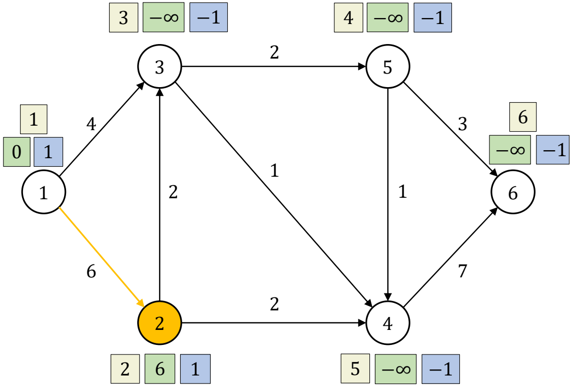
Shortest and Longest Paths

Example



Shortest and Longest Paths

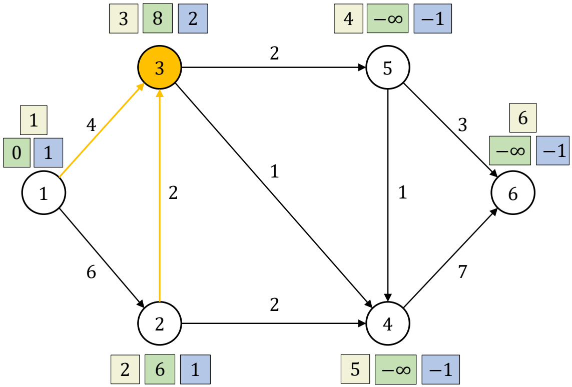
Example



order $= 2$

Shortest and Longest Paths

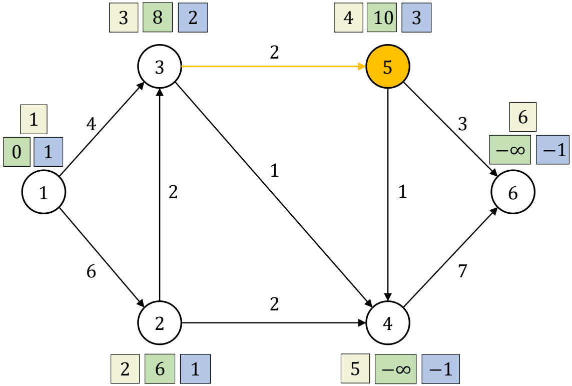
Example



order $= 3$

Shortest and Longest Paths

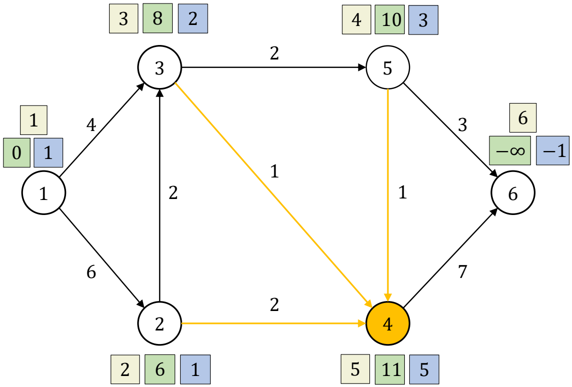
Example



order $= 4$

Shortest and Longest Paths

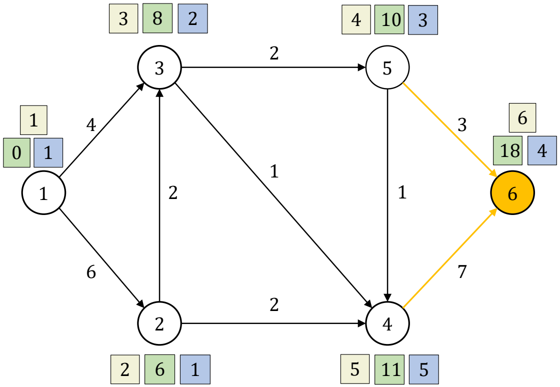
Example



order $= 5$

Shortest and Longest Paths

Example



order = 6

Your Moment of Zen

Cornered
by Mike Baldwin
11-2 © 2006 Mike Baldwin / Dist. by Universal Press Syndicate www.cornered.com
cornered@comic.com

“He passed the sobriety test. Now I think he’s just showing off.”