Computer Science 112
Data Structures

Lecture 22:
Graphs:
Breadth First Search
Shortest Path

Review: Graph Traversals

Depth First Traversal

- **for each vertex v in graph:**

  call dfsG(v)

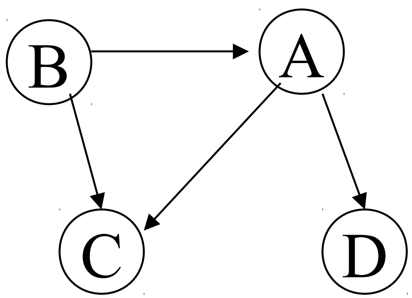
- **dfsG(v):**

  if (marked(v)) return;
  visit v;
  mark v;

  for each vn in neighbors(v)

  dfsG(vn)

Graph Traversals



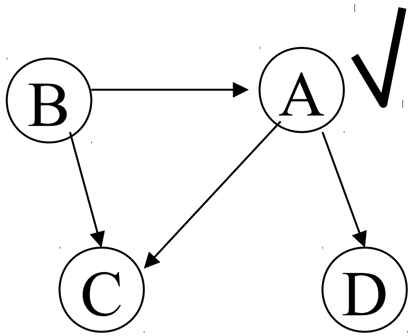
Driver

v = <A>

dfsG

v = <A>

Graph Traversals



Driver

v = <A>

dfsG

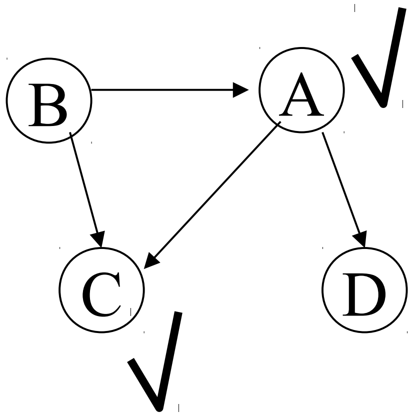
v = <A>

vn = <C>

dfsG

v = <C>

Graph Traversals

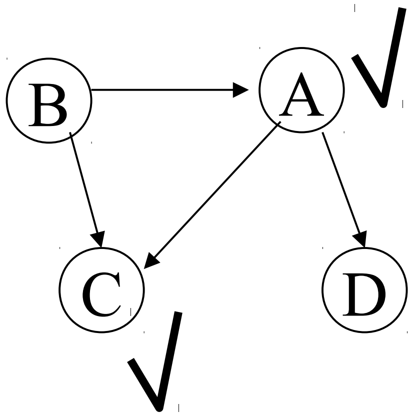


Driver
v = <A>

dfsG
v = <A>
vn = <C>

dfsG
v = <C>

Graph Traversals



Driver

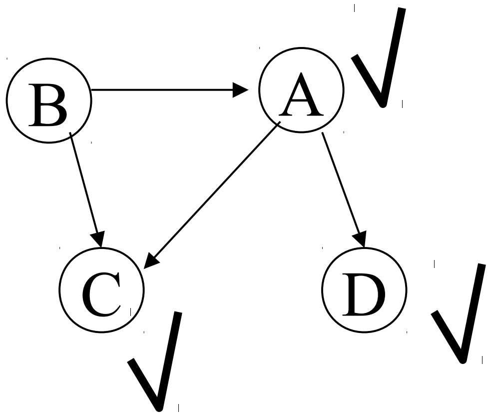
v = <A>

dfsG

v = <A>

vn = <D>

Graph Traversals

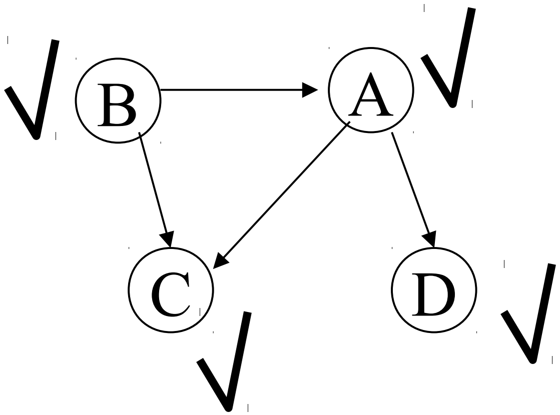


Driver
v = <A>

dfsG
v = <A>
vn = <D>

dfsG
v = <D>

Graph Traversals



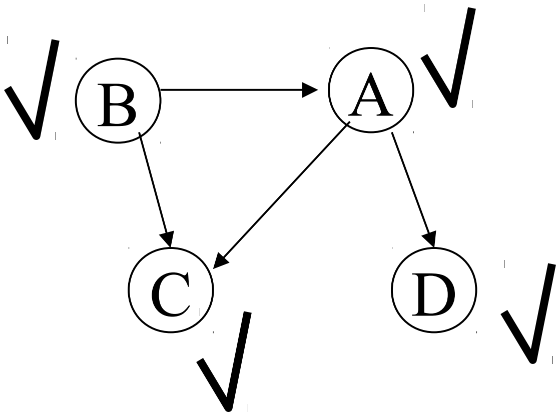
Driver

v = <B>

dfsG

v = <B>

Graph Traversals



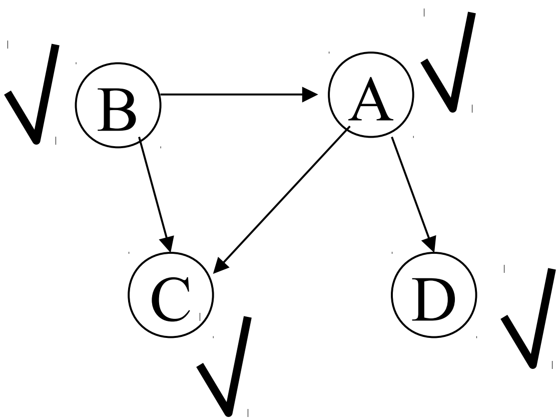
Driver

v = <C>

dfsG

v = <C>

Graph Traversals



Driver

v = <D>

dfsG

v = <D>

Graph Traversals

- **Time:**
  - **Visit each vertex**
  - **inspect each edge**

  O(n + e) n vertices, e edges

Uses of DFS Traversal

- Connected Components
  - See GraphCC.java
- Topsort
  - See GraphTS.java

Review: Topological Sort

- Acyclic Digraph <=> partial order
- Topsort: find total order consistent with partial order

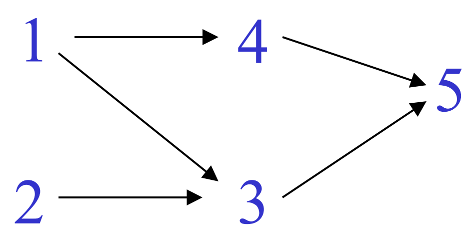
1 **a=1;**

2 **b=2;**

3 **c=a*b;**

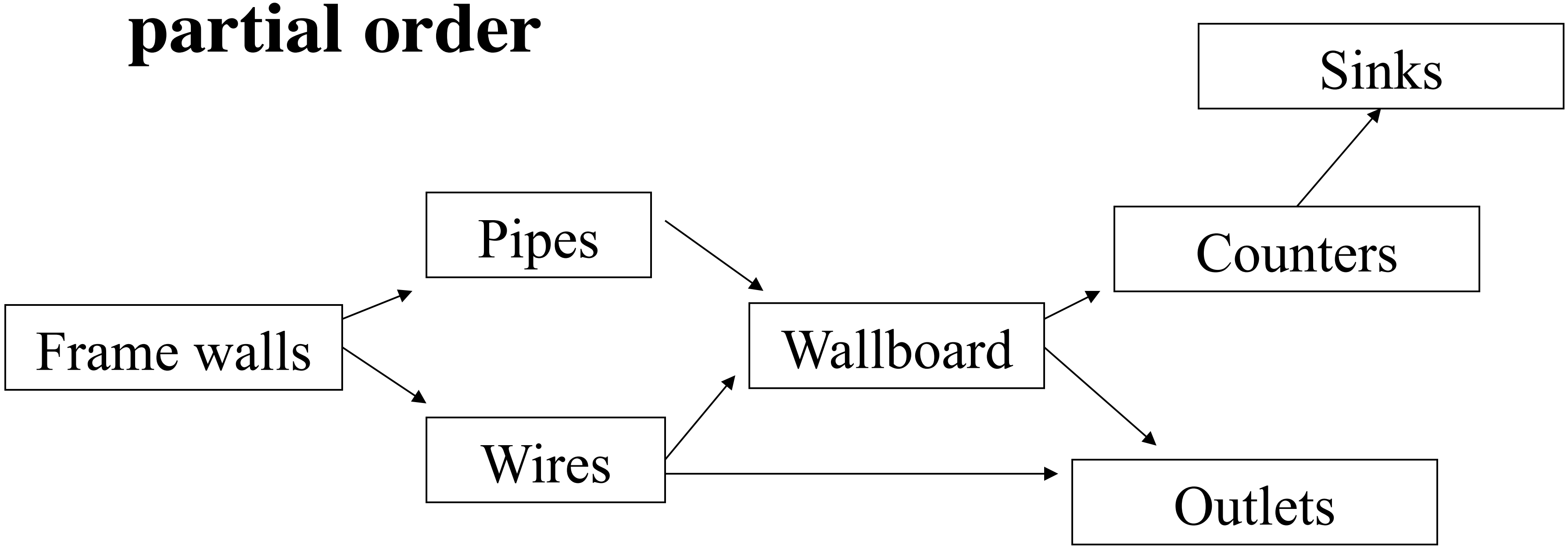
4 **d=a+4;**

5 **c=c+d**



Topological Sort

- **Acyclic Digraph <=> partial order**
- **Topsort: find total order consistent with partial order**



Topsort Algorithms

- **Most work by assigning numbers to vertices**
  - **topsorted order = numerical order**
- **Depth first**

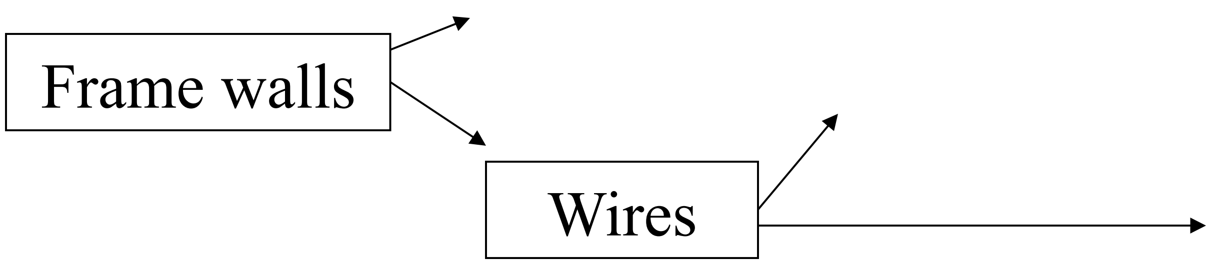
DFS Topsort Algorithm

- **Algorithm:**
  - **Do DFS**
  - **Number vertices as you leave them**
- **Problem: leave vertex *after* leave reachable vertices, but needs number *smaller* than reachable vertices**
  - **Solution: number from largest to smallest numbers**
- **See GraphTS.java**

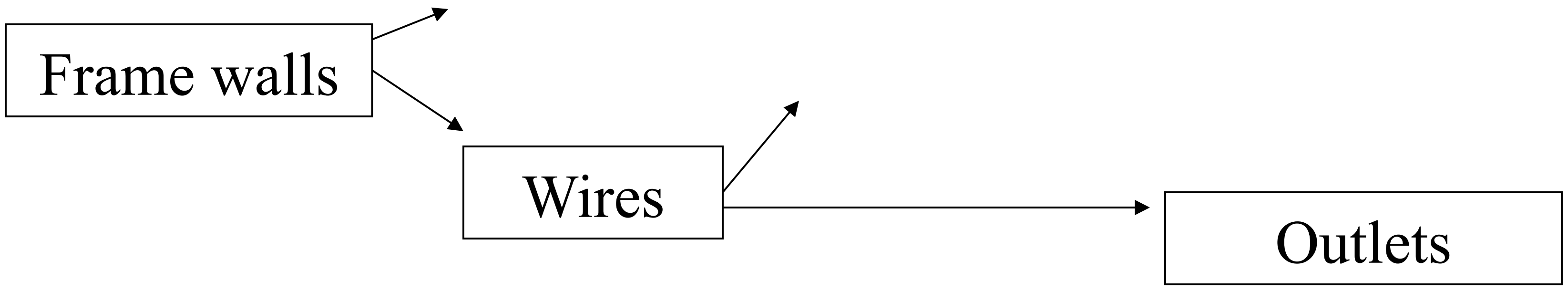
New: Topsort Example

Frame walls

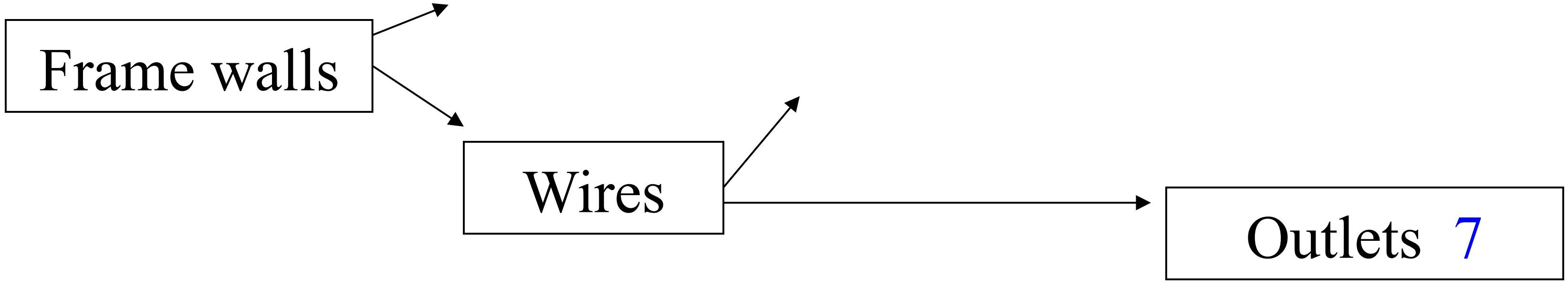
New: Topsort Example



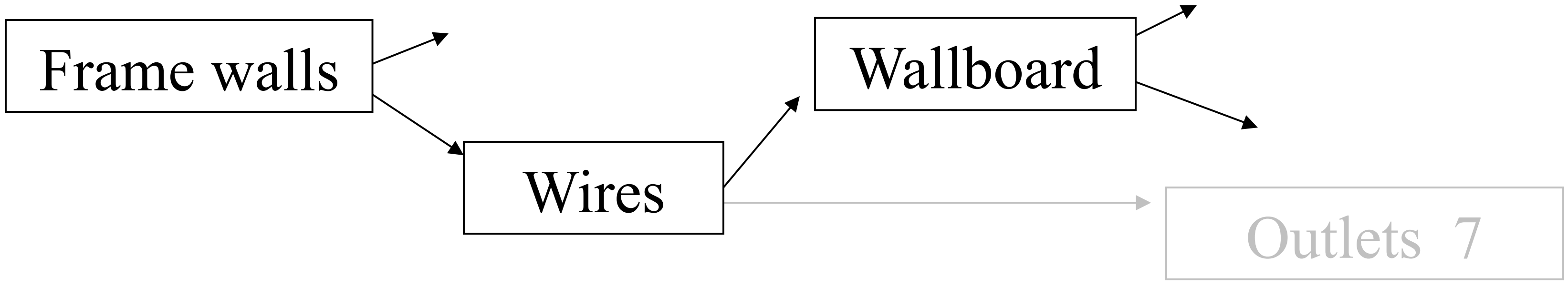
Topsort Example



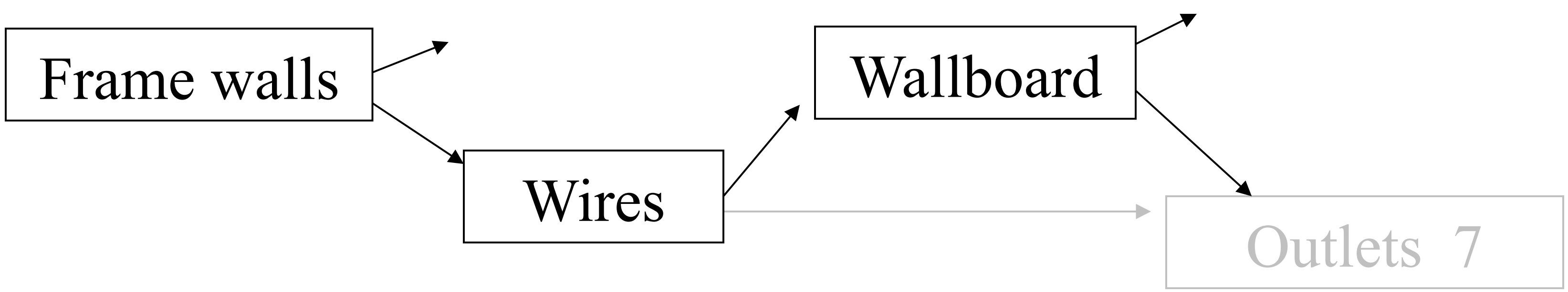
Topsort Example



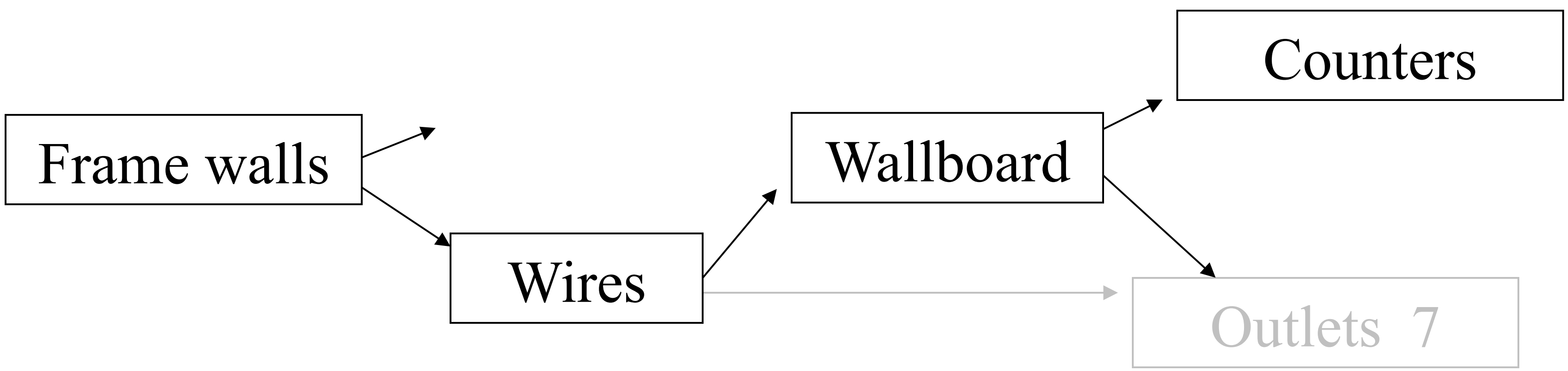
Topsort Example



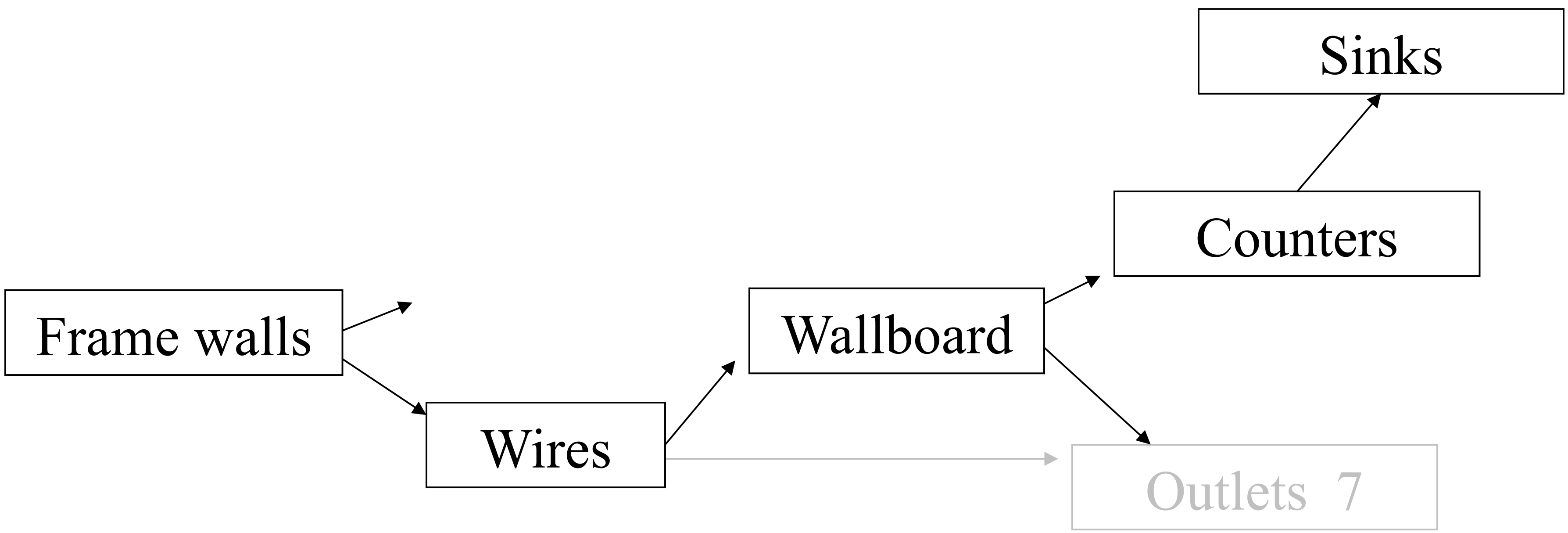
Topsort Example



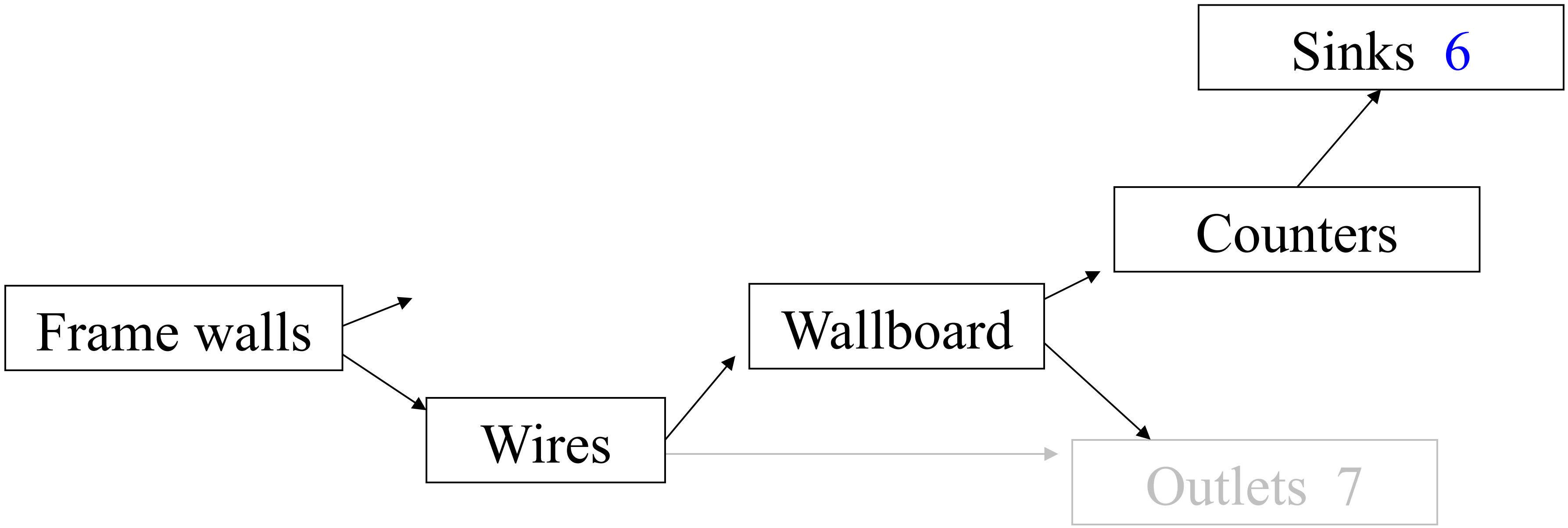
Topsort Example



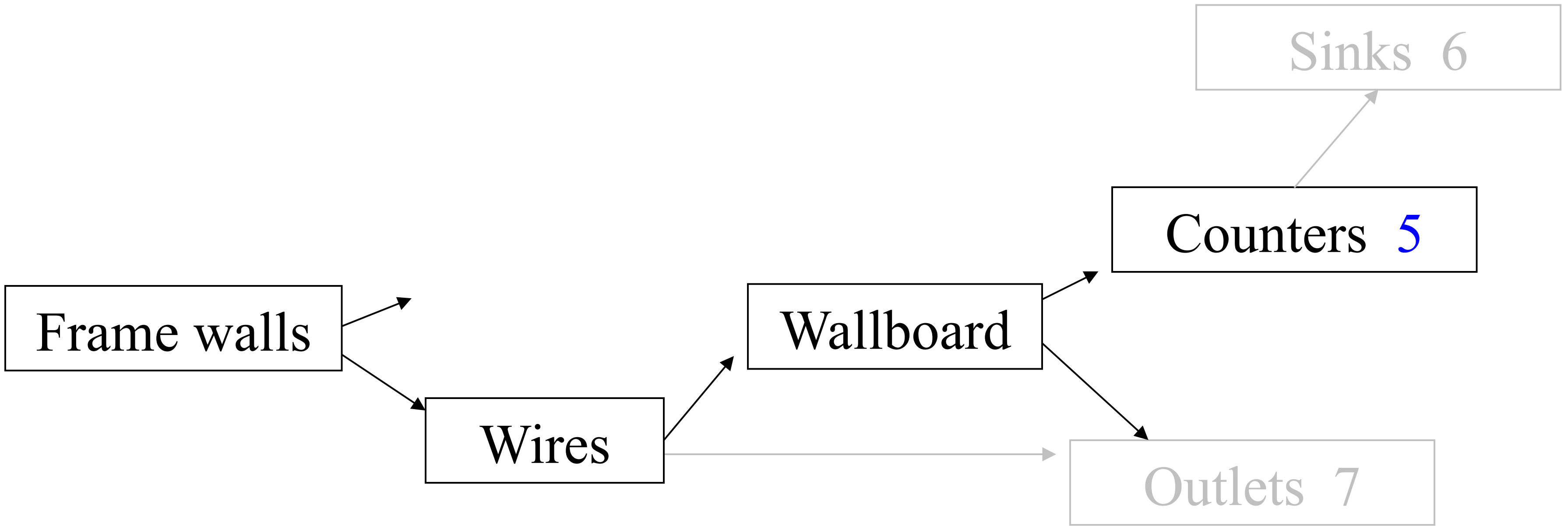
Topsort Example



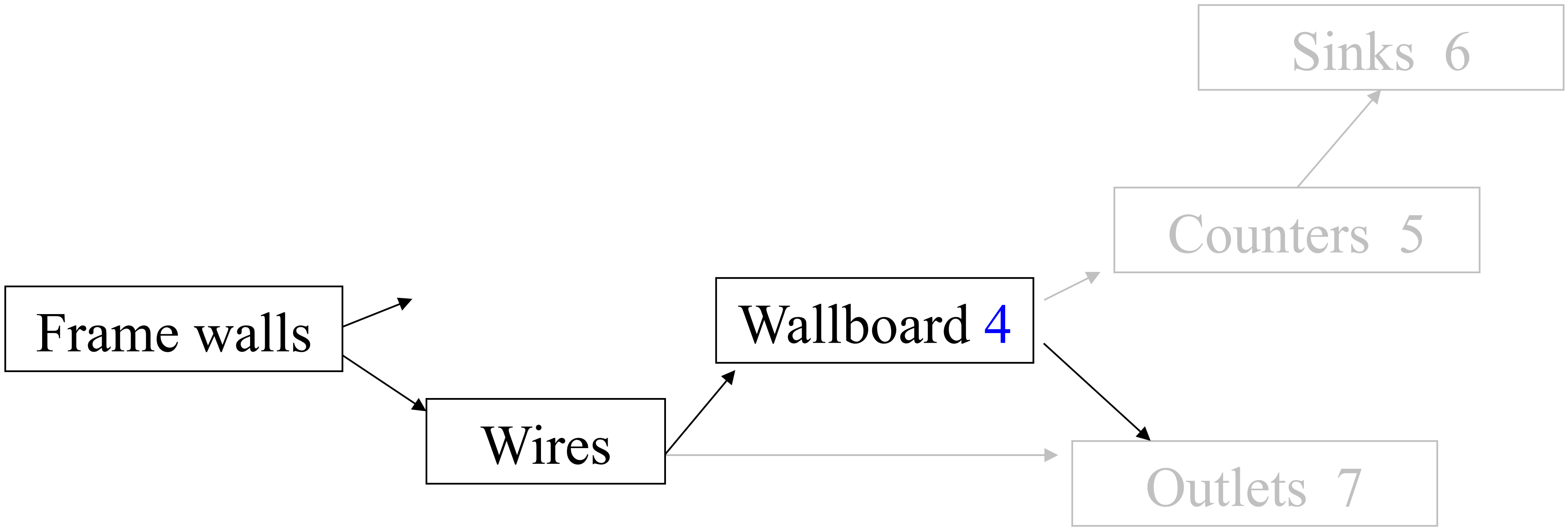
Topsort Example



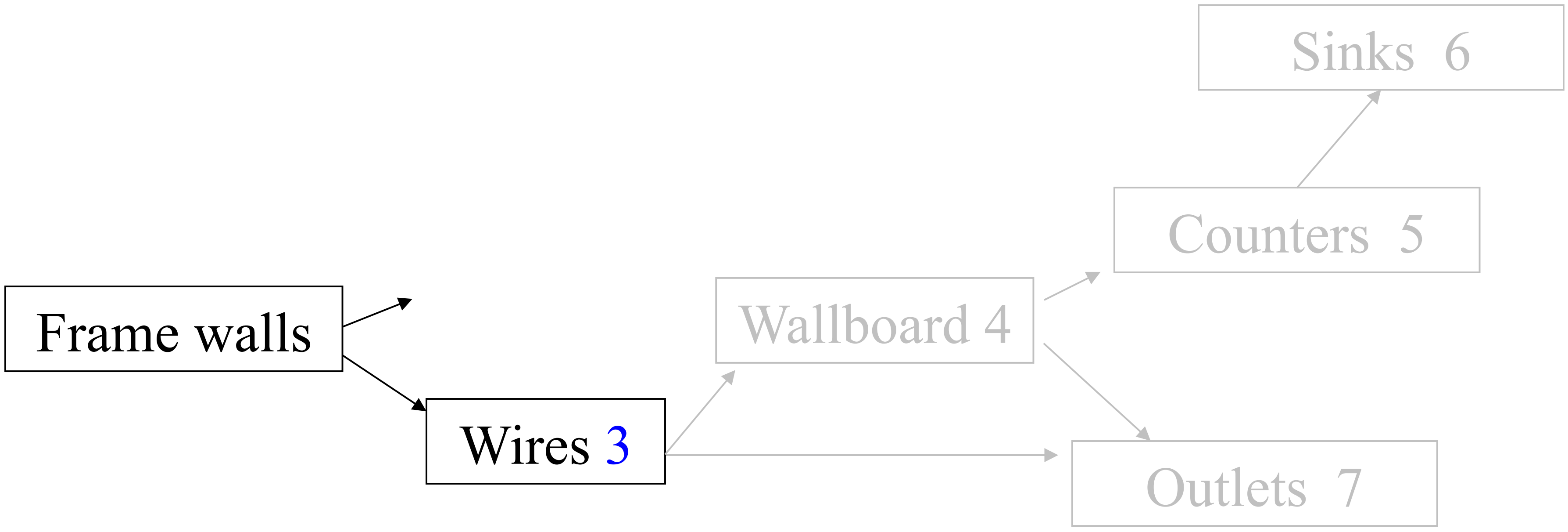
Topsort Example



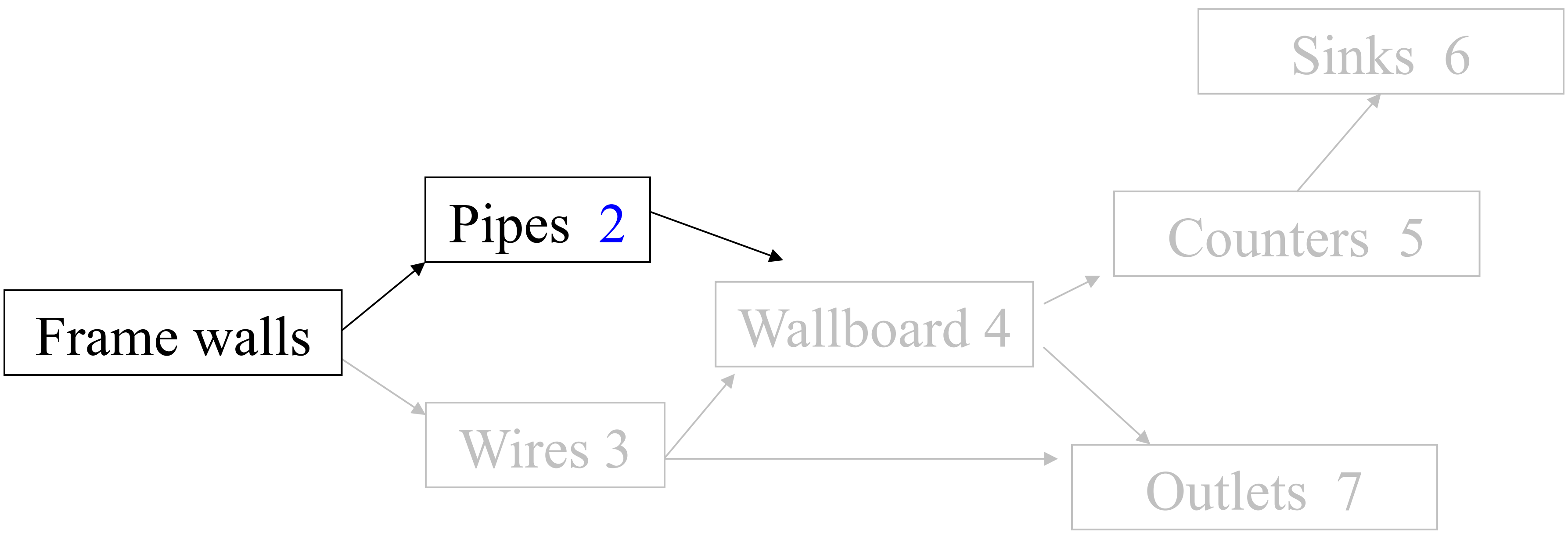
Topsort Example



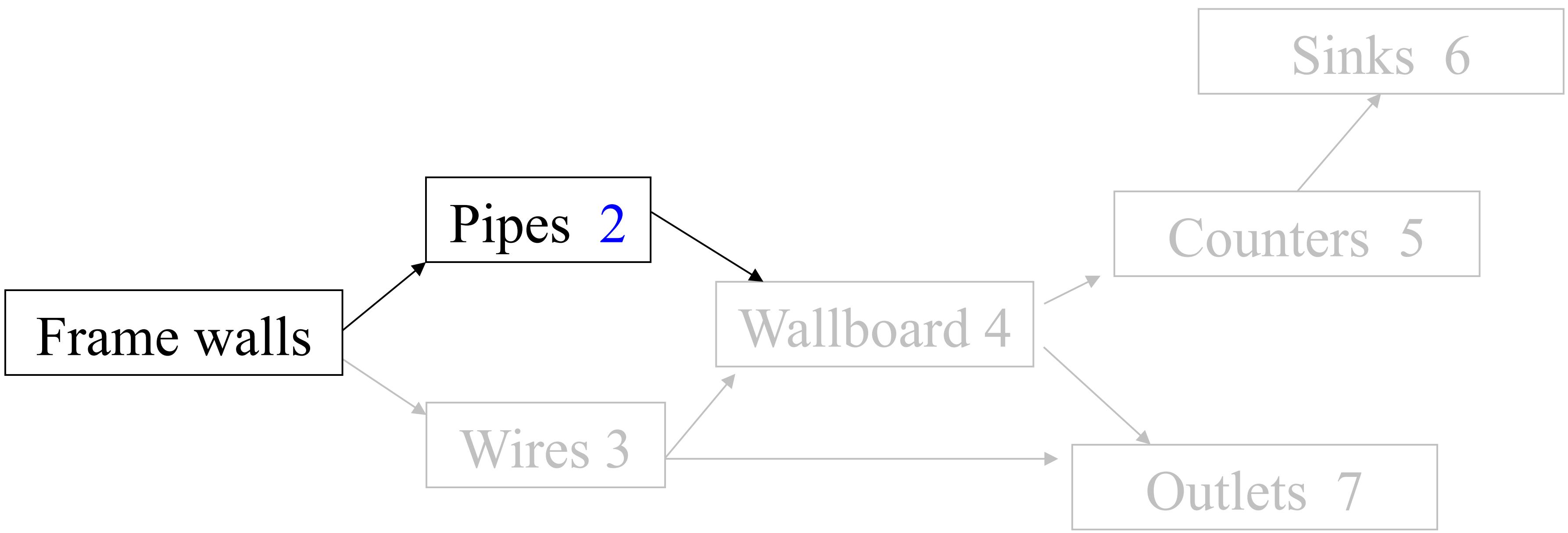
Topsort Example



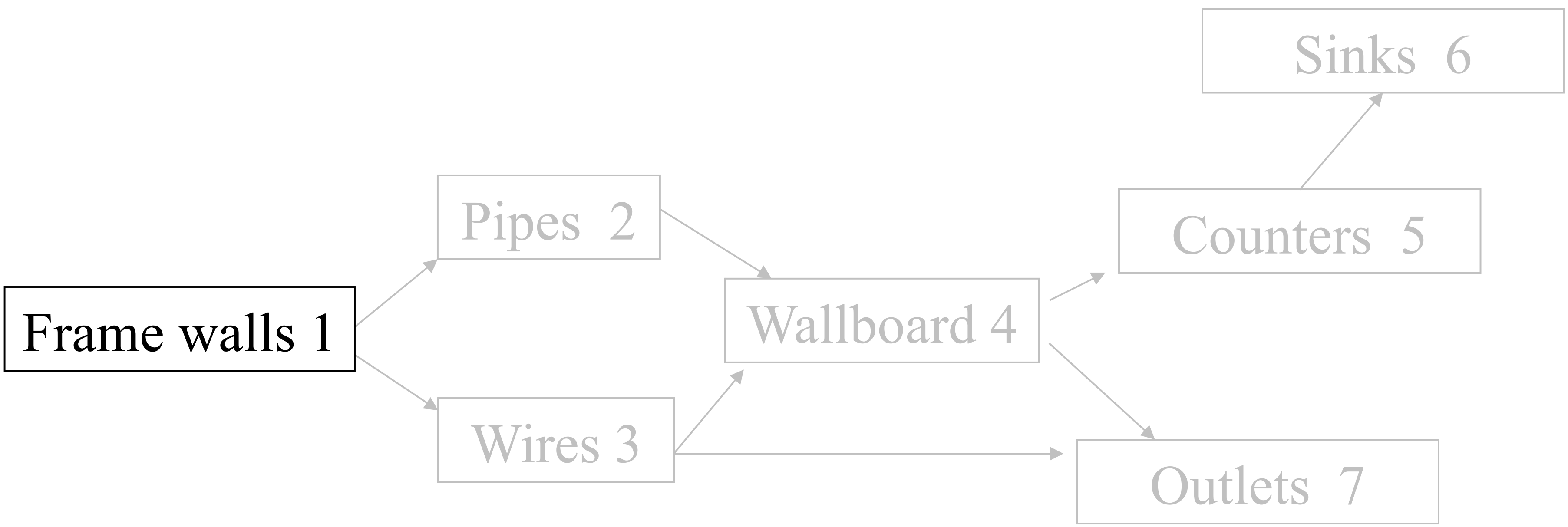
Topsort Example



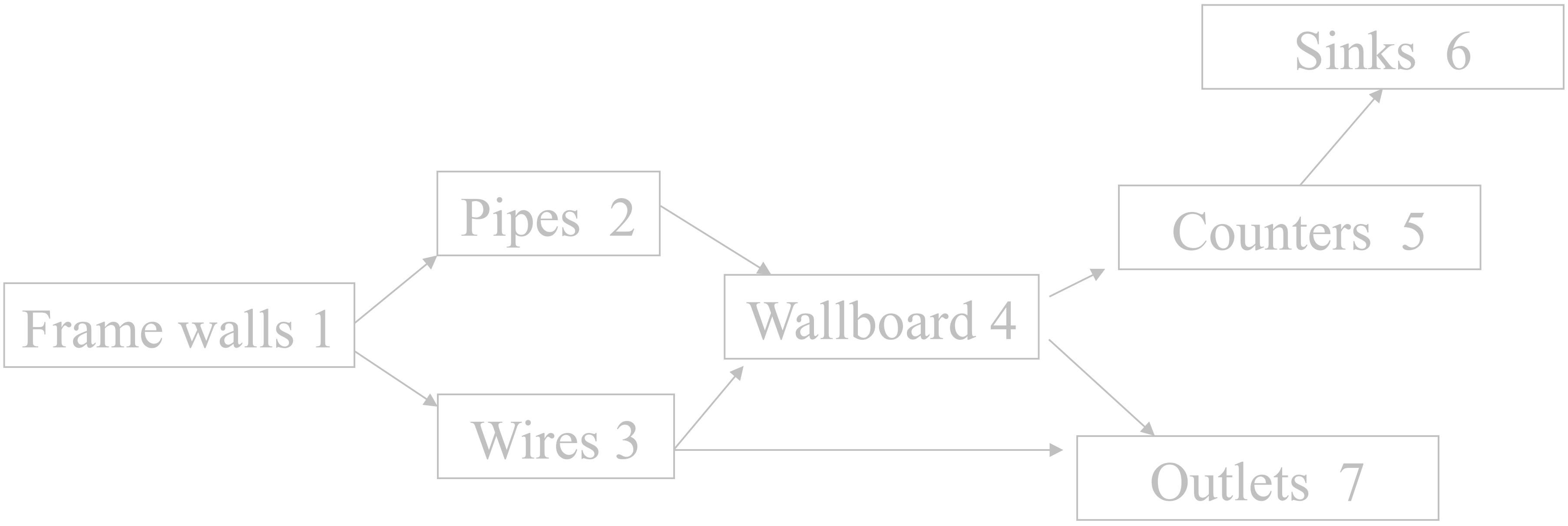
Topsort Example



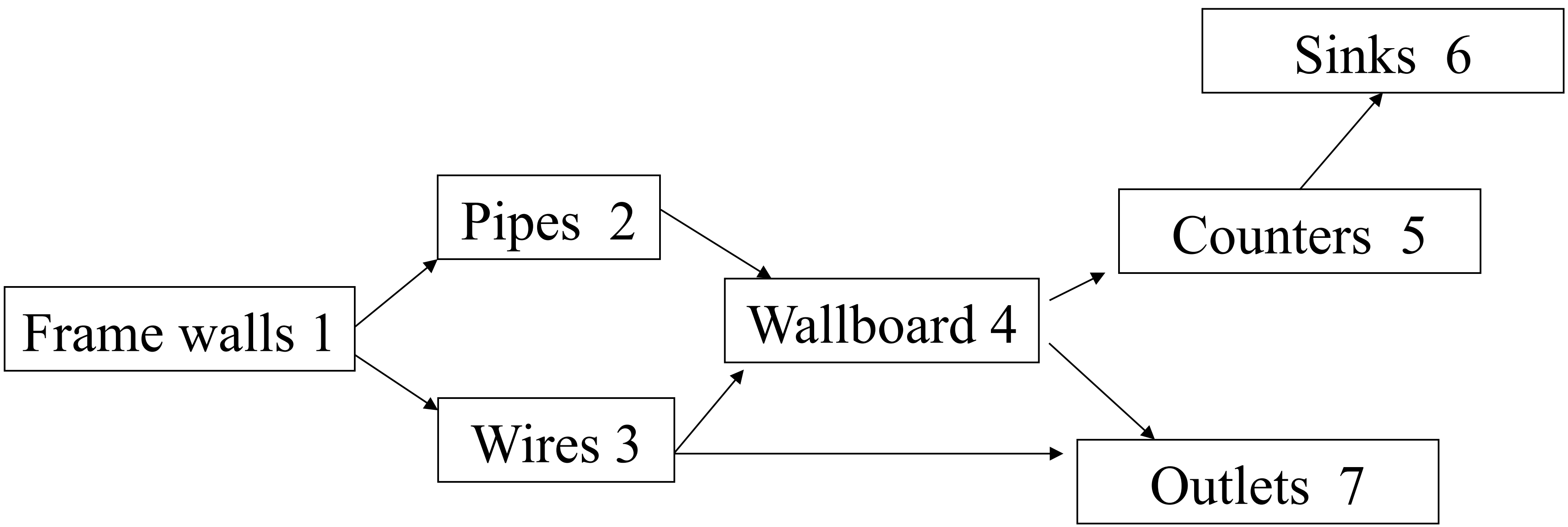
Topsort Example



Topsort Example



Topsort Example



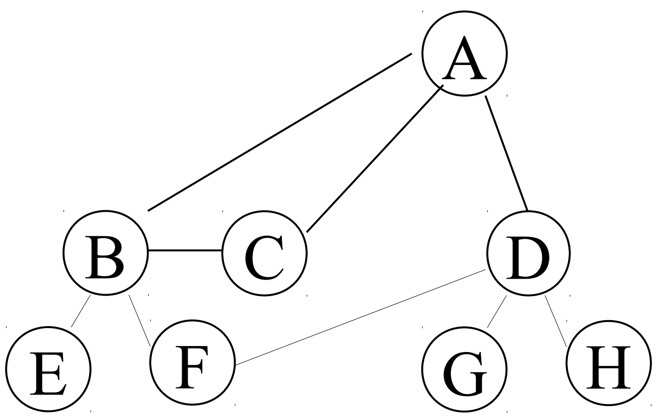
Topsort Cost

- **DFS + numbering**

  = O(n+e) + O(n) = O(n+e)

New: Breadth First Search

- **Like breadth first search on tree**



- **Breadth First: A B C D E F G H**
- **Depth First: A B E F D G H C**

Breadth First Algorithm

- **bfsG(v):**
  visit and mark v
  enqueue v
  while not queue.empty( )
      dequeue into w
      for each neighbor n of w:
          if n not visited:
              visit and mark n
              enqueue n

Breadth First Cost

- **like depth-first:**
  - **visit every vertex**
  - **cross every edge**

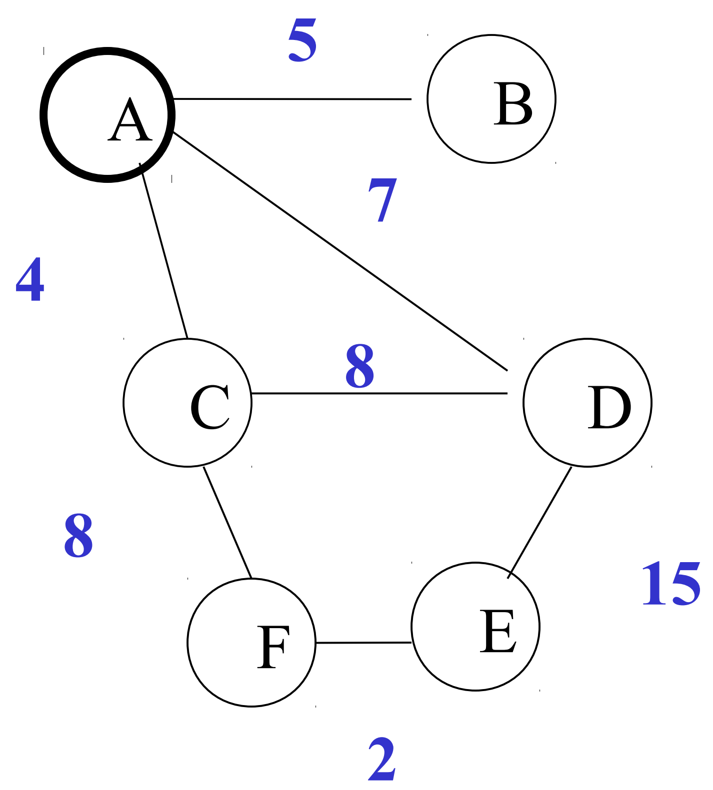
  O(n+e)

Shortest Path

- weighted digraph
  - weights are all > 0
- "length" of a path = sum of weights of arcs on path
- given start vertex, end vertex, find shortest path from start to end

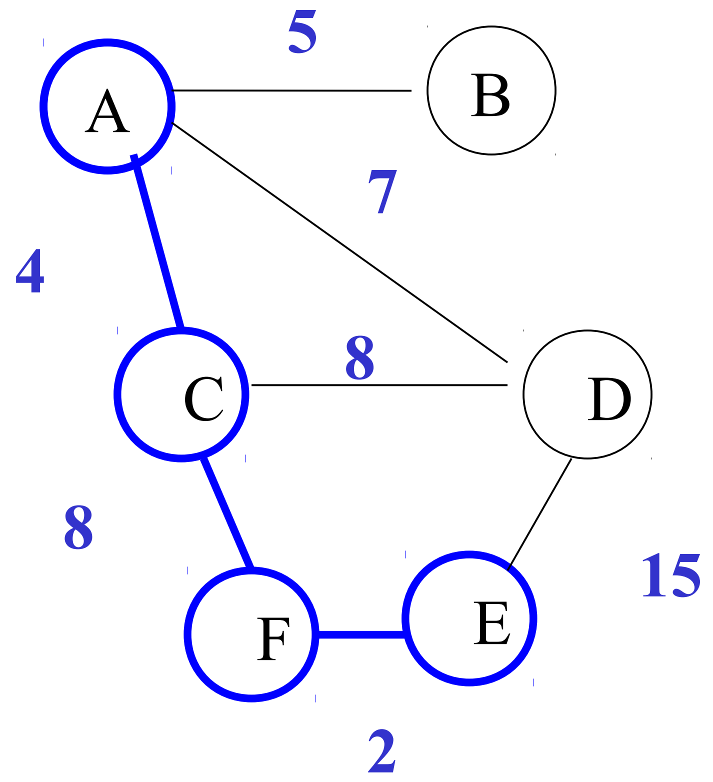
Shortest Paths

- **What is the shortest path**
  - **from A to E?**
  - **from A to F?**



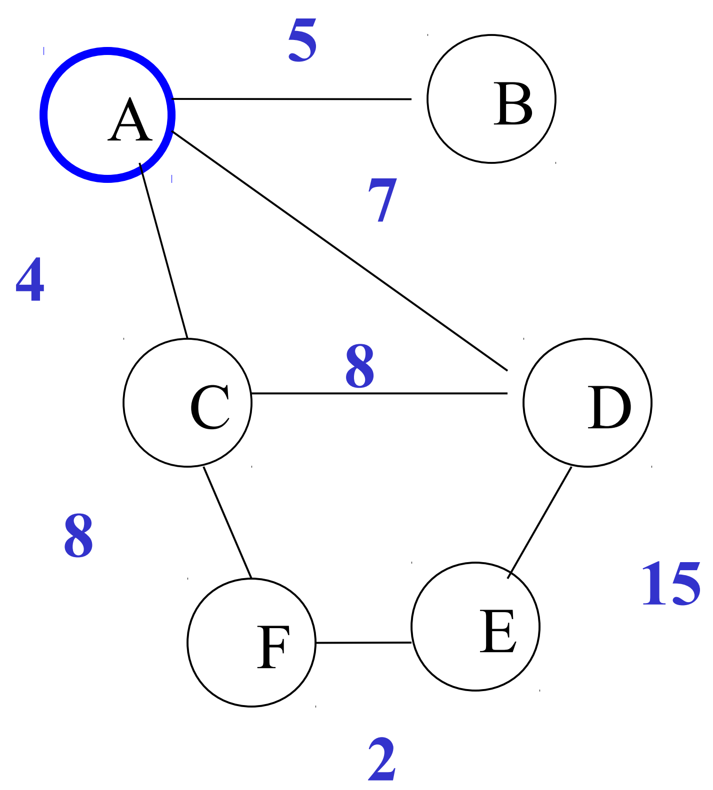
Shortest Paths

- **If a shortest path from A to E runs through F, the part from A to F is a shortest path from A to F**



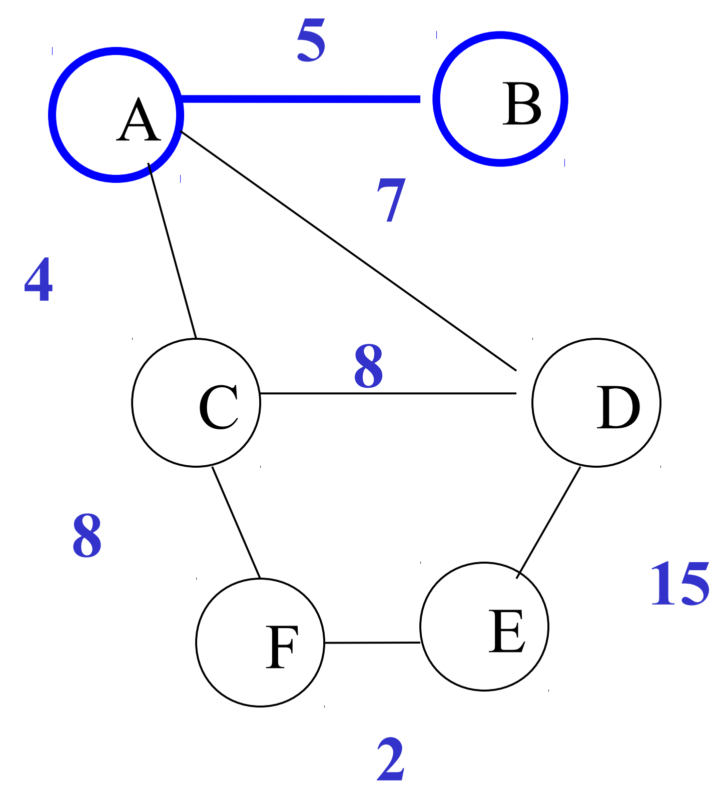
Dijkstra's Algorithm

- **Consider the shortest paths from A to each other vertex.**



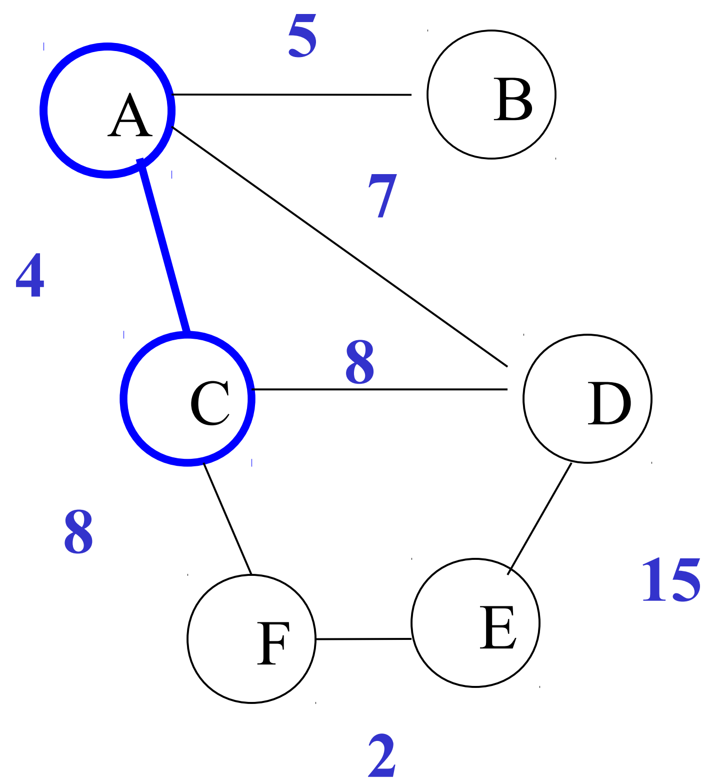
Dijkstra's Algorithm

- **Consider the shortest paths from A to each other vertex.**



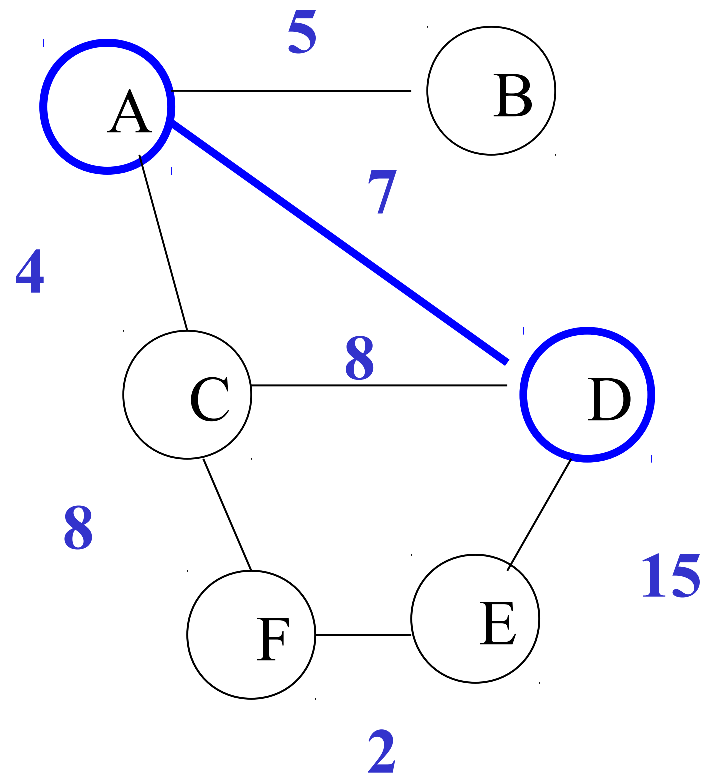
Dijkstra's Algorithm

- **Consider the shortest paths from A to each other vertex.**



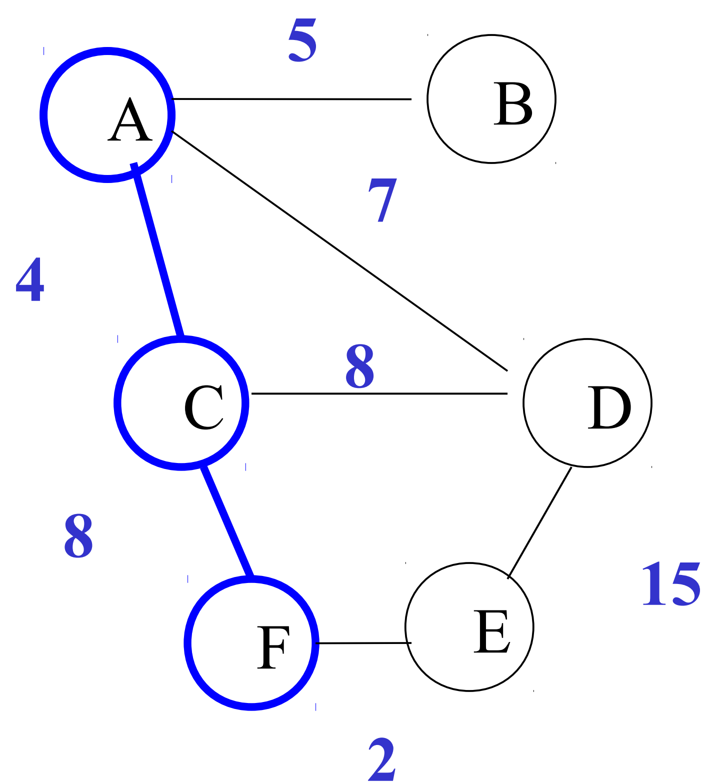
Dijkstra's Algorithm

- **Consider the shortest paths from A to each other vertex.**



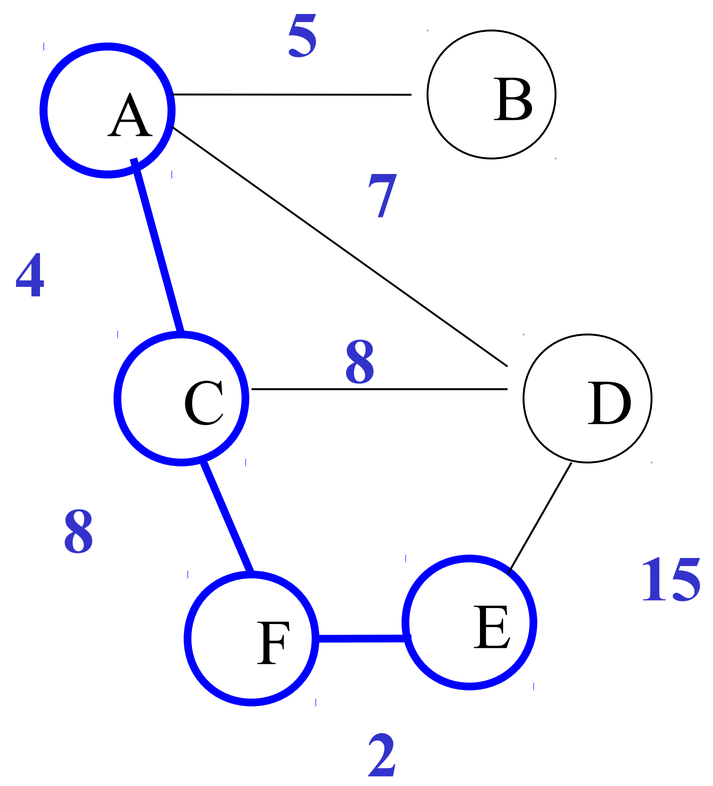
Dijkstra's Algorithm

- **Consider the shortest paths from A to each other vertex.**



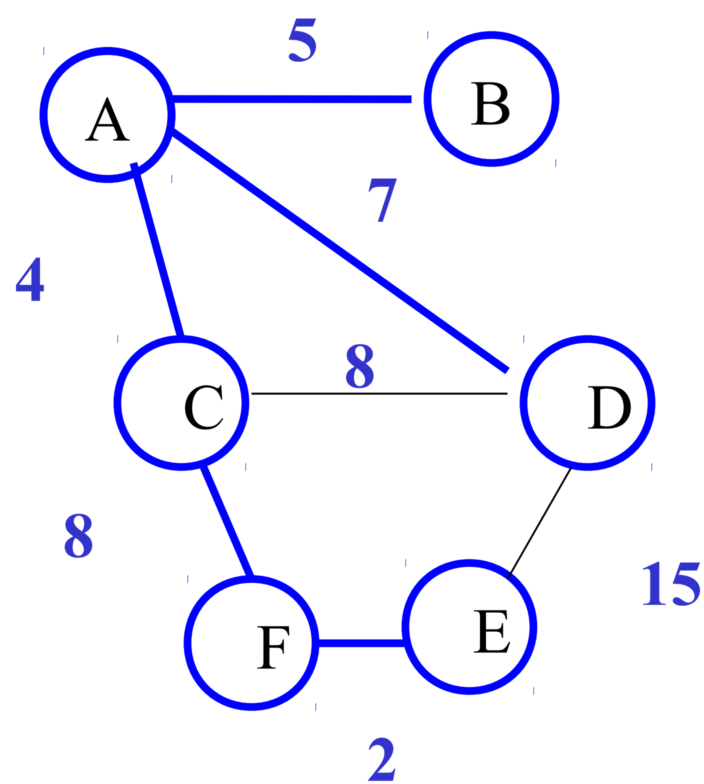
Dijkstra's Algorithm

- **Consider the shortest paths from A to each other vertex.**



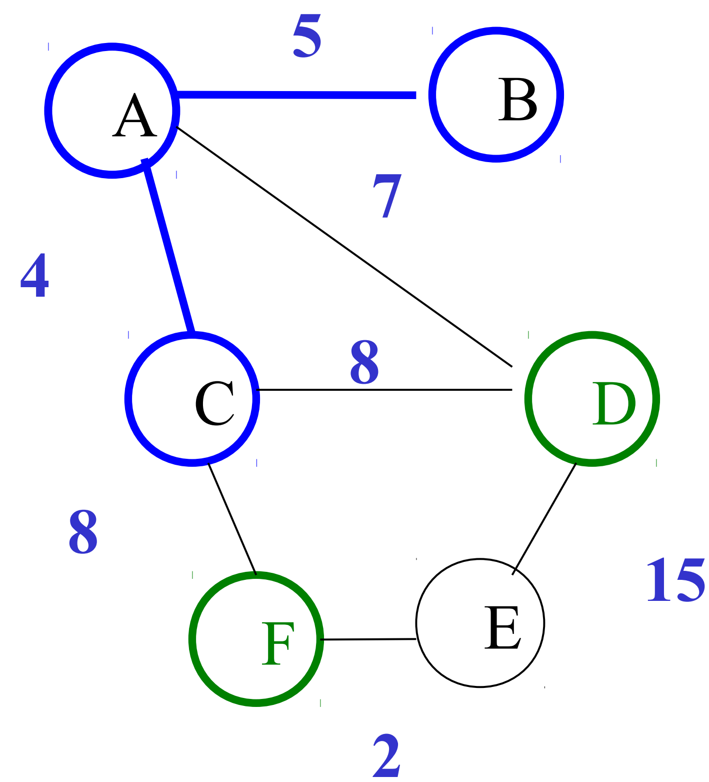
Dijkstra's Algorithm

- **These can be put together to form a tree**
- **A *Shortest Path Tree***



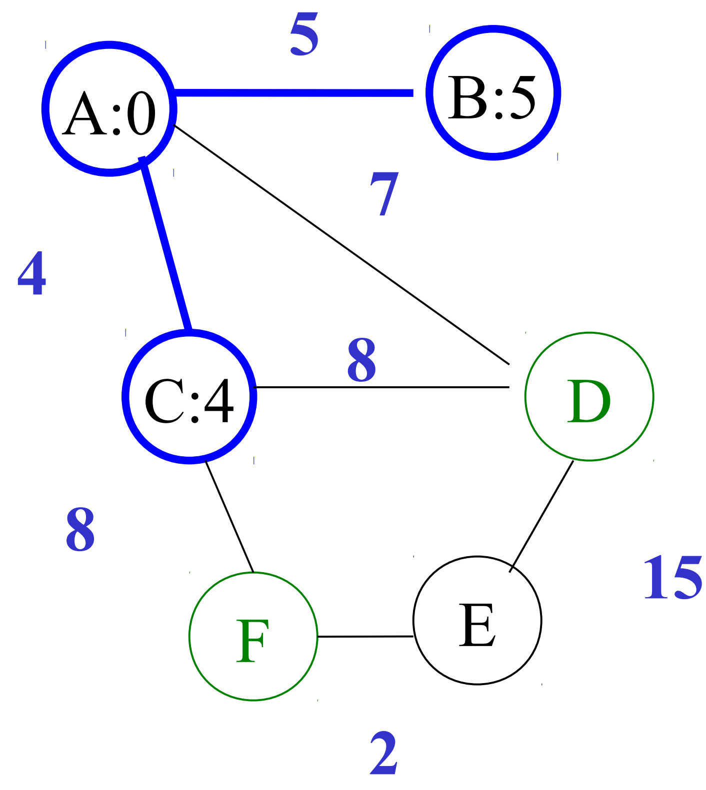
Dijkstra's Algorithm

- **Grow a tree of shortest paths from start**
  - **grow it one vertex at a time, closest to farthest**
- **Fringe: nodes that are not in the tree yet but have a neighbor in the tree**



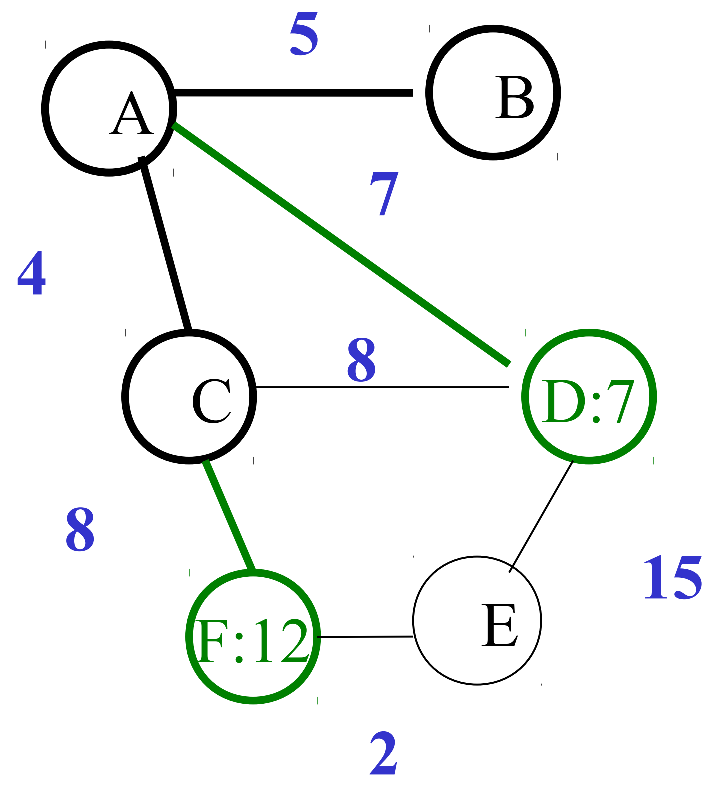
Dijkstra's Algorithm

- **Vertices in the tree have**
  - **a link: first step on the shortest path back to start**
  - **a distance: the length of that whole path**



Dijkstra's Algorithm

- **Vertices in the fringe have**
  - **a link: an arc to the tree if > 1 of these, use the arc that gives the shortest path back to start**
  - **a distance: the length of the path using link**



Algorithm:

Put start vertex in the tree

While there are any vertices in fringe

Let v be vertex in fringe with

smallest distance-from-start.

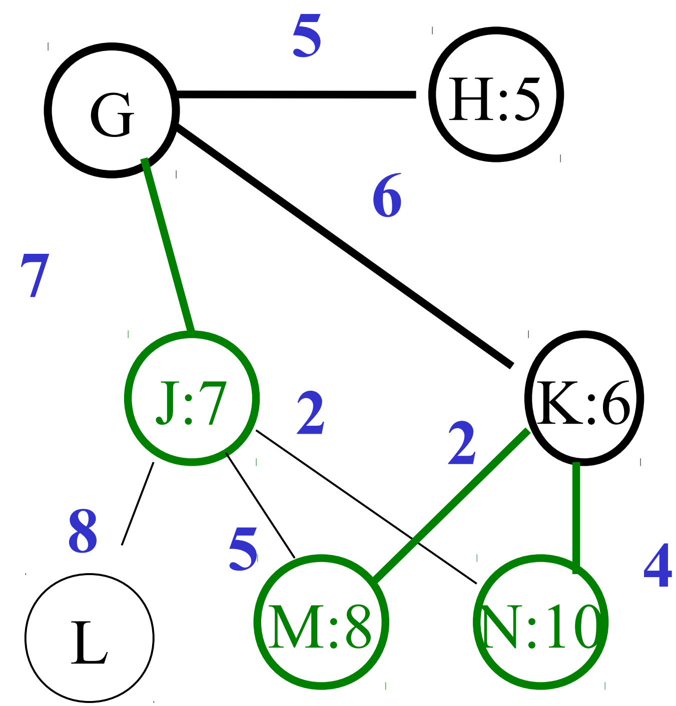
Put v in the tree.

Update fringe

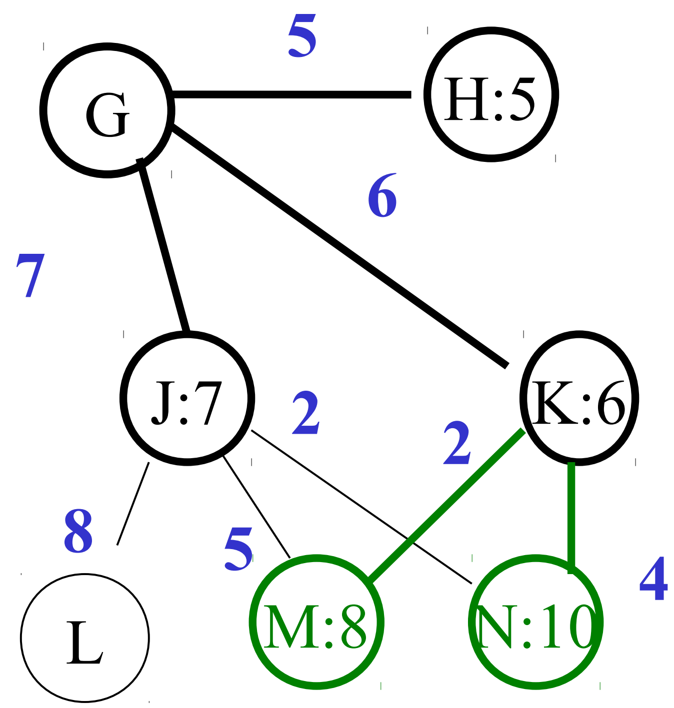
Update fringe

- **Neighbors of v that are not in tree or fringe get added to fringe**
- **Neighbors of v that are in the fringe get checked: would changing link to be v result in a smaller distance? If so, change link and distance**

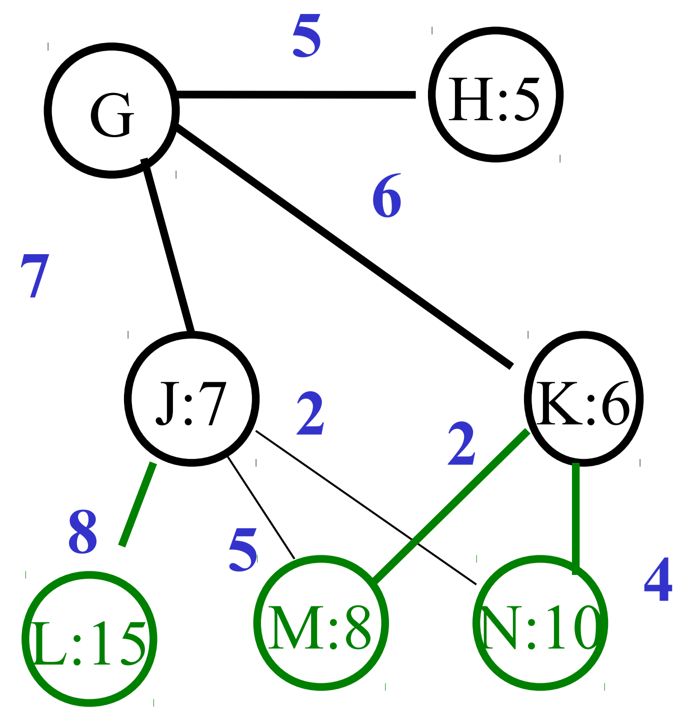
J → tree, Update fringe



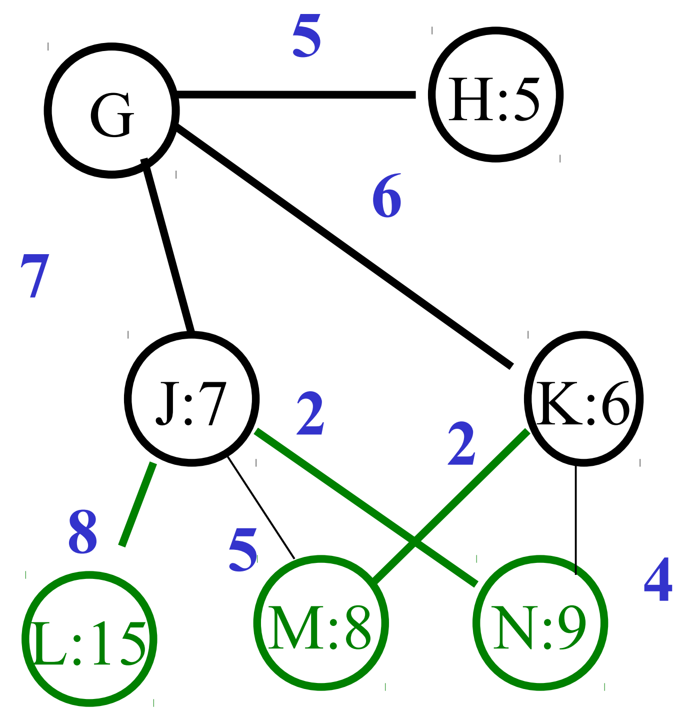
J → tree



Update fringe: neighbors → fringe

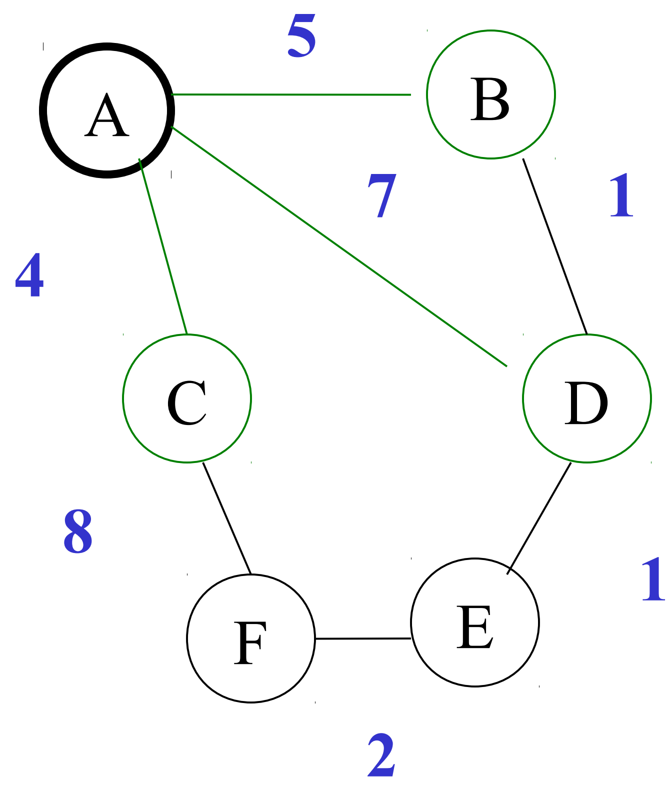


Update fringe: Check neighbors' links



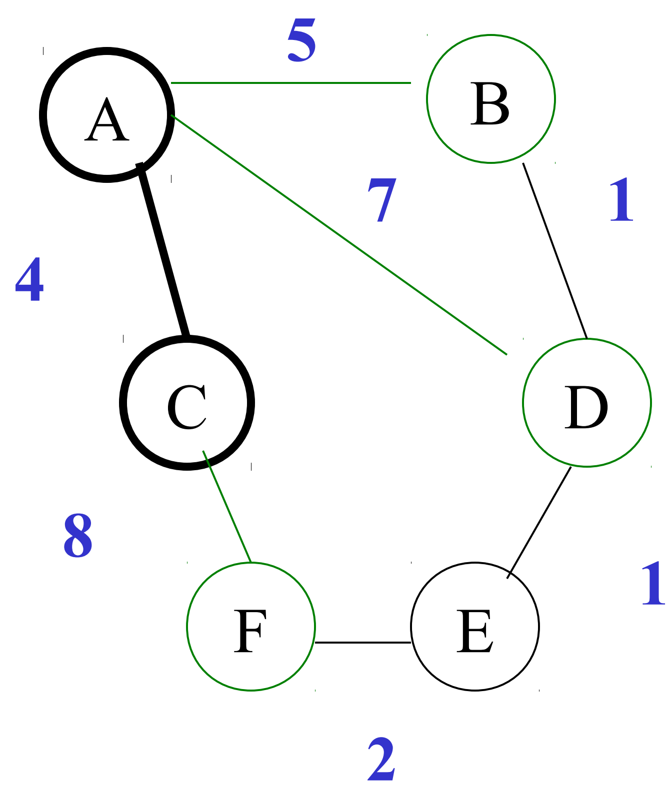
Example

Node	Status	Link	Distance
A	Tree	--	0
B	Fringe	A	5
C	Fringe	A	4
D	Fringe	A	7
E			
F			



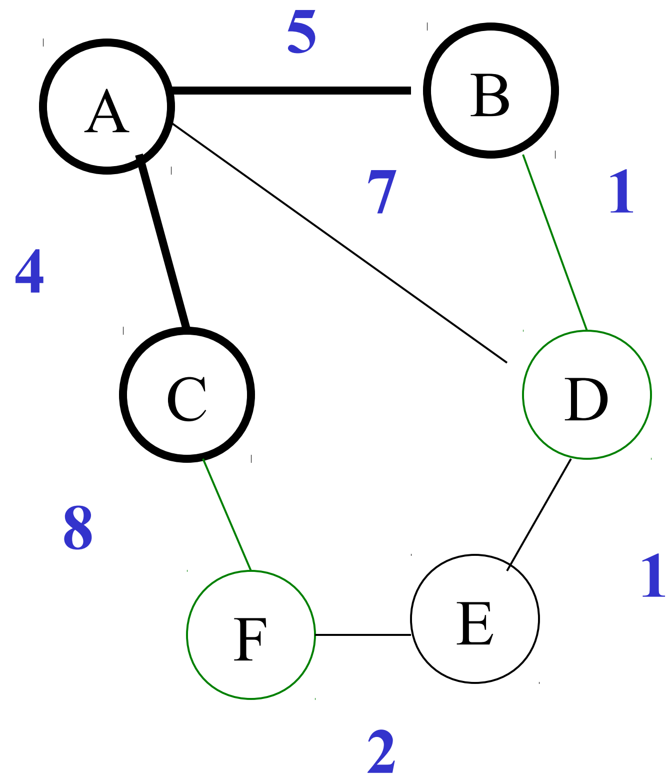
Example

Node	Status	Link	Distance
A	Tree	--	0
B	Fringe	A	5
C	Tree	A	4
D	Fringe	A	7
E			
F	Fringe	C	12



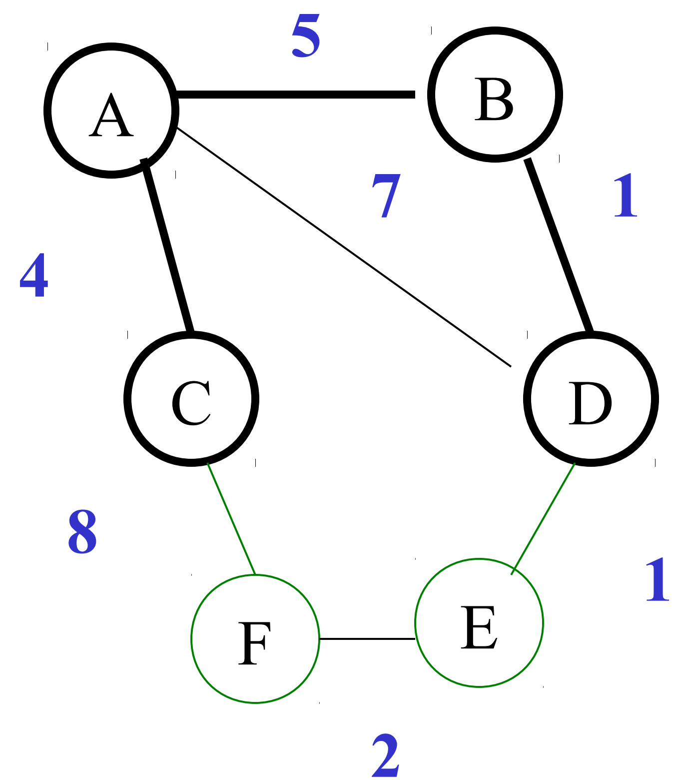
Example

Node	Status	Link	Distance
A	Tree	--	0
B	Tree	A	5
C	Tree	A	4
D	Fringe	B	6
E			
F	Fringe	C	12



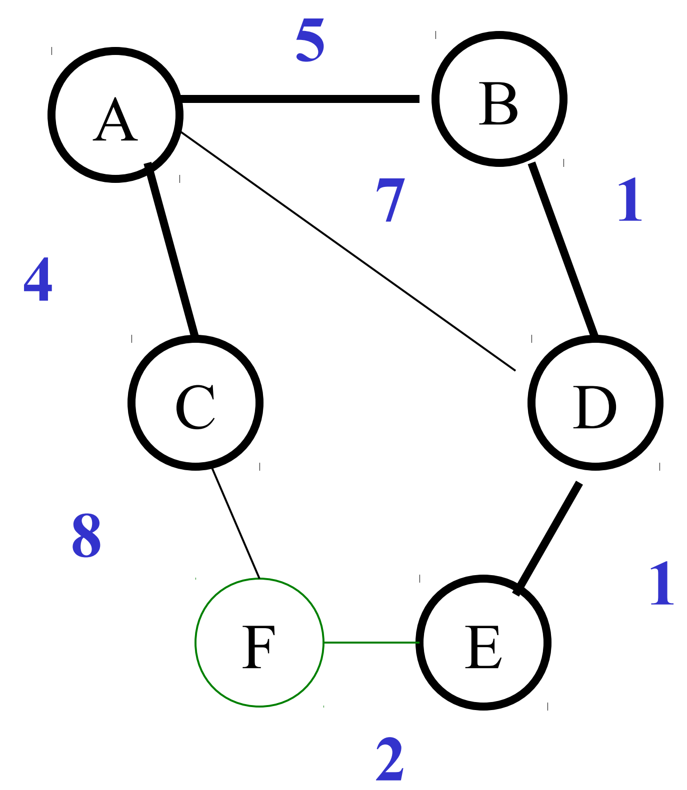
Example

Node	Status	Link	Distance
A	Tree	--	0
B	Tree	A	5
C	Tree	A	4
D	Tree	B	6
E	Fringe	D	7
F	Fringe	C	12



Example

Node	Status	Link	Distance
A	Tree	--	0
B	Tree	A	5
C	Tree	A	4
D	Tree	B	6
E	Tree	D	7
F	Fringe	E	9



Example

Node	Status	Link	Distance
A	Tree	--	0
B	Tree	A	5
C	Tree	A	4
D	Tree	B	6
E	Tree	D	7
F	Tree	E	9