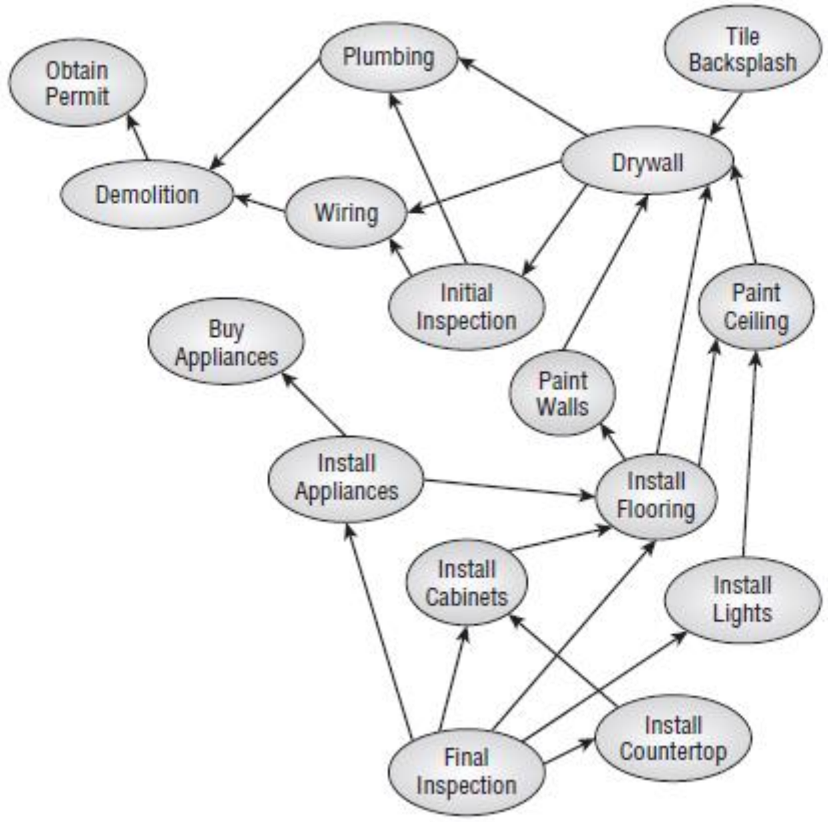
More Network Algorithms



Agenda

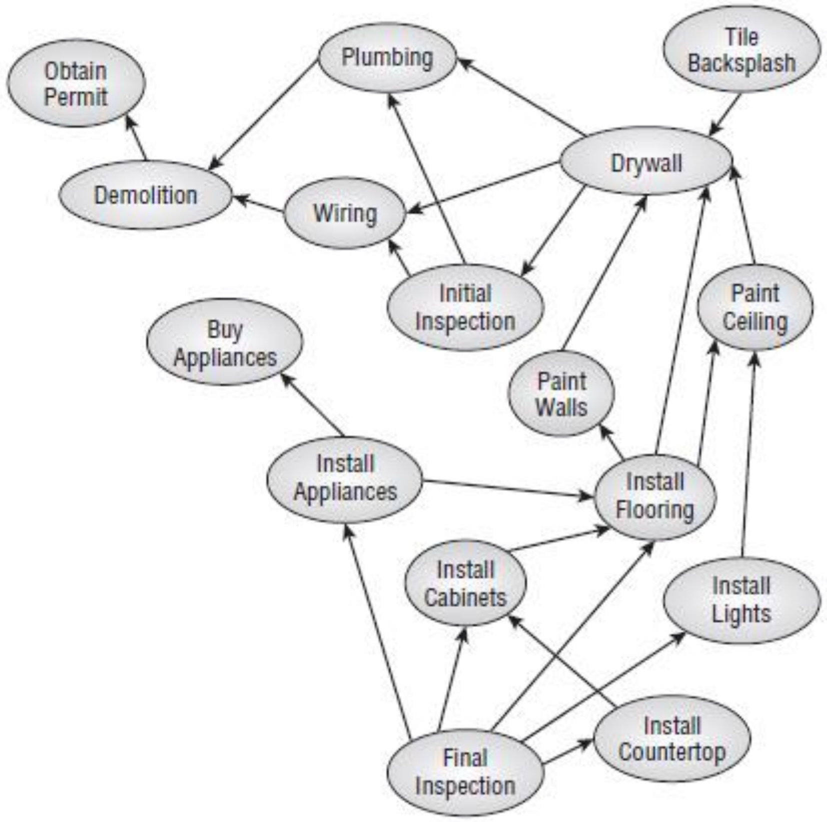
Topological Sorting

- Extend a partial ordering to a complete ordering

Partially Ordered Tasks

TASK	PREREQUISITE
Obtain permit	—
Buy appliances	—
Install appliances	Buy appliances
Demolition	Obtain permit
Wiring	Demolition
Drywall	Wiring
Plumbing	Demolition
Initial inspection	Wiring
Initial inspection	Plumbing
Drywall	Plumbing
Drywall	Initial inspection
Paint walls	Drywall
Paint ceiling	Drywall
Install flooring	Paint walls
Install flooring	Paint ceiling
Final inspection	Install flooring
Tile backsplash	Drywall
Install lights	Paint ceiling
Final inspection	Install lights
Install cabinets	Install flooring
Final inspection	Install cabinets
Install countertop	Install cabinets
Final inspection	Install countertop
Install flooring	Drywall
Install appliances	Install flooring
Final inspection	Install appliances

A Task Network



Basic Algorithm

- Repeat:
  - Find a task with no prerequisites
  - Add it to the ordered task list

The trick is doing this efficiently

Improved Algorithm

- Give each task a `Prerequisites` count
- Give each task an `AfterMe` list
- Add all tasks with no prerequisites to a ready list
- Repeat:
  - Remove a task `Ready` from the ready list
  - Add it to the ordered task list
  - For each task `After` in `Ready.AfterMe`:
    - Decrement `After.Prerequisites`
    - If `After.Prerequisites == 0`, add After to the ready list

Cycle Detection

- This follows immediately from topological sorting

```
// Return True if the network contains a cycle.
Boolean: ContainsCycle()
    // Try to topologically sort the network.
    If (ExtendPartialOrdering() == null) Then Return True
    Return False
End ContainsCycle
```

Map Coloring

- Color a map so no two adjacent regions share the same color

- Two-coloring is easy (if possible)
- Three-coloring is very hard
- Four-coloring is possible but hard
- Five-coloring is complicated but fast

Two-Coloring

- Color any starting region
- Repeat:
  - Color a region adjacent to an already colored region

If a two-coloring is possible, this method will find it

Network Simplification

- If a node N has fewer than three neighbors:
  - Remove N from the network
  - Color the rest of the network
  - Restore N and color it

Three-Coloring

- No known polynomial time solution
- Obvious approach:
  - Try all possible combinations in $O(3^N)$ time
- Network simplification:
  - If a node has fewer than t

Four-Coloring

- *Four-coloring theorem*: any planar map can be colored with at most four colors.
- First proposed by Francis Guthrie in 1852
- Studied extensively for 124 years
- Proven by Kenneth Appel and Wolfgang Haken in 1976
- Requires a computer to exhaustively examined a set of 1,936 specially selected maps

Four-Coloring Algorithms

- The four-coloring theorem doesn’t give you a good way to find a four-coloring
- Try all possible combinations in $O(4^N)$ time
- Use network simplification methods

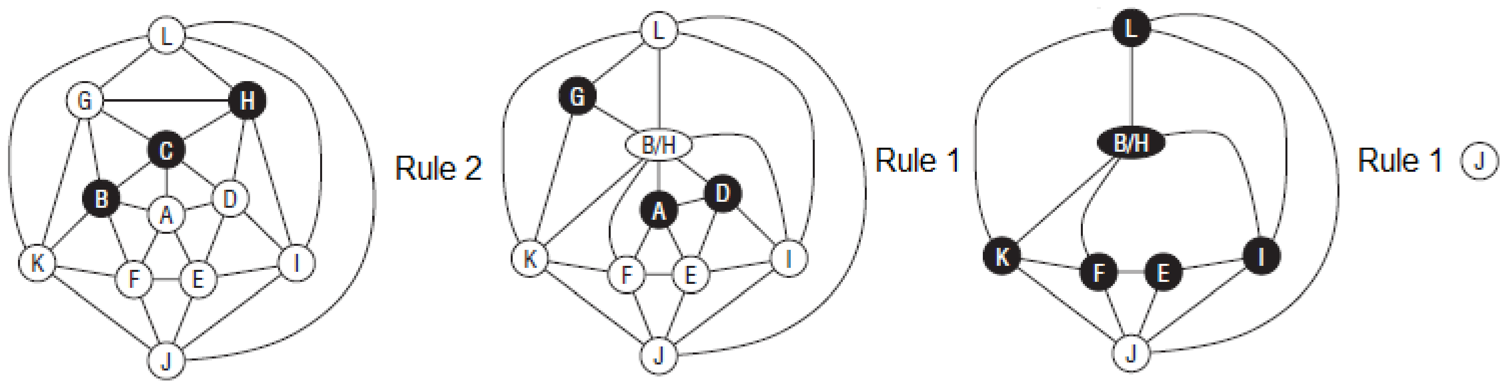
Five-Coloring

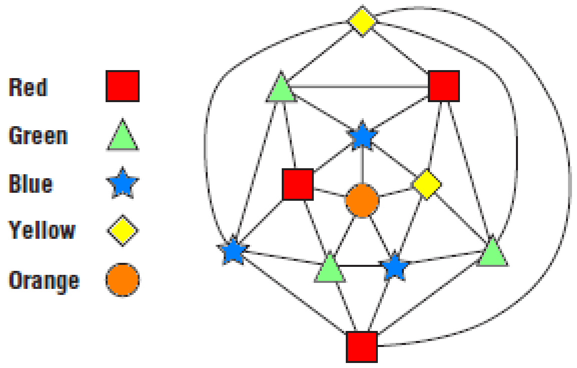
- If the network has a node with fewer than five neighbors, use network simplification
- Otherwise:
  - It can be shown that there is at least one node K with neighbors M and N where:
    - K has exactly five neighbors
    - M and N have at most seven neighbors
    - M and N are not neighbors of each other
  - Find K, M, and N, and give M and N the same color
  - Give K the color that is not used by its neighbors

Five-Coloring Algorithm

1. Repeat while the network has more than one node:
   a. If there is a node with degree less than 5, remove it
   b. If the network contains no node of degree less than 5, find a node K with degree exactly 5 and two children M and N, as described earlier. Remove nodes K, M, and N, and create the new node M/N.
2. When the network contains a single node, assign it a color.
3. Restore the previously removed nodes, coloring them appropriately.

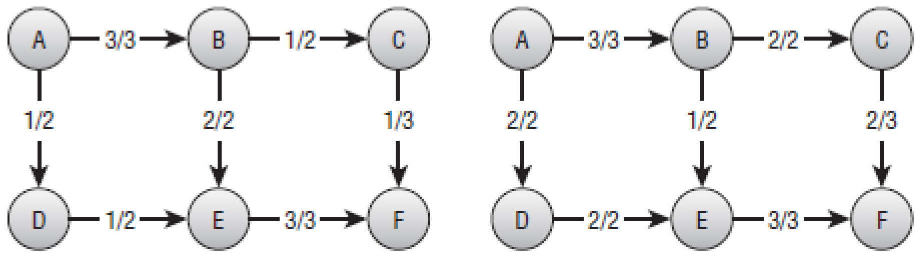
Five-Coloring Example





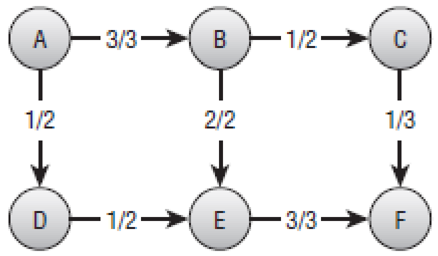
Maximal Flow

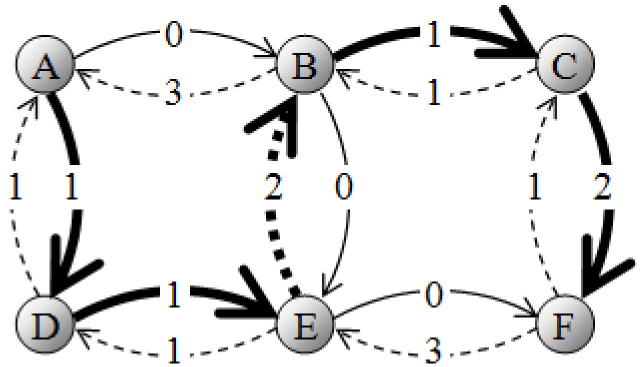
- Capacitated network



Residual Capacity Network

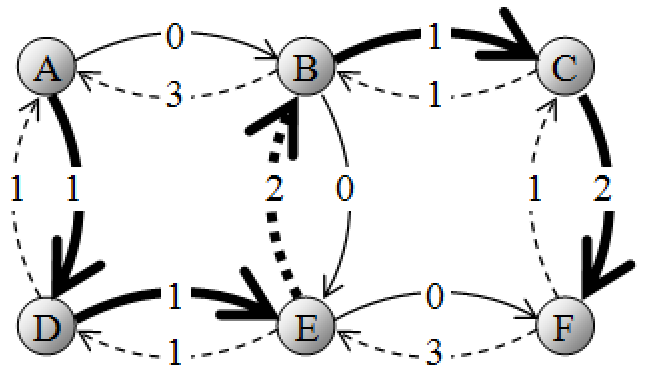
- Additional flow that could be added
- Existing flow that could be removed

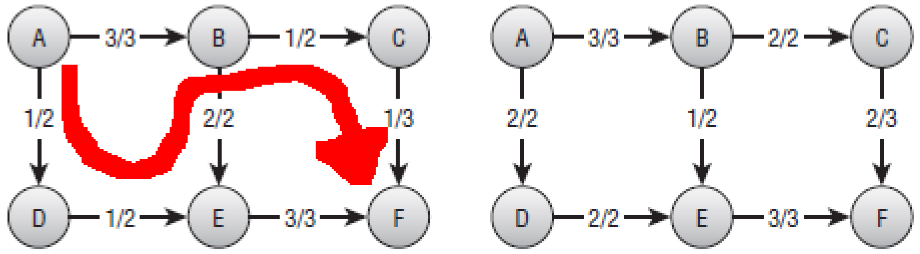




Augmenting Path

- A path through the residual capacity network





Ford-Fulkerson Algorithm

1. Repeat as long as you can find an augmenting path through the residual capacity network:
    a. Find an augmenting path from the source node to the sink node
    b. Follow the augmenting path, and find the smallest residual capacity along it
    c. Follow the augmenting path again, and update the links' flows to correct the augmenting path

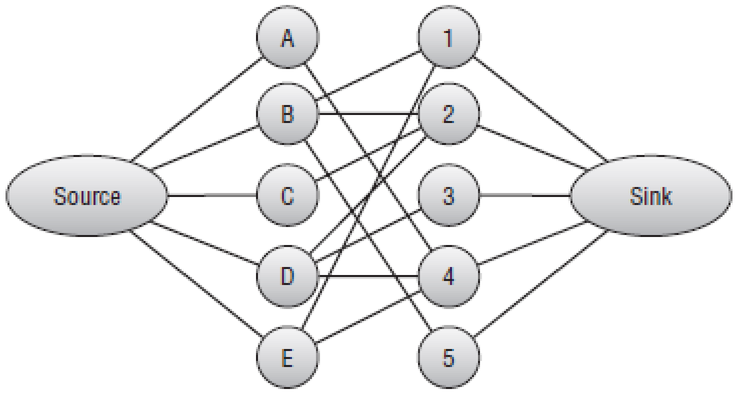
Work Assignment

- Suppose you have 100 employees and 100 jobs
- Each employee can only work certain jobs
- What is the maximum number of jobs you can assign?

- Trying all combinations gives $100! \approx 9.3 \times 10^{157}$ possibilities

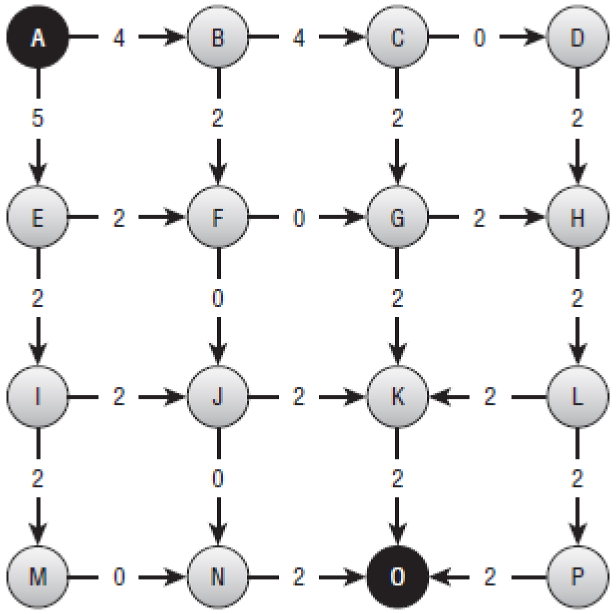
Work Assignment Networks

- Make a work assignment network
- Calculate maximal flow

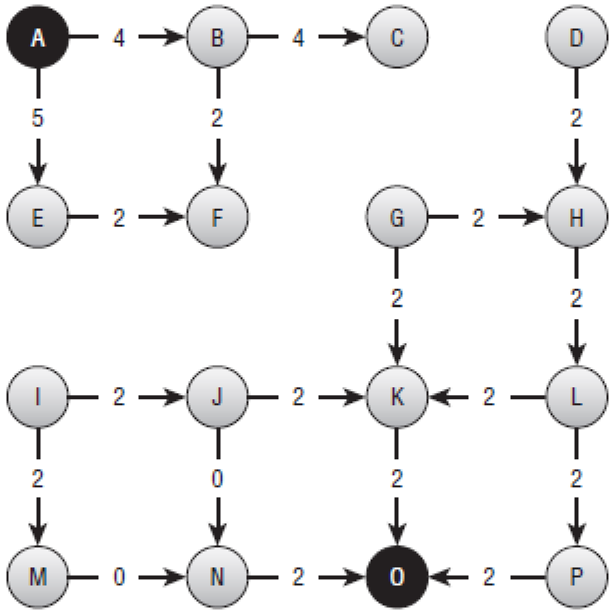


Minimal Flow Cut

- Find the best (minimal cost) links to remove to separate node A from node O



Minimal Flow Solution



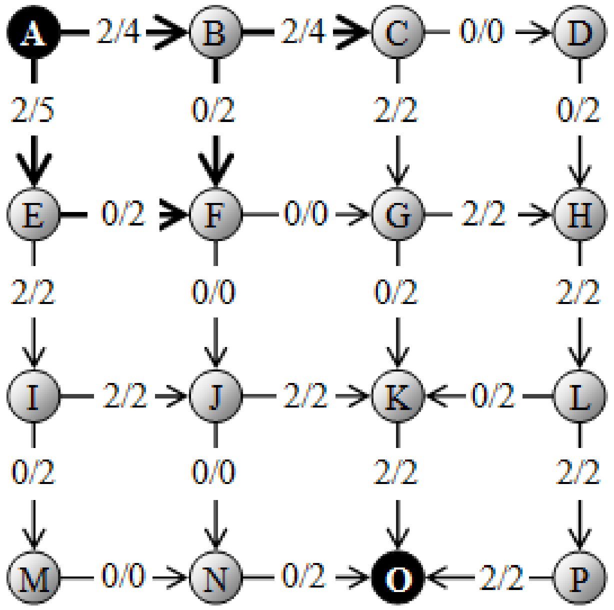
Combinatoric Approach

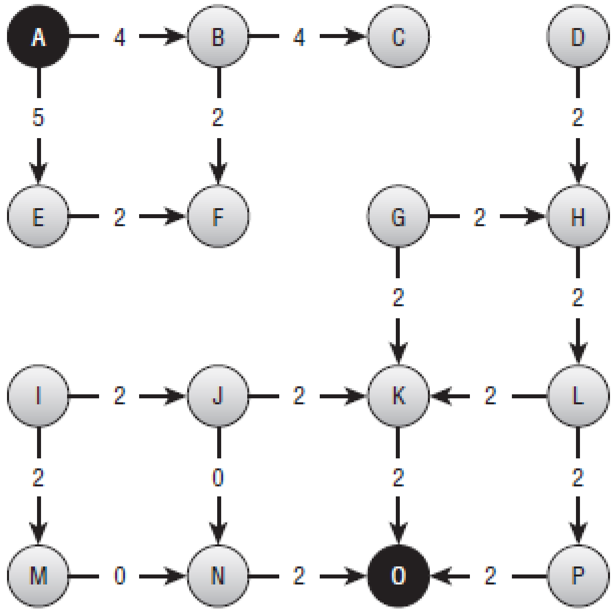
- If there are N links, there are 2^N combinations
- In the previous network, N = 24 so there are $2^{24} \approx 16.8$ million possibilities

Minimal Flow Cut Algorithm

1. Perform a maximal flow calculation between the source and sink nodes
2. Starting from the sink node, visit all the nodes you can using only links and backlinks that have residual capacities greater than 0
3. Place all the nodes visited in Step 2 in set A and all the other nodes in set B
4. Remove links that lead from nodes in set A to nodes in set B

Minimal Flow Solution





Why This Works

- The cut must separate the source and sink
- If F is the maximal flow, then any cut must have total flow at least F
- Suppose flow moves back and forth across the cut. Suppose L moves from set B to set A. Then L’s backlink has residual capacity > 0.

Summary

- Topological Sorting
- Cycle Detection
- Map Coloring
- Maximal Flow
- Work Assignment
- Minimal Flow Cut

Exercises

- Chapter 14 Exercises 1 – 6, 9, 10, 12 – 14.
- Bonus: 7, 8, 11, 15, 16.
- Read *Essential Algorithms, 2e* Chapter 14 pages 470 – 492. (The rest of Chapter 14.)