Network Flow And Special Graphs

COMP 4951: Advanced Problem Solving
Micah Stairs

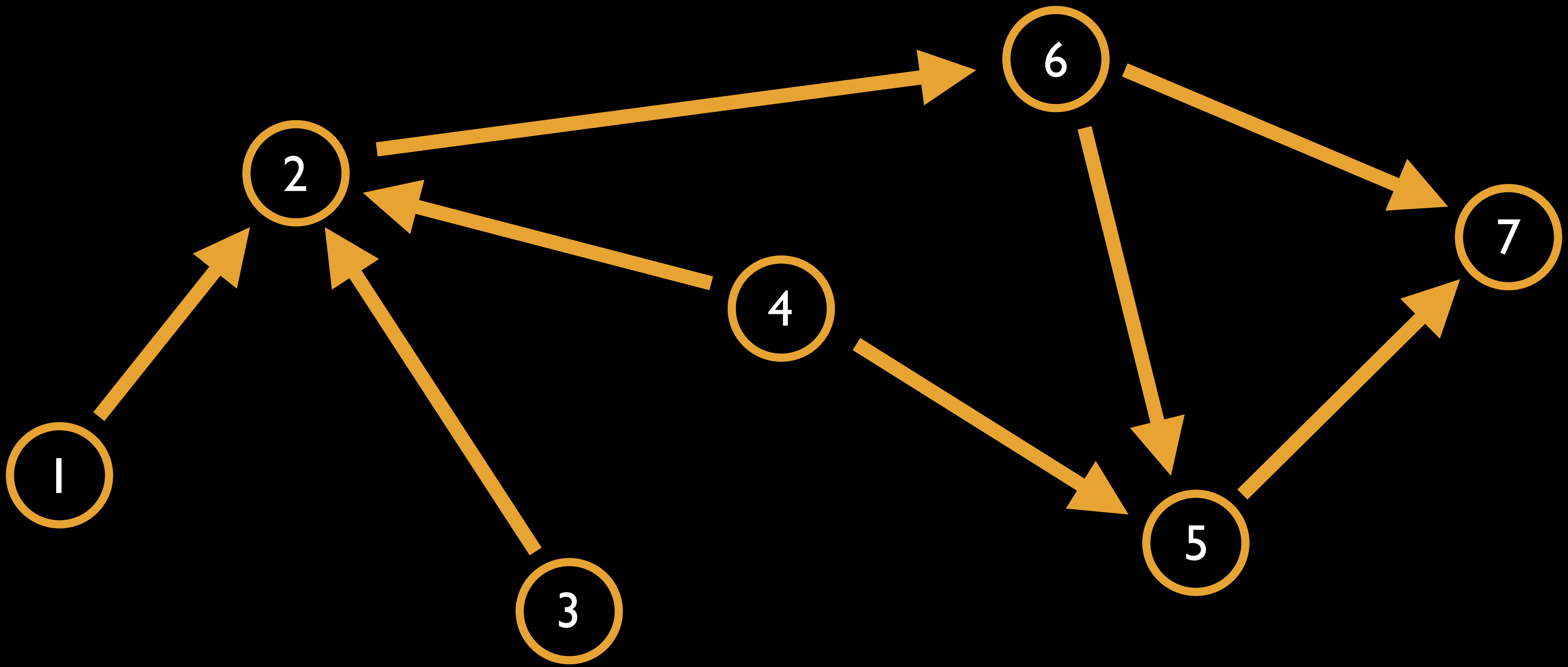
Outline

- Special Graphs + Problems
- Network Flow + Algorithms
- Easy Network Flow Problems
- Bipartite Matching + Problems
- Min Cost Max Flow + Problems
- Hard Network Flow Problems
- Further Network Flow Variants
- Conclusion
- Collection of Relevant Problems
- Sources (no pun intended)

Directed Acyclic Graphs (DAGs)

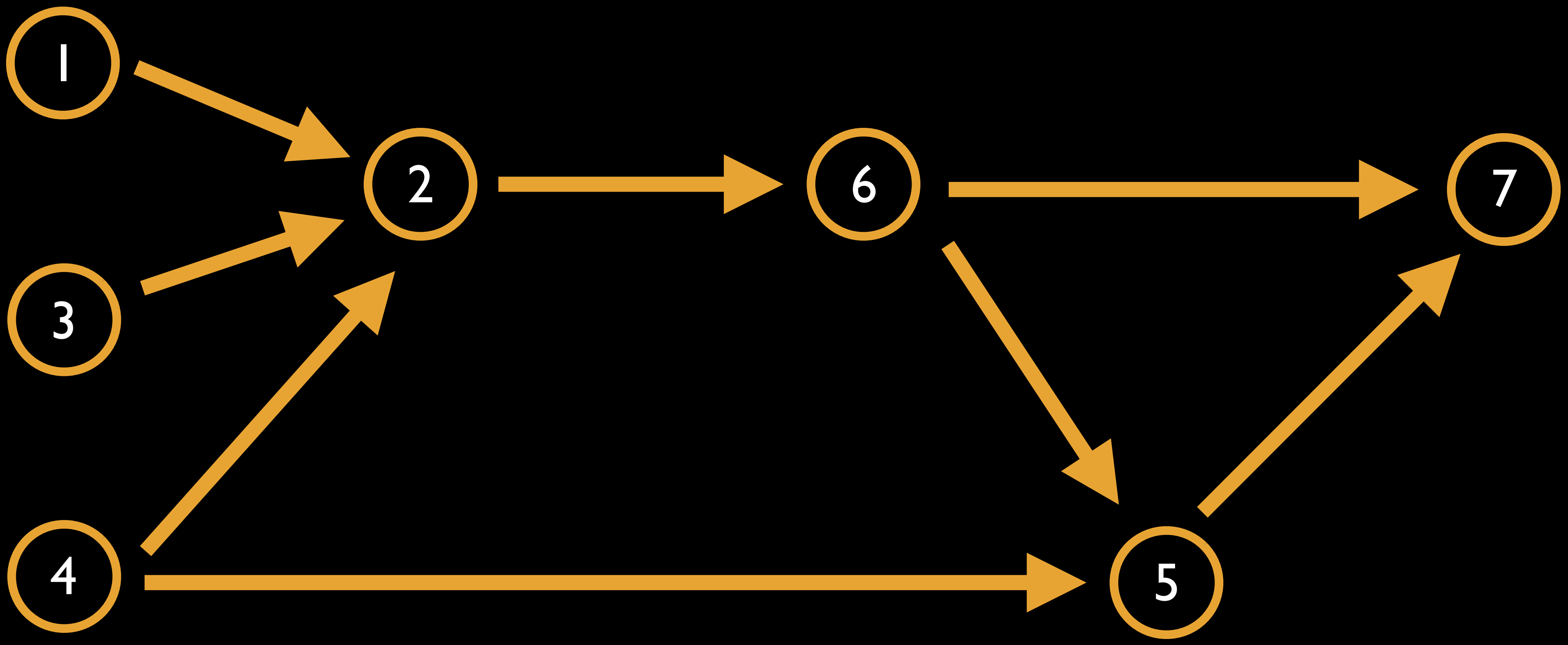
DAG

Definition: A directed graph which has no cycles.



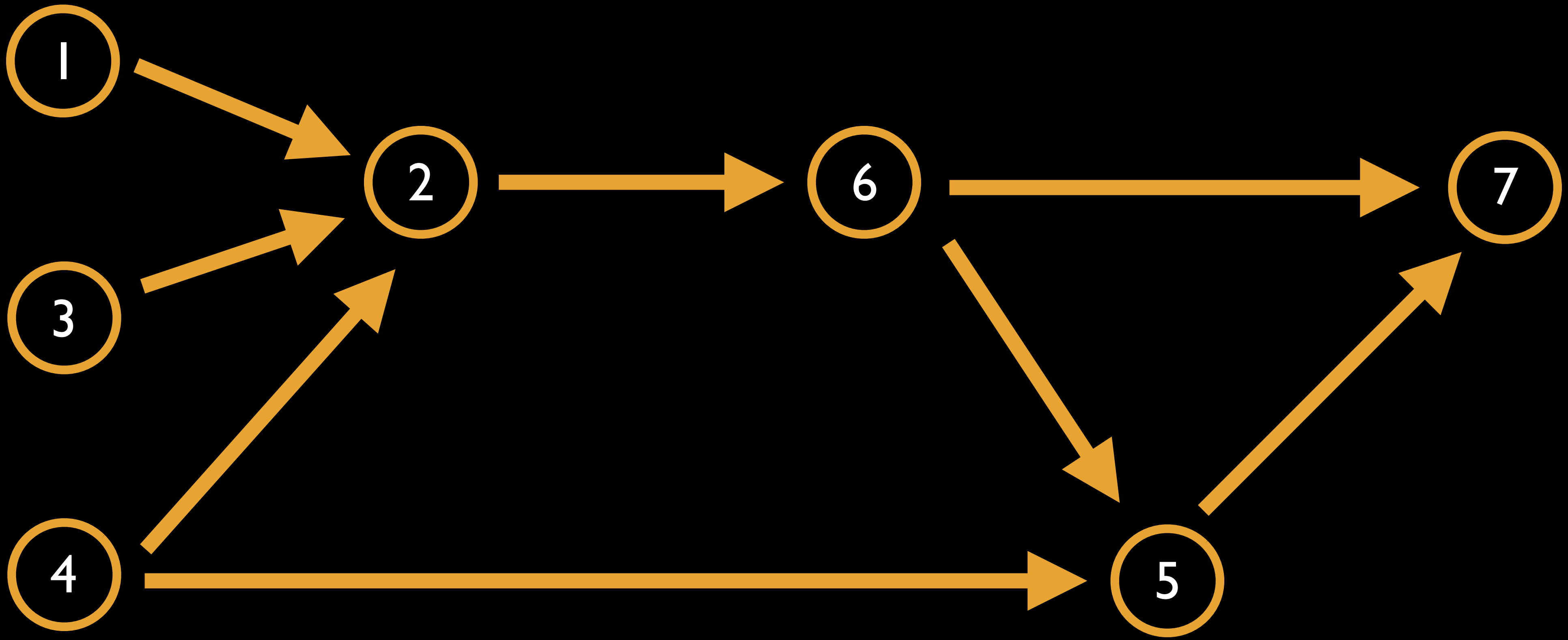
DAG

Always has at least one topological order.



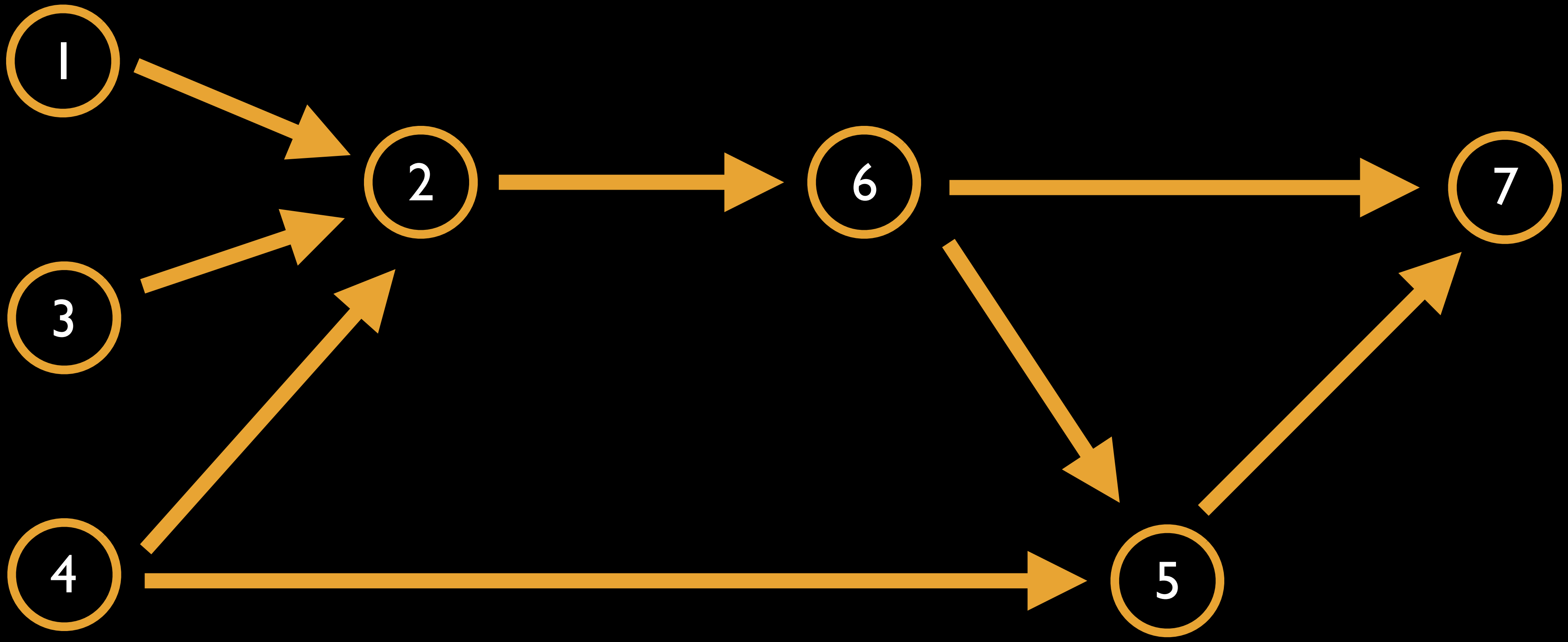
DAG

This topological order tells us what order we should process the nodes.



DAG

Leads to O(V+E) solutions for many types of problems (shortest path, longest path, counting paths, etc.).



Sample Problems

Build Dependencies: Requires you to build the directed graph from the input and output a valid topological sort.

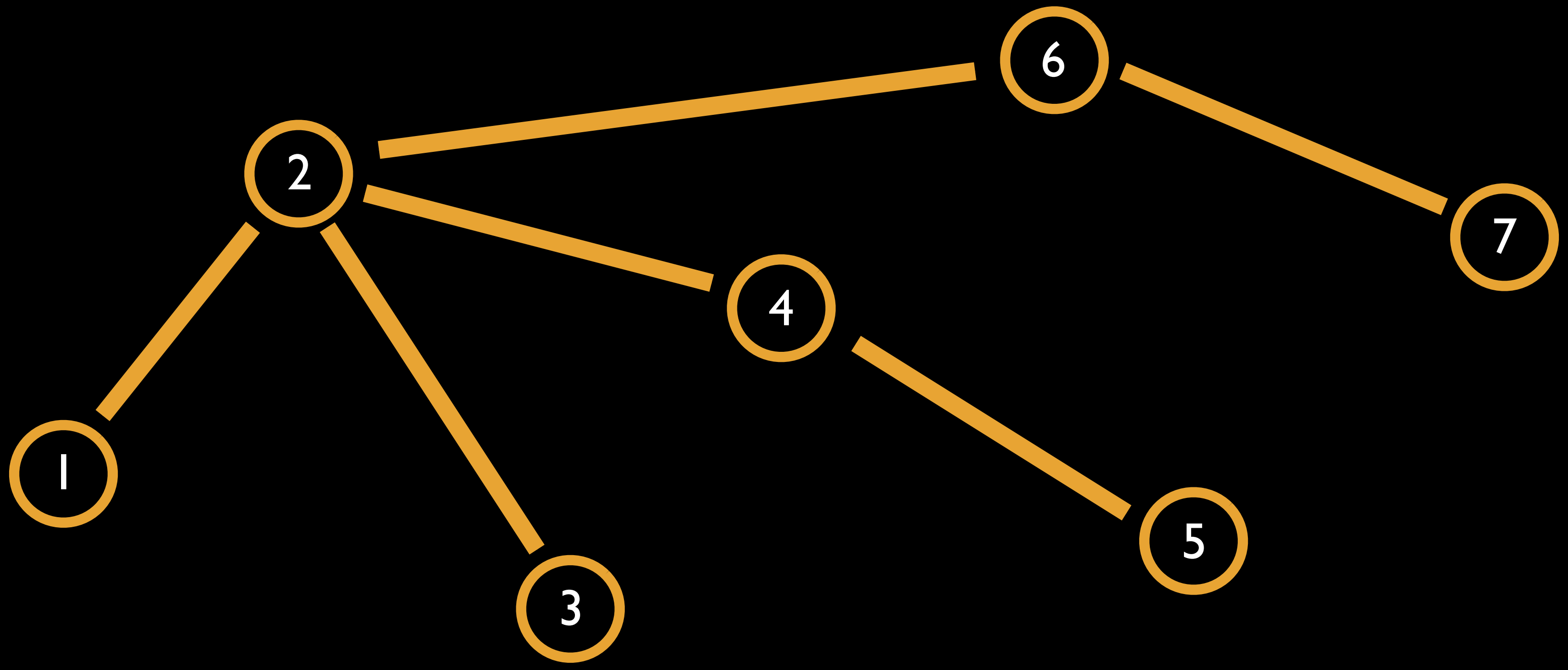
Barica: Given a set of nodes on a 2D plane and restrictions on the movement between these nodes (giving a DAG). Using a variant of the longest path problem, requires you to output one of these paths.

Trees

Tree

Definition: A connected, undirected graph with the following <u>equivalent</u> properties:

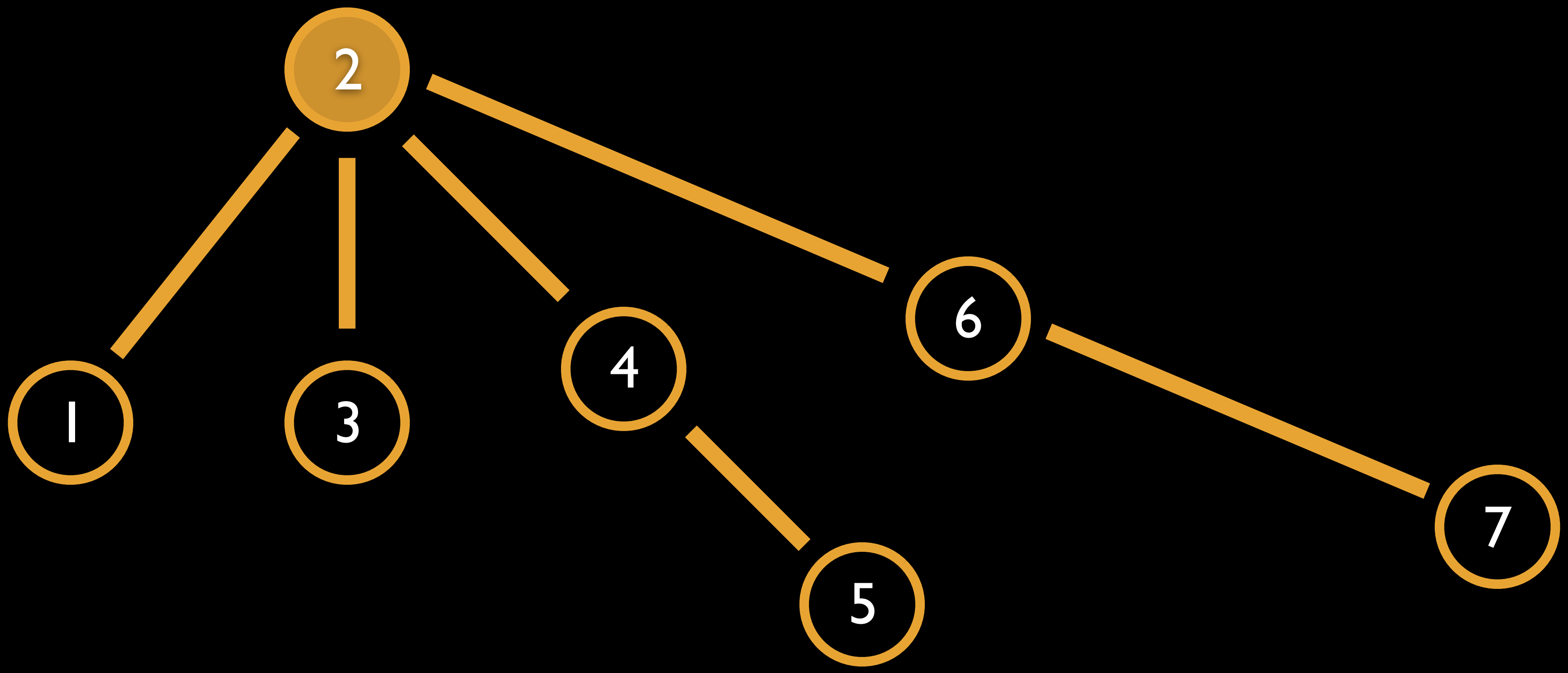
- *n* vertices, *n-1* edges
- No cycles
- Unique path between each pair of nodes
- Each edge is a bridge



Tree

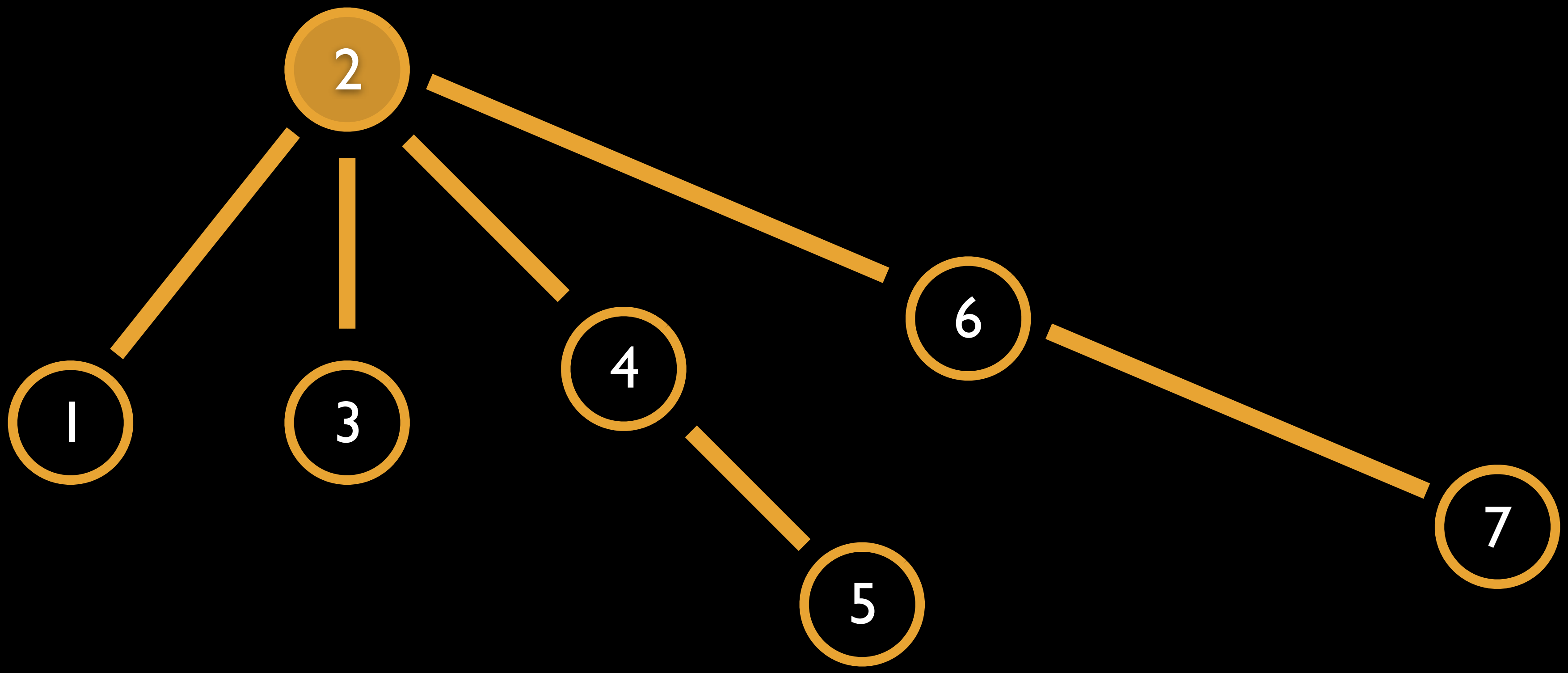
In some cases we are given a “rooted tree”. Otherwise we can arbitrarily choose a root.

Note: If working with a rooted tree, it may be easiest to represent it using a directed graph.



Tree

Since trees do not have any cycles, it allows many algorithms to run in O(V) time (traversals, computing subtree sizes/depths, finding diameter, etc.).



Sample Problems

Genealogical Research: Requires to build and traverse a tree.

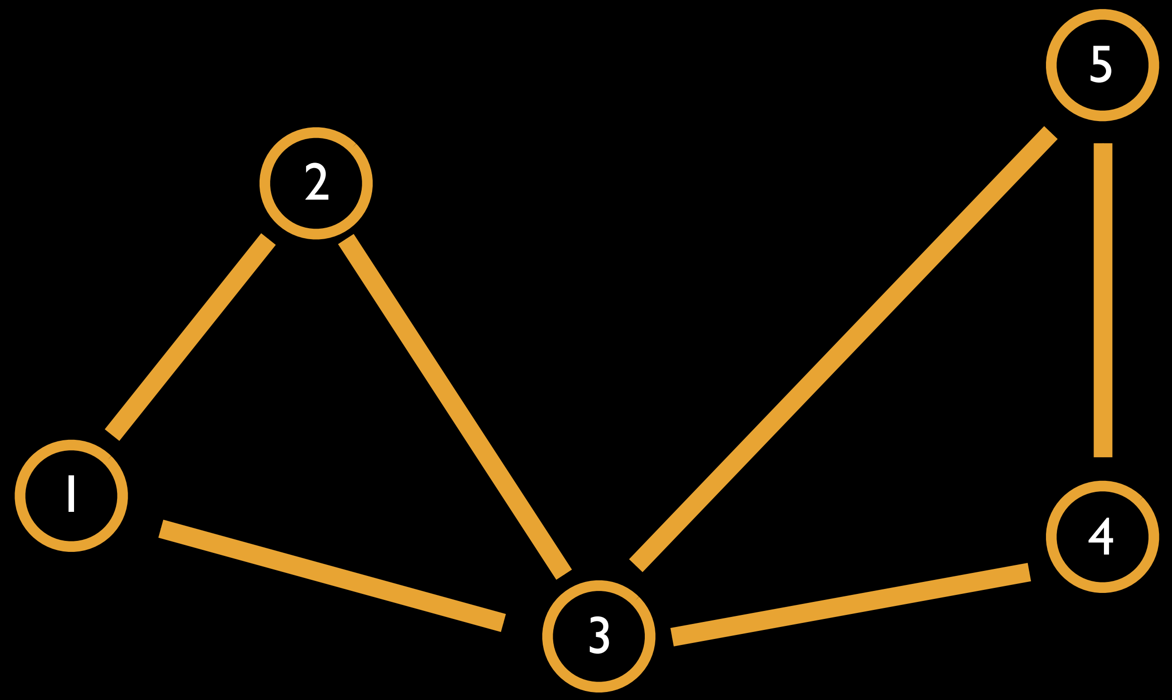
Kitten on a Tree: Requires you to build a tree and traverse it backwards, outputting the path to the root.

Eulerian Graphs

Eulerian Graph

Definition: A graph which has either a Eulerian path or Eulerian circuit (they are mutually exclusive).

<u>Note</u>: The graph can be either directed or undirected.



[3, 1, 2, 3, 5, 4, 3] is a Eulerian circuit in this graph since each edge is visited and it starts and ends on the same node.

Eulerian Graph

Checking if a graph has a Eularian path/circuit can be done simply by counting the degree of each node.

Actually finding a Eulerian path or circuit is more difficult to implement but can still be done in O(V+E).

Sample Problems

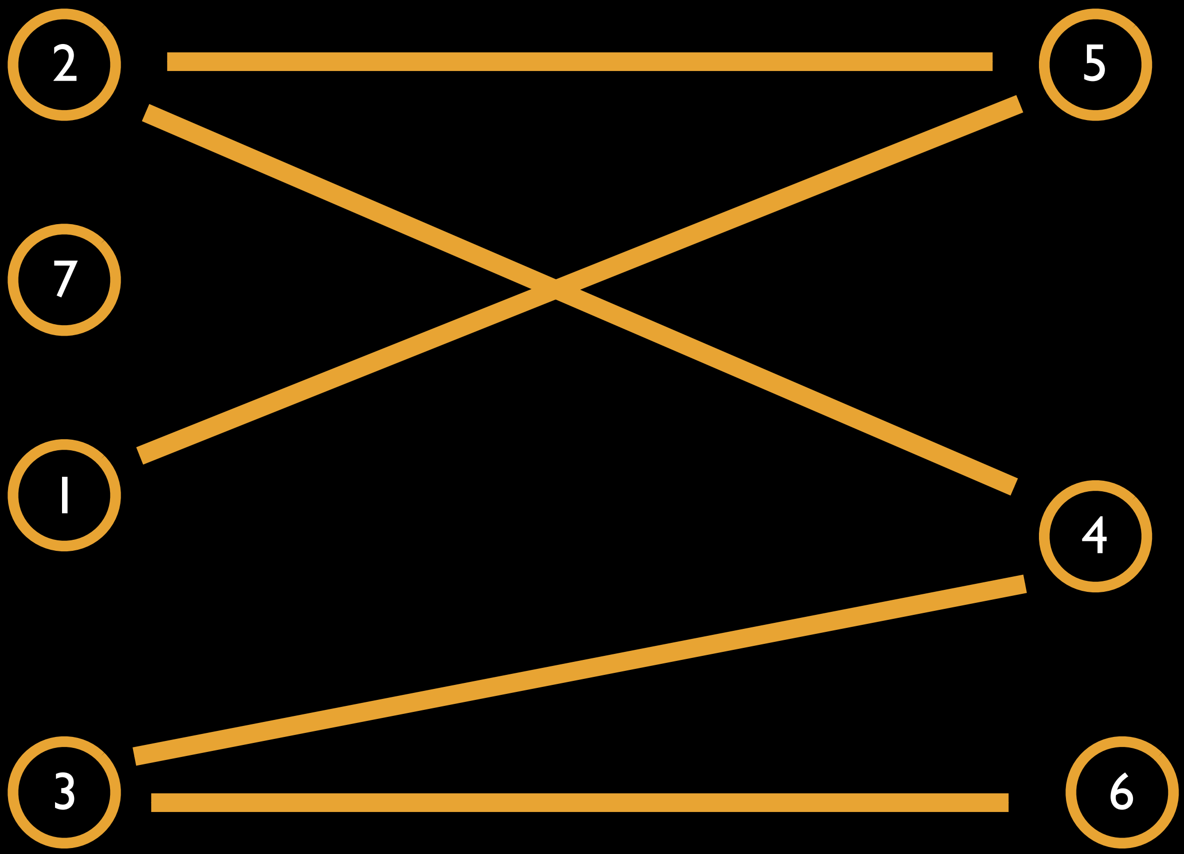
Eulerian Path: Find and output a Eulerian path.

Catenyms: Infer graph from input where we have one node for each letter in the alphabet, and one edge for each word (from the first letter of the word to the last letter). Requires you to then find a Eulerian path.

Bipartite Graphs

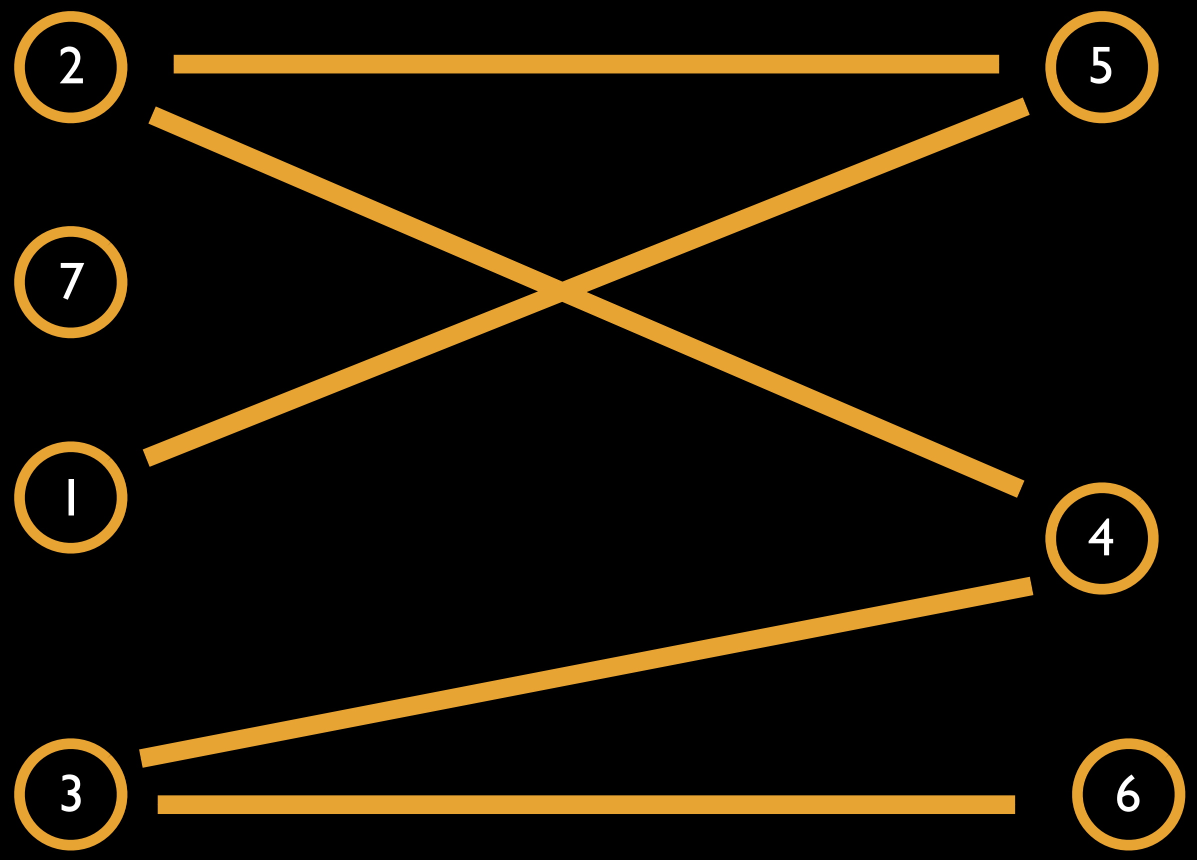
Bipartite Graph

Definition: An undirected graph which has no odd length cycles.



Bipartite Graph

Checking to see if a graph is bipartite can be done in O(V+E).



Sample Problems

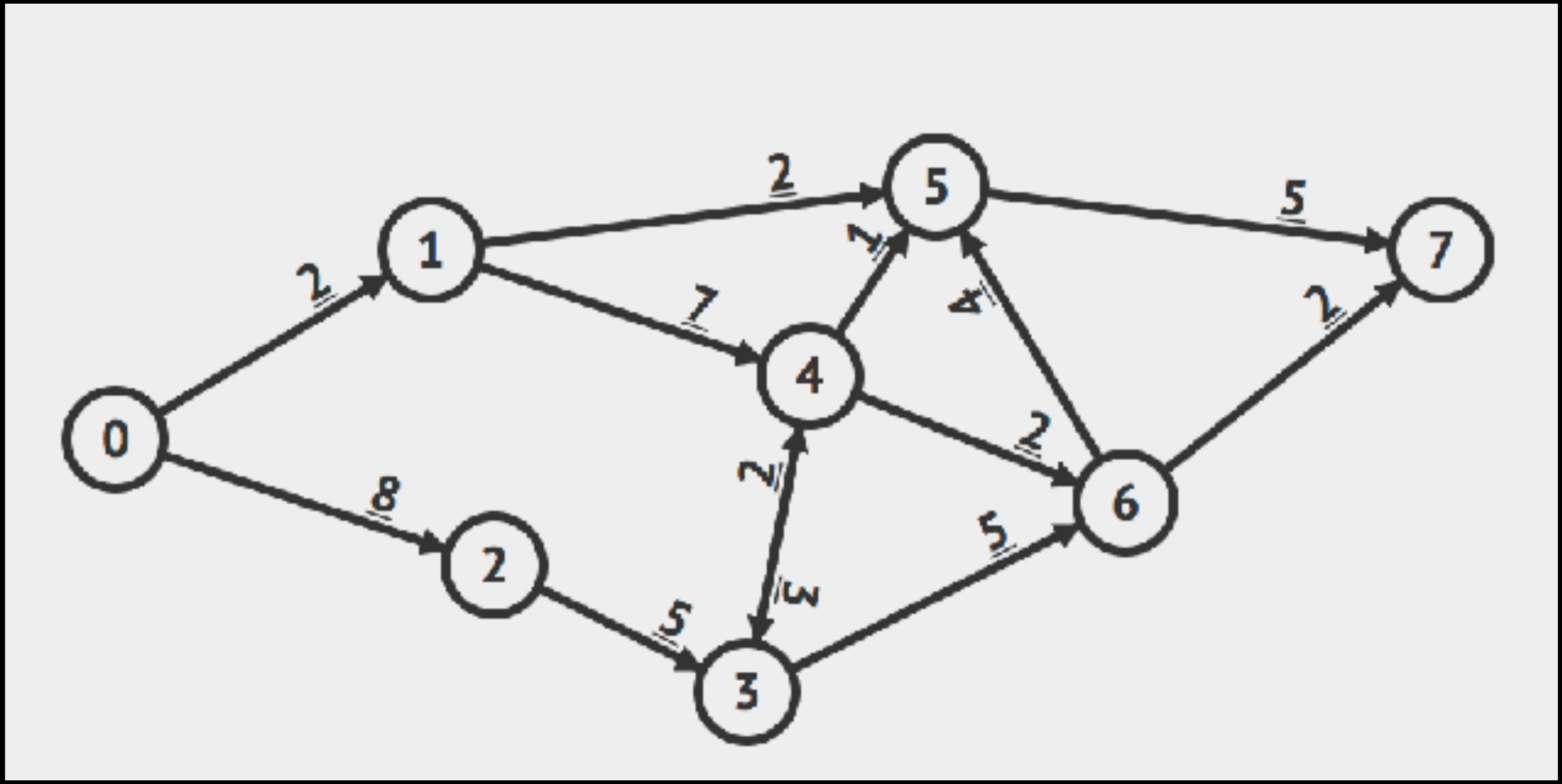
Breaking Bad: Given many pairs of items which are not allowed to be purchased by the same person. Asked if it possible for two people to purchase all of the items.

Amanda Lounges: Solution involves checking to see if a graph is bipartite.

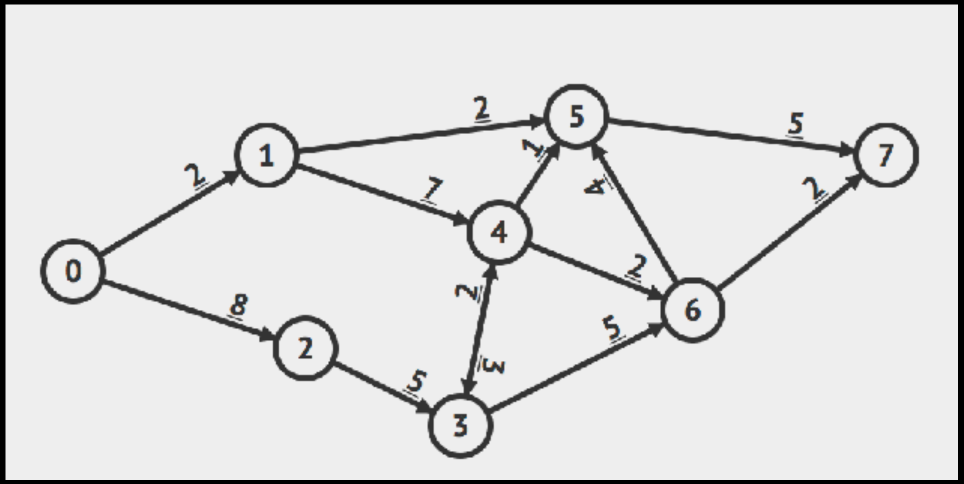
Network Flow

Network Flow

Given a network of pipes, each with a certain capacity, and source and sink nodes, we want to find the maximum flow that can pass through the network.



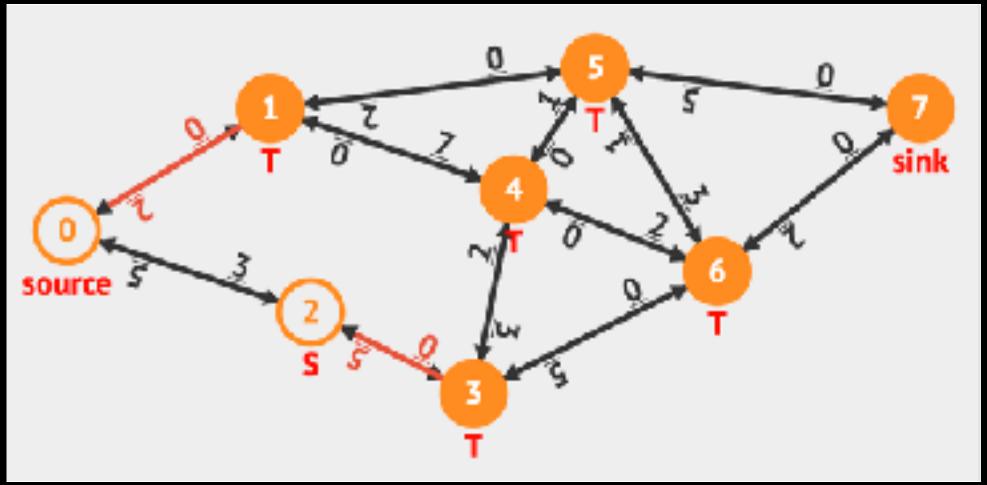
Network Flow



Step 1: Build initial graph.

…

Step 2: Run a Network Flow algorithm.

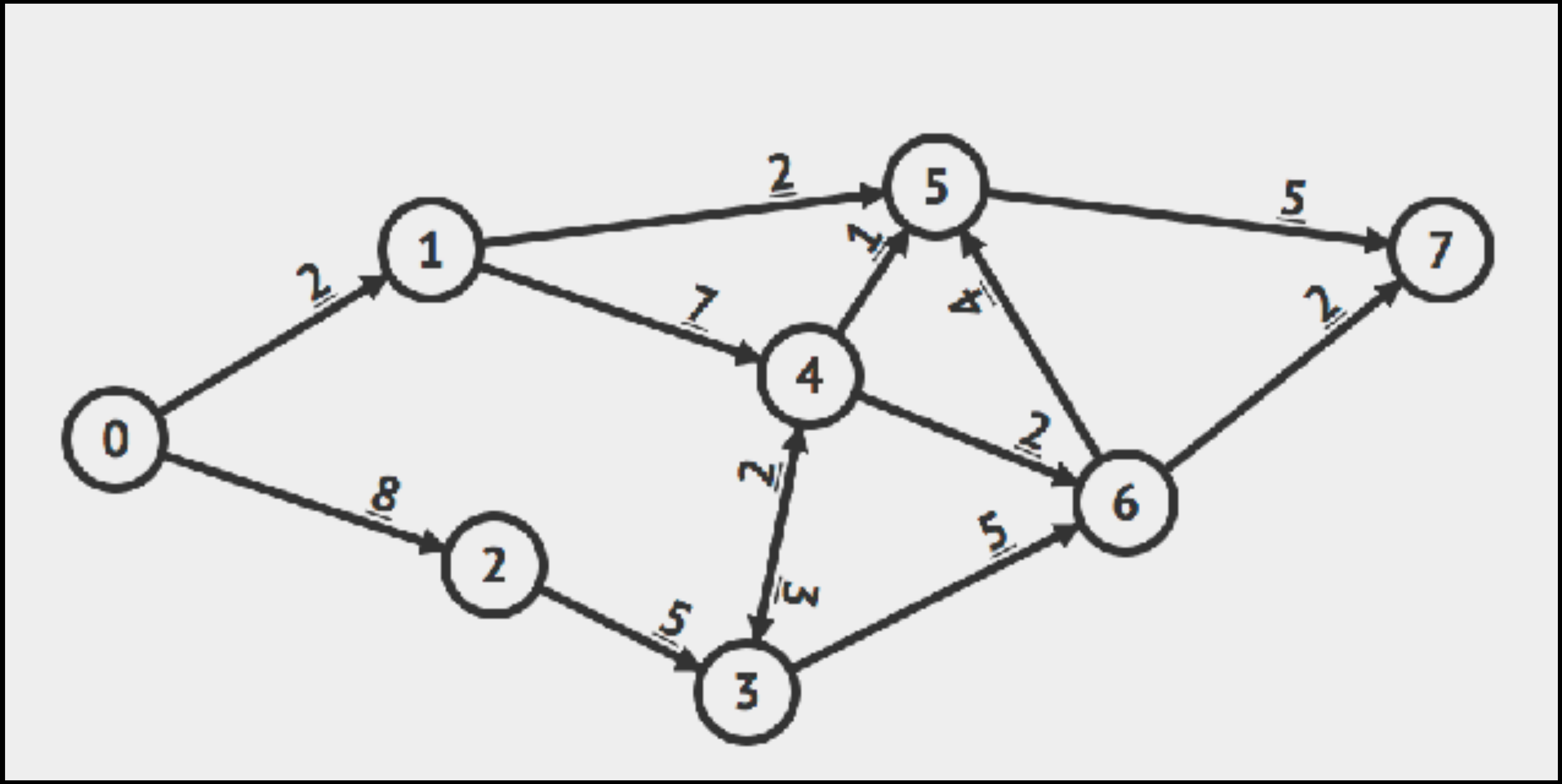


Step 3: End up with max flow and min cut.

Ford-Fulkerson

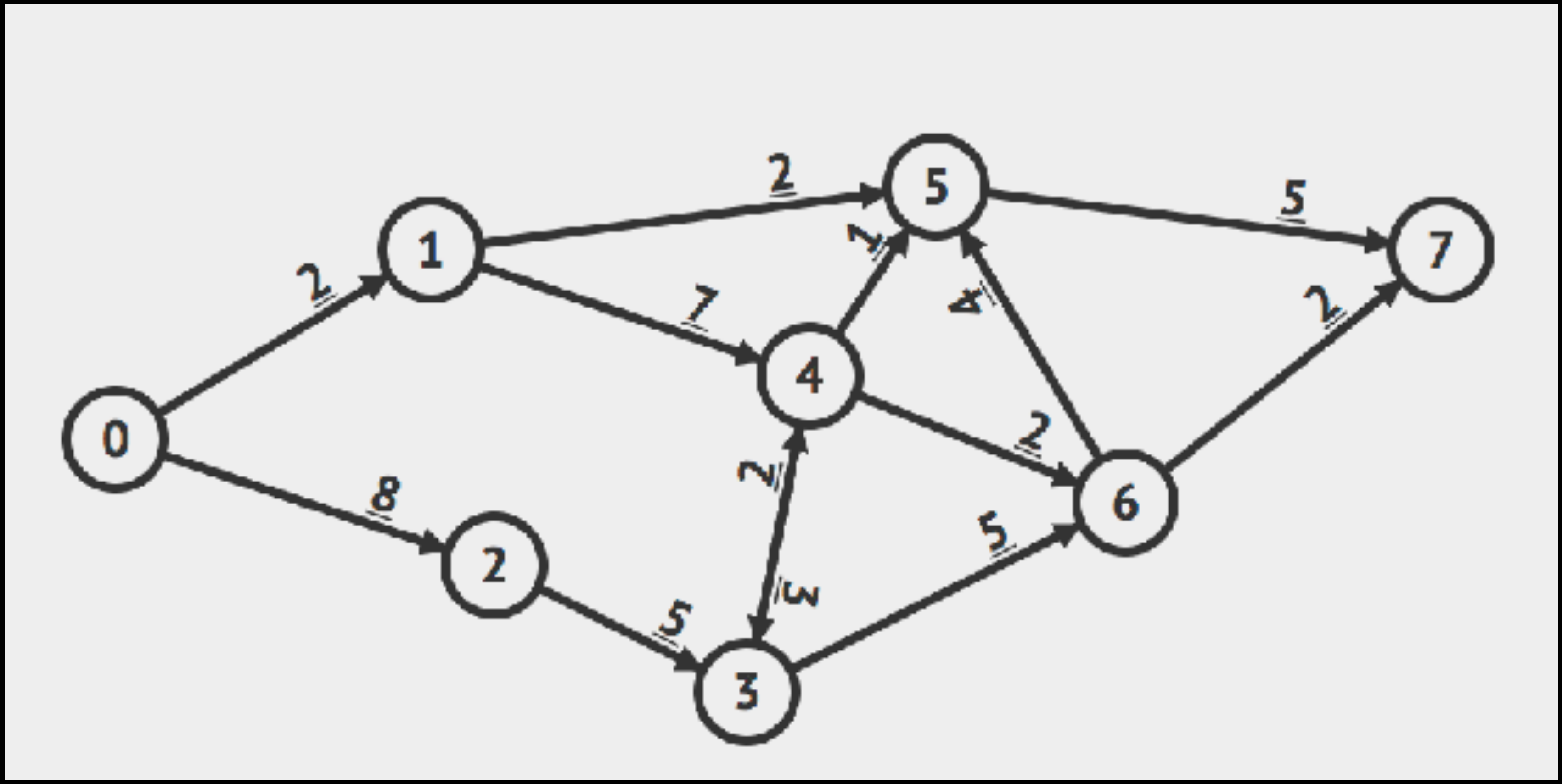
Ford-Fulkerson

Finds the maximum flow by repeatedly finding paths which can push some flow from the source to the sink.



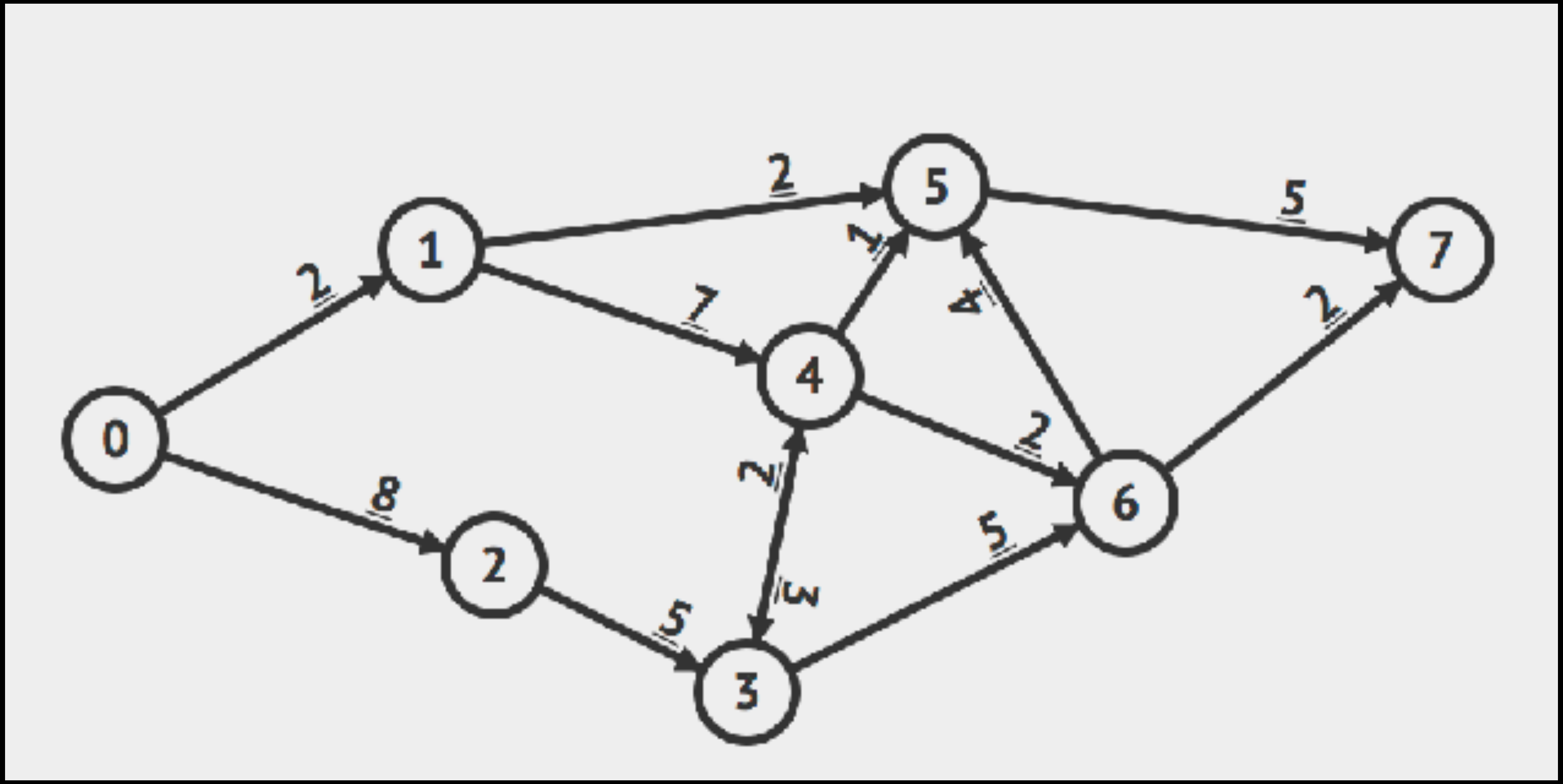
Ford-Fulkerson

Each iteration pushes at least 1 unit of flow through the network (assuming integer capacities), so the time complexity if O(Ef), where *f* is the maximum flow.



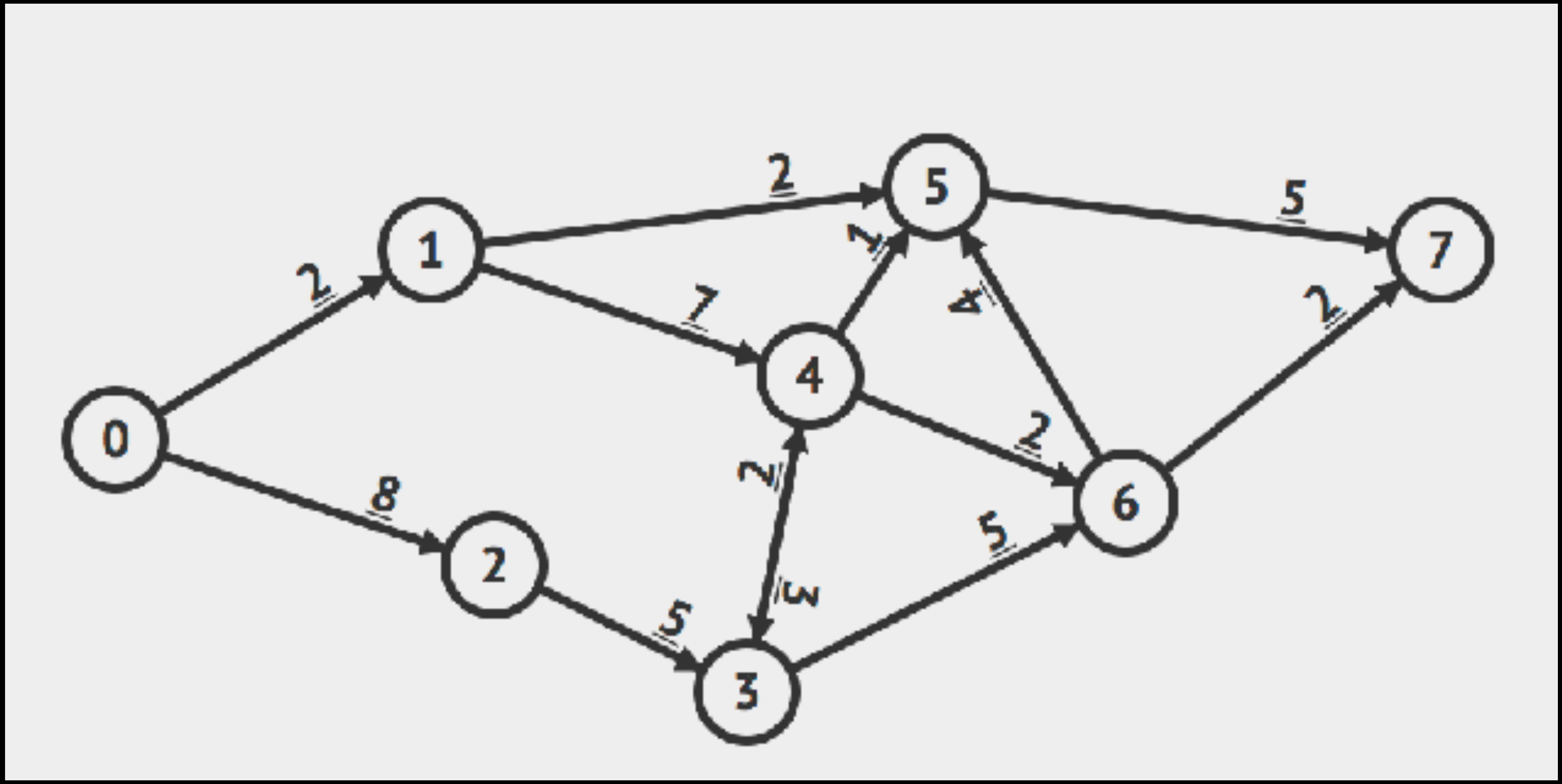
Ford-Fulkerson

Ford-Fulkerson is sometimes referred to as a "method" instead of an "algorithm" since the method used to find the paths is unspecified.



Ford-Fulkerson

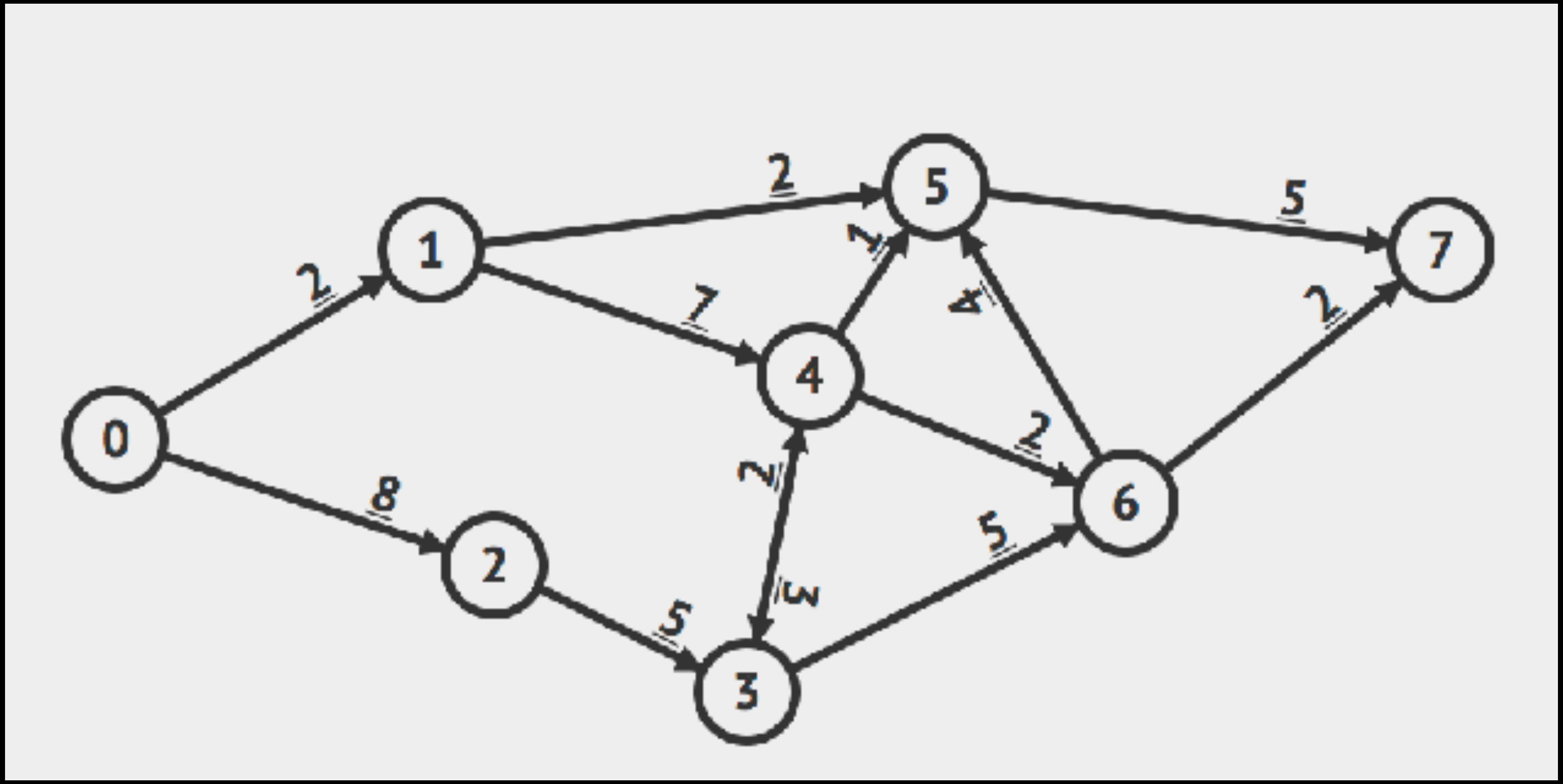
For the rest of this presentation, when we talk about Ford-Fulkerson, we will assume that we are simply using a DFS to find the paths.



Edmonds-Karp

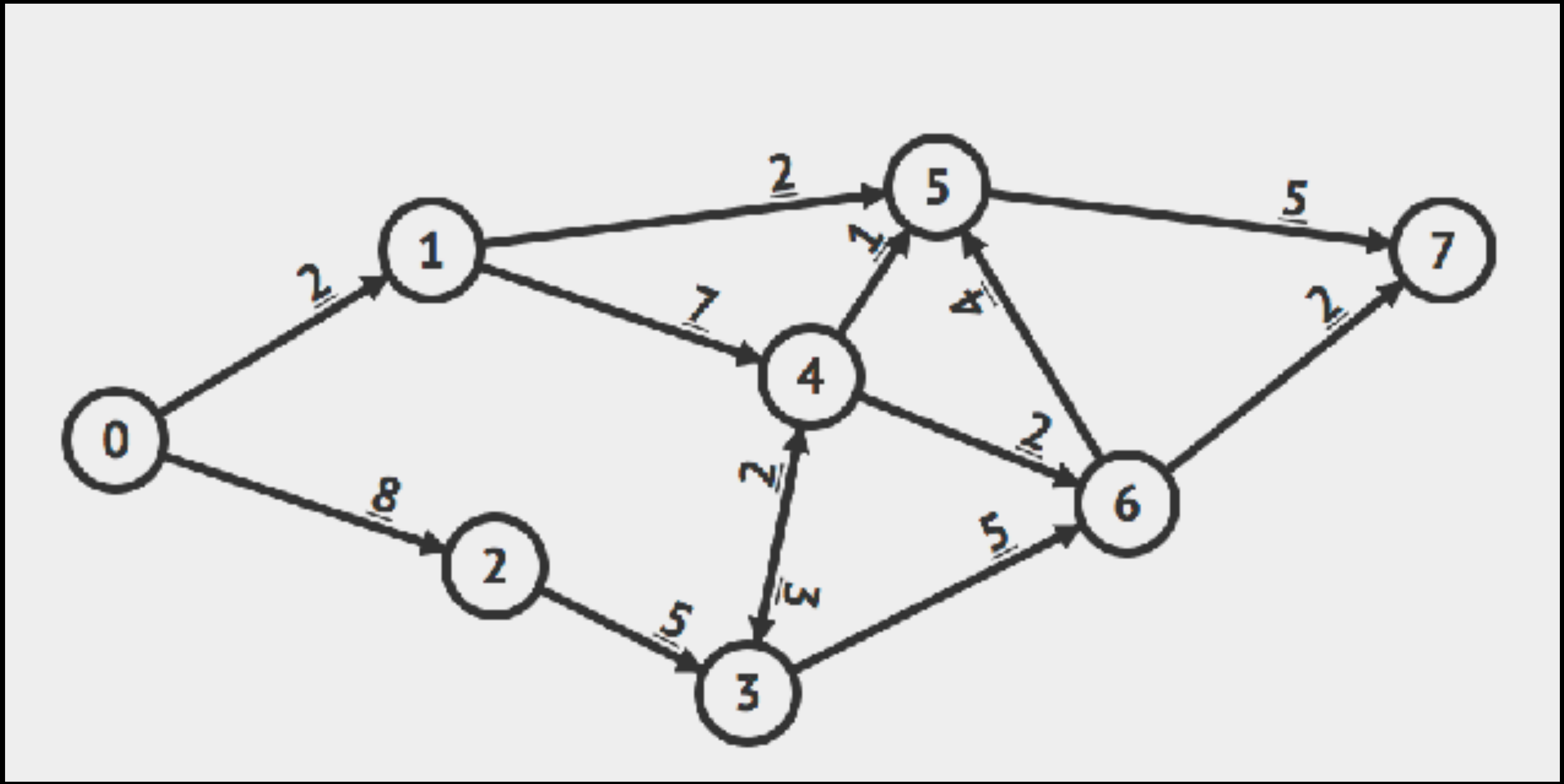
Edmonds-Karp

This algorithm is a specification of Ford-Fulkerson, where we use a BFS to find paths, treating the graph as unweighted.



Edmonds-Karp

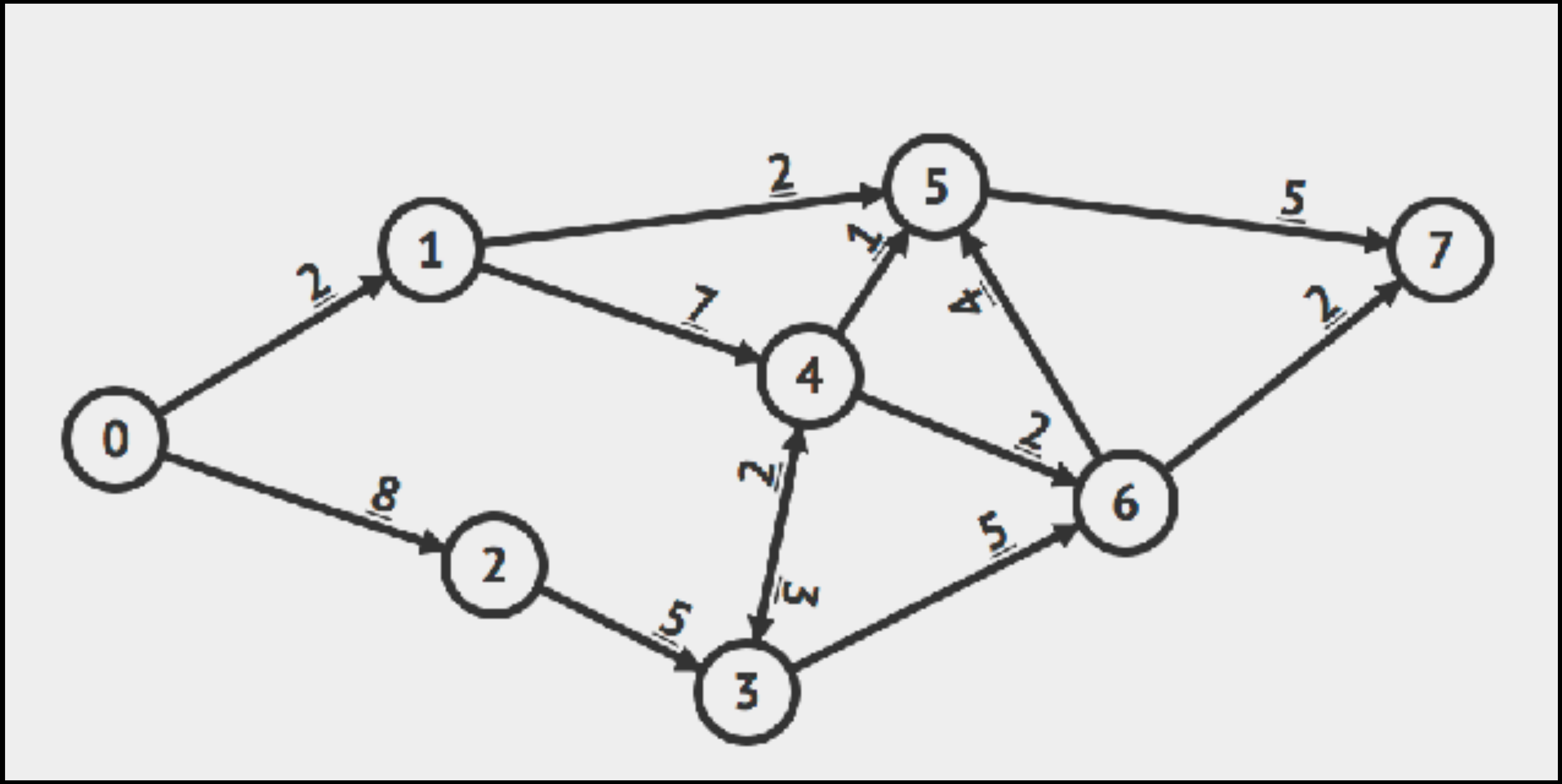
The time complexity is $O(VE^2)$, so it no longer depends on the maximum flow.



Dinic

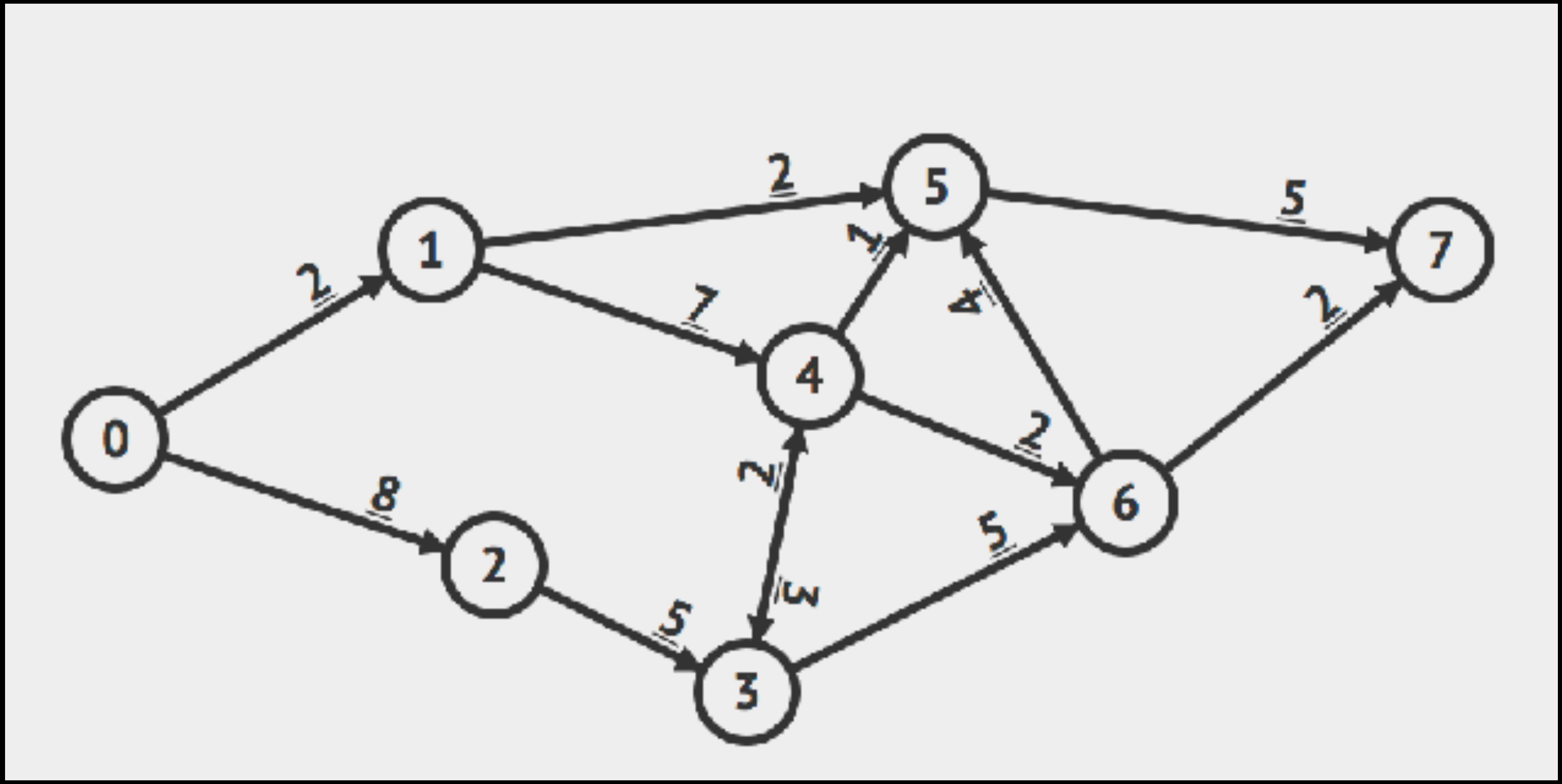
Dinic

This algorithm is similar to Edmonds-Karp, except some optimizations have been made in order to bring the running time down to $O(EV^2)$.



Dinic

By taking advantage of a data structure known as "dynamic trees", this running time can even be brought down to O(EVlogV).

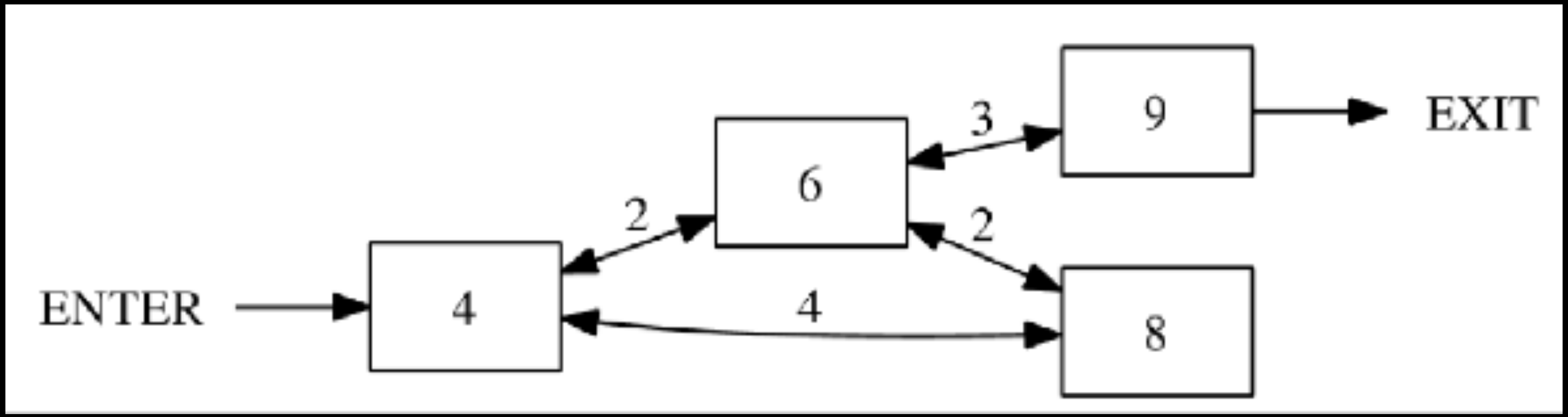


Easy Network Flow Problems

Max Flow Problems

Maximum Flow: Straight-forward application of algorithm. Input size forces use of Dinic's Algorithm.

Maze Movement: Given a list of rooms, and rules allowing you to construct edges with capacities between rooms, determine the maximum flow of people through the maze.



Min Cut Problems

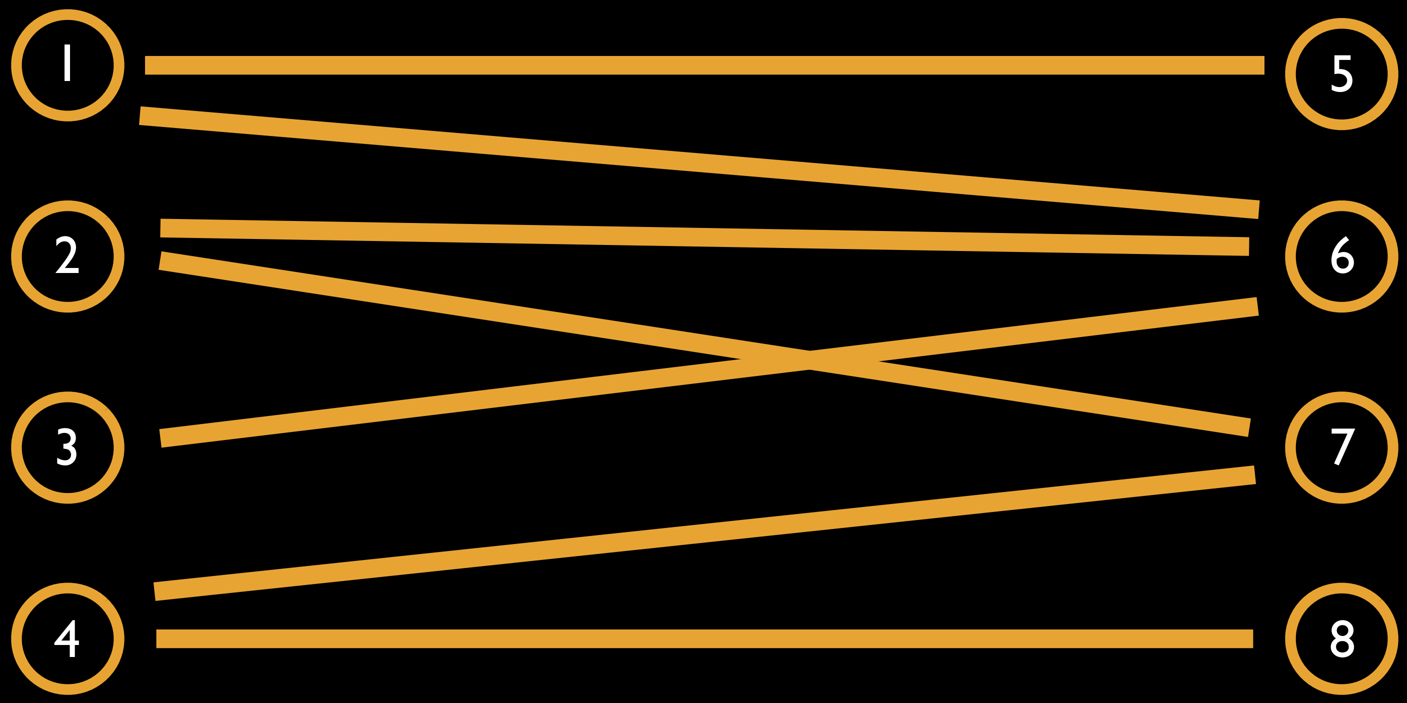
Minimum Cut: Straight-forward application of algorithm. Input size forces use of Dinic's Algorithm.

The King of the North: Given a 2D grid describing the number of men needed to defend each cell, as well as the location of the castle. Asked the minimum number of men needed to fortify some boundary around the castle.

Bipartite Matching

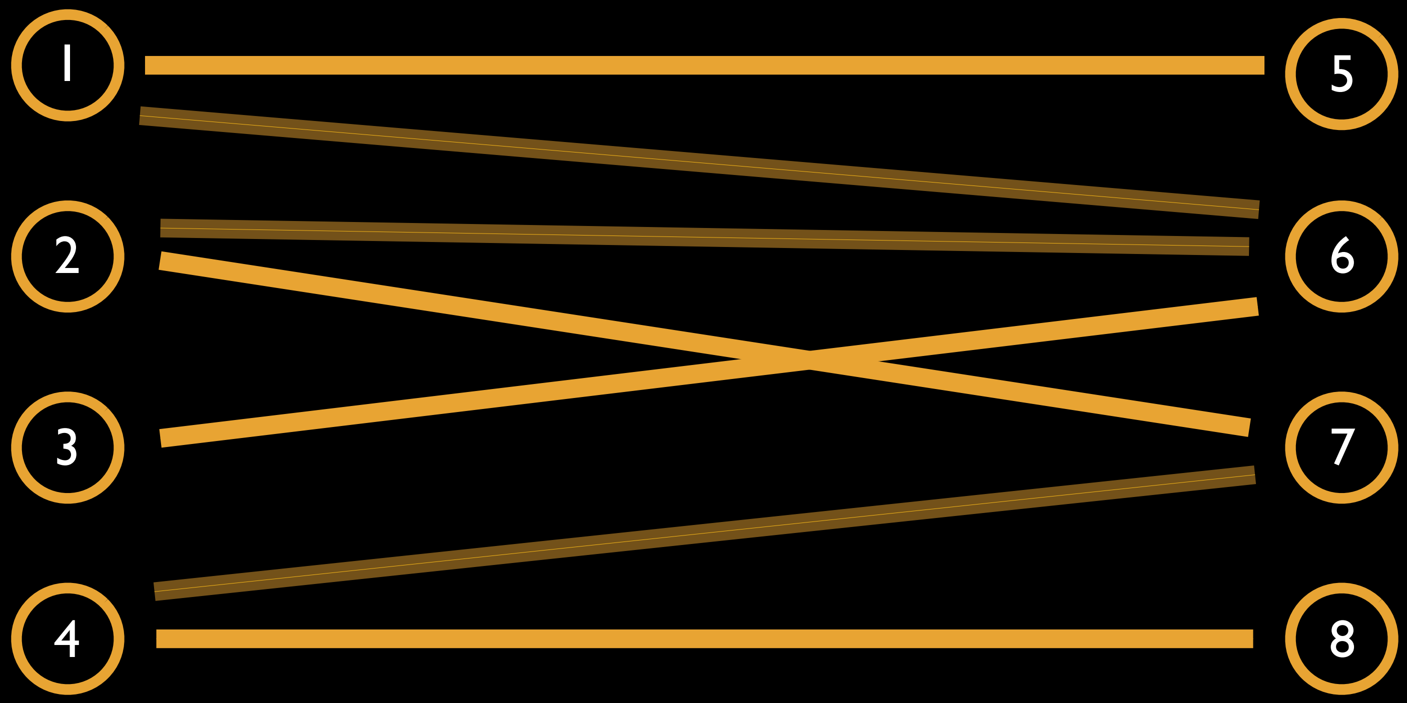
Bipartite Matching

Given a bipartite graph, we want to maximize the number of chosen edges such that each node is only used once.



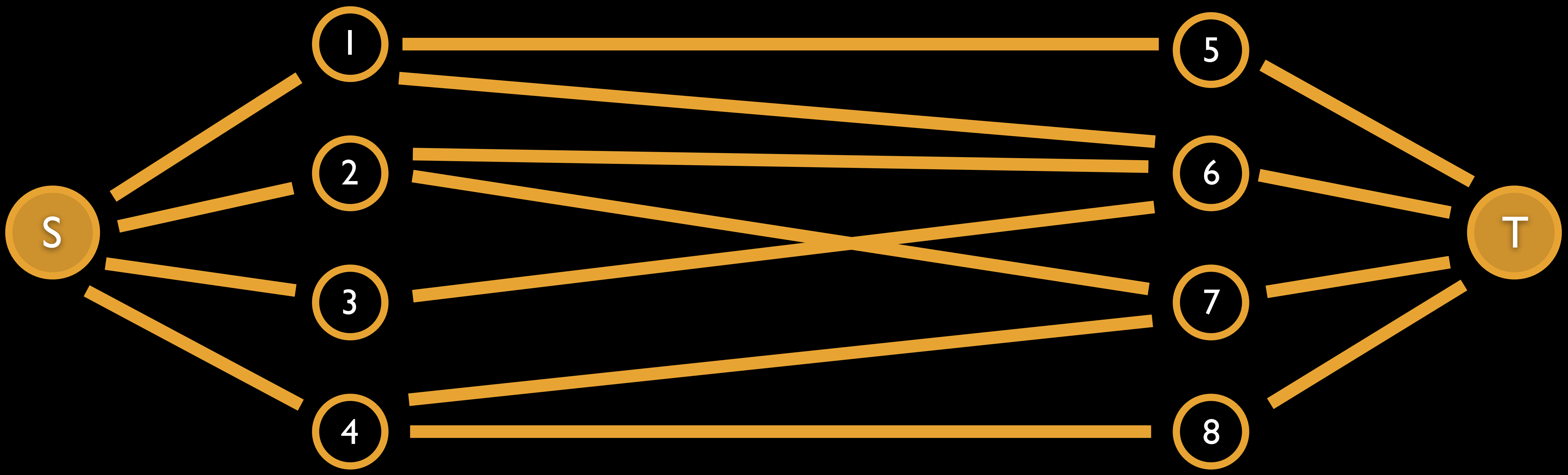
Bipartite Matching

Given a bipartite graph, we want to maximize the number of chosen edges such that each node is only used once.



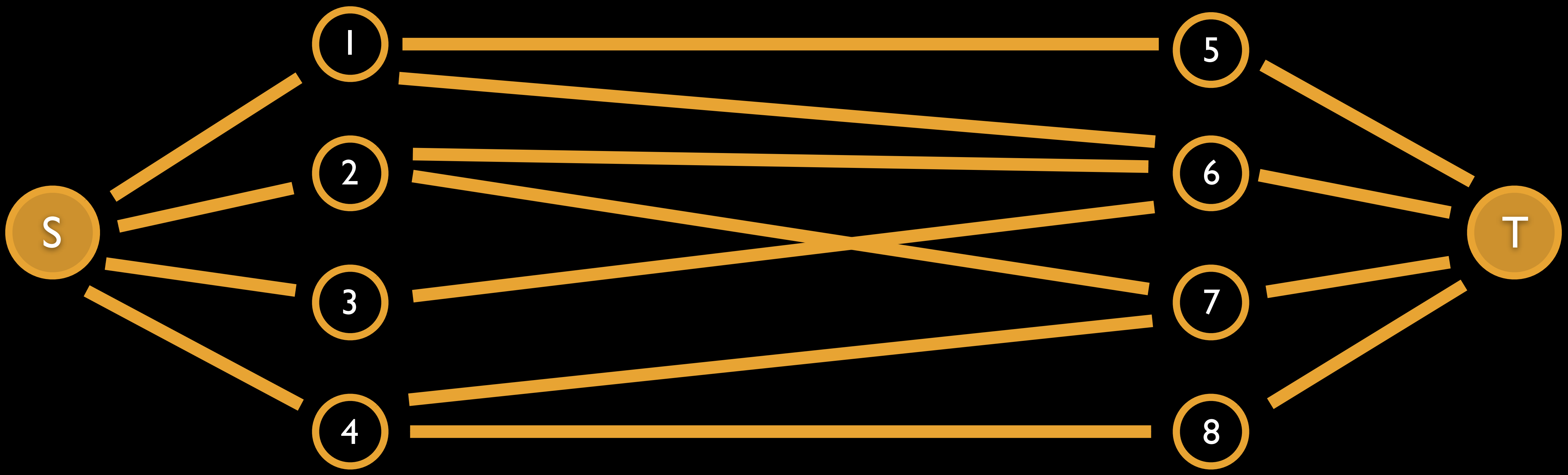
Bipartite Matching

We can use network flow to solve this problem. All we need to do is add a source node and sink node, and then add some connections. All capacities in the graph are simply set to 1.



Bipartite Matching

Since the maximum flow in such a graph is bounded by O(V) then we can use Ford-Fulkerson with a DFS and get a time complexity of O(VE).



Bipartite Matching Problems

Matching Problems

Paintball: Each player has one bullet and is able to shoot some subset of the other players. Determine if it is possible for everyone to be shot at the same time.

Book Club: Given members of a book club and the books that they like, determine if it is possible to trade books around and have everyone leave with a book that they like. Each member brought one book.

Matching Problems

Gopher II: The are ***n*** gophers and ***m*** gopher holes, each at distinct *(x, y)* coordinates. A hawk arrives and if a gopher does not reach a hole in ***s*** seconds it is vulnerable to being eaten. A hole can save at most one gopher. All the gophers run at the same velocity ***v***. The gopher family needs an escape strategy that minimizes the number of vulnerable gophers.

Matching Problems

Elementary Math: Given a list consisting of up to 2500 pairs of integer operands. Asked to assign operators (+, -, *) such that the result of each expression is different. You must output a valid solution or indicate that one does not exist.

Min Cost Max Flow

Min Cost Max Flow

An interesting variant of the Maximum Flow problem is when costs* are added to each edge. Our goal is to find the minimum total cost which achieves the maximum flow.

There exist numerous algorithms to solve this problem (with various time complexities), but in general, we will want small input sizes (not many more than 100 nodes).

***** *This cost will be applied for each unit of flow going through it.*

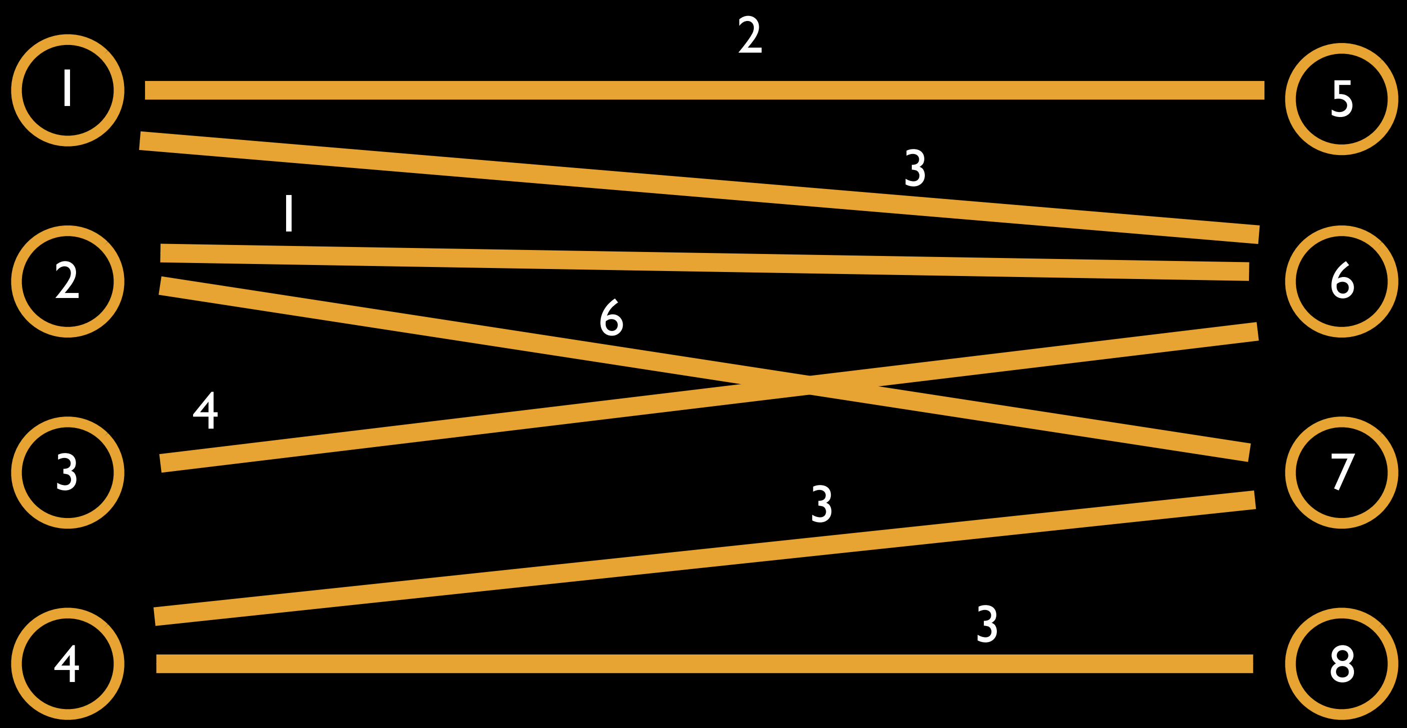
Min Cost Max Flow

One algorithm mentioned in Competitive Programming 3 takes Edmonds-Karp's algorithm, replacing the BFS with Bellman-Ford. The time complexity of this would be $O(V^2E^2)$.

It should be noted that both positive and negative costs are allowed.

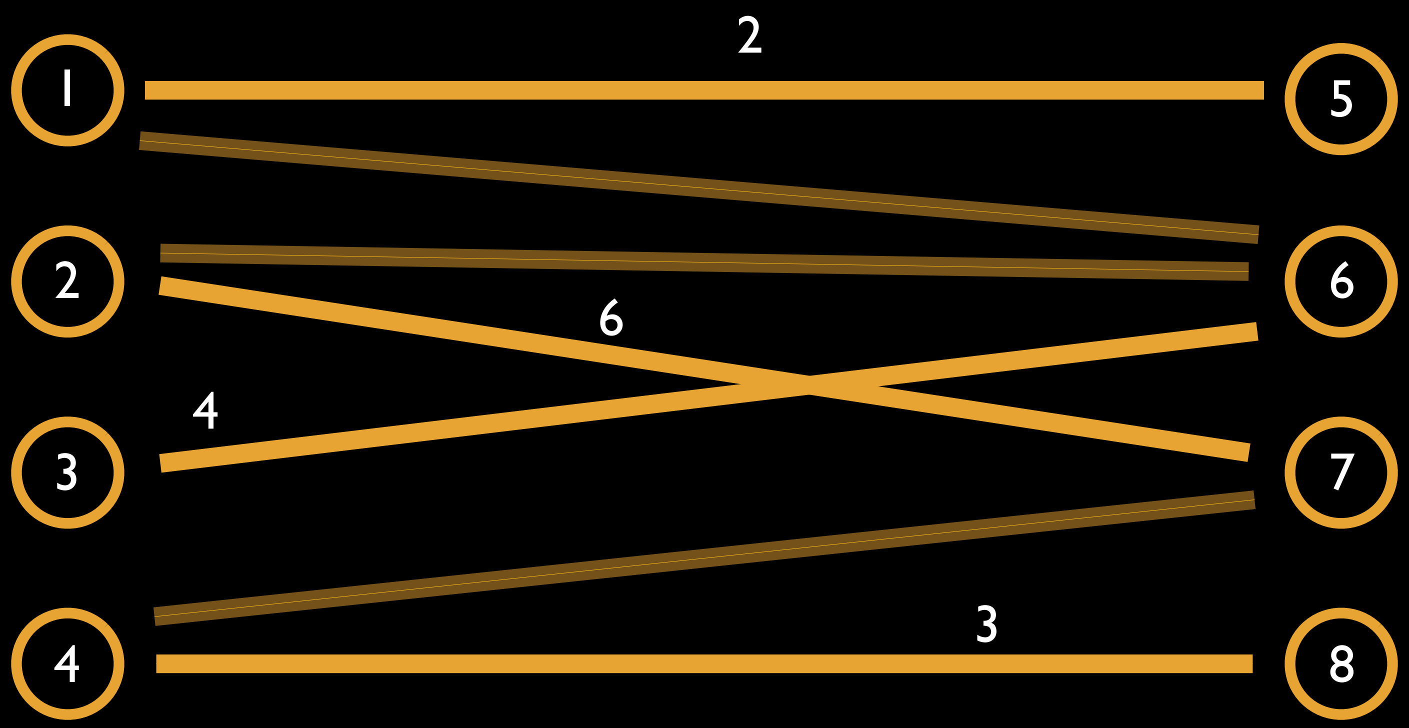
Min Cost Max Flow

For example, we can add costs to our Maximum Bipartite Matching problem and use a Min Cost Max Flow algorithm to solve it.



Min Cost Max Flow

For example, we can add costs to our Maximum Bipartite Matching problem and use a Min Cost Max Flow algorithm to solve it.



Max Flow: 4

Min Cost: 15

Min Cost Max Flow Problems

Min Cost Problems

Minimum Cost Maximum Flow: A straight-forward application of a Min Cost Max Flow algorithm.

Bond: Given ***n*** missions and ***n*** people. For each person you are given the probability of successfully completing each mission. You need to determine the maximum probability of completing all of the missions, if each person were assigned to a mission.

Min Cost Problems

Catering: There are ***k*** catering teams and ***n*** catering requests. The requests are listed in order that they must be serviced. A catering team used for a previous request may be re-used for later requests. You are given the costs associated with transporting the catering teams between different pairs of locations. You must compute the minimum cost needed to service all of the requests.

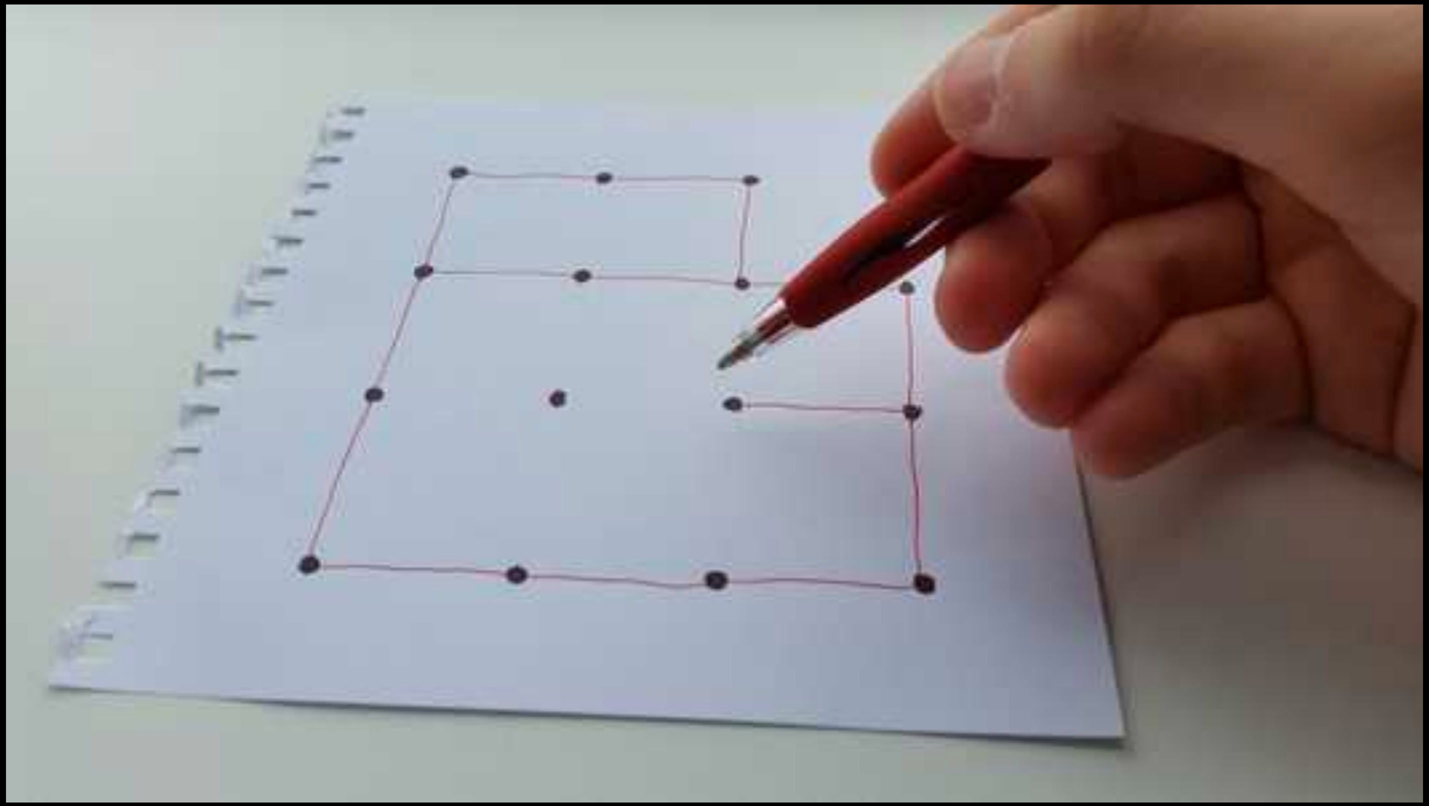
Hard Network Flow Problems

Hard Problems

Airports: Given ***n*** airports (and time it takes to travel between each pair) and a set of ***m*** flights (defined by a starting and ending airport, as well the departing time). Each airport has a certain amount of time that a plane inspection takes (planes must be inspected before taking off again). You must determine the minimum amount of planes needed in order to accommodate all of the flights.

Hard Problems

<u>Dots and Boxes</u>: Given partially completed game of dots and boxes (a grid of up to 80x80). Determine the maximum number of moves which could be made before a square is completed.



Hard Problems

Fake Scoreboard: Imagine we are given an empty table with the sums of each row and column. We want find a way to fill the table with 1's and 0's so that all of the sums match. In actuality, we are representing the 1's with Y's and the 0's with N's, and we are required to find the lexicographically smallest solution (if we were to concatenate the rows in order).

Further Network Flow Variants

Further Variants

- Vertex capacities / costs
- Multiple sources
- Multiple sinks
- Maximum flow with both lower and upper capacity constraints on each edge

Conclusion

- We can take advantage of the properties in special graphs in order to more efficiently solve them.
- Network flow is a powerful and useful concept, and it has many variants.
- It can sometimes be difficult to identify network flow problems.

Collection of Relevant Problems

DAG Problems

- **Build Dependencies:** open.kattis.com/problems/builddeps
- **Barica:** open.kattis.com/problems/barica
- **Family DAG:** open.kattis.com/problems/familydag
- **Slalom:** open.kattis.com/problems/slalom

Tree Problems

- **Genealogical Research:** open.kattis.com/problems/genealogical
- **Kitten on a Tree:** open.kattis.com/problems/kitten
- **Frozen Rose-Heads:** open.kattis.com/problems/frozenrose
- **Subway Tree System:** open.kattis.com/problems/subway

Eulerian Problems

- **Eulerian Path:**
  open.kattis.com/problems/eulerianpath
- **Catenyms:**
  open.kattis.com/problems/catenyms

Bipartite Problems

- **Breaking Bad:** open.kattis.com/problems/breakingbad
- **Amanda Lounges:** open.kattis.com/problems/amanda

Max Flow Problems

- **Maximum Flow:** open.kattis.com/problems/maxflow
- **Maze Movement:** open.kattis.com/problems/mazemovement
- **Dots and Boxes:** open.kattis.com/problems/dotsboxes
- **Fake Scoreboard:** open.kattis.com/problems/fakescoreboard
- **Chess Competition:** open.kattis.com/problems/chesscompetition

Min Cut Problems

- **Minimum Cut:** open.kattis.com/problems/mincut
- **The King of the North:** open.kattis.com/problems/thekingofthenorth
- **Landscaping:** open.kattis.com/problems/landscaping

Matching Problems

- **Paintball:** open.kattis.com/problems/paintball
- **Book Club:** open.kattis.com/problems/bookclub
- **Gopher II:** open.kattis.com/problems/gopher2
- **Elementary Math:** open.kattis.com/problems/elementarymath
- **Airports:** open.kattis.com/problems/airports
- **It Can Be Arranged:** open.kattis.com/problems/itcanbearranged

Min Cost Problems

- **Minimum Cost Maximum Flow:** open.kattis.com/problems/mincostmaxflow
- **Bond:** open.kattis.com/problems/bond
- **Catering:** open.kattis.com/problems/catering

Sources

- Algorithms Live! Episode 1 - Trees and Diameters *www.youtube.com/watch?v=2PFl93WM_ao*
- Competitive Programming 3 by Steven Halim
- *https://visualgo.net/*
- *open.kattis.com*