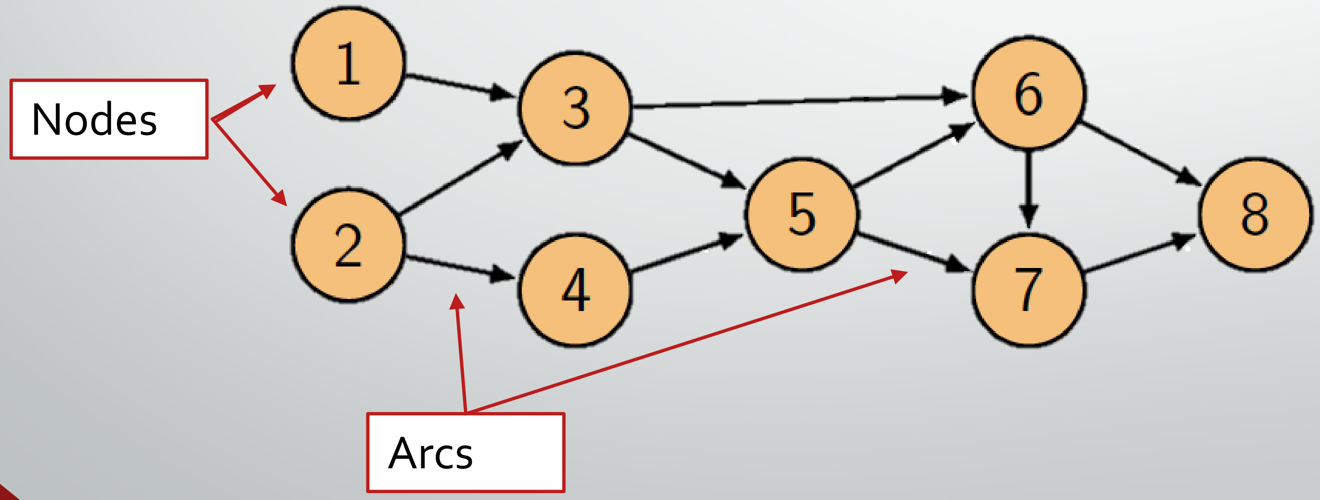
Lecture 3 – Intro to Network Flow Problems

Module 5 – Special Cases of Linear Programs
CS/ISyE/ECE 524

Some important definitions for combinatorial optimization: **graphs**, **vertices**, **edges**

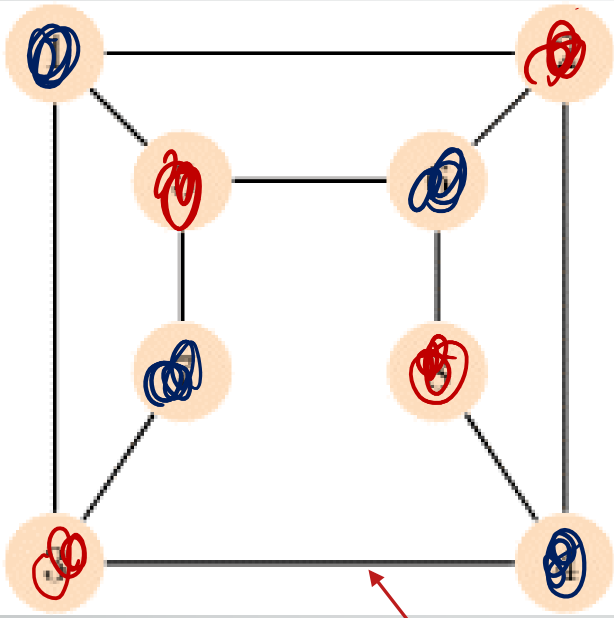
- There are several optimization problems that can be modeled as **graphs** (or **networks**)
  - We often depict graphs as a set of points and lines



A **graph** (or **network**) G is a mathematical structure comprised of a set of **vertices** (or **nodes**) V and a set of **edges** (or **arcs**) E.

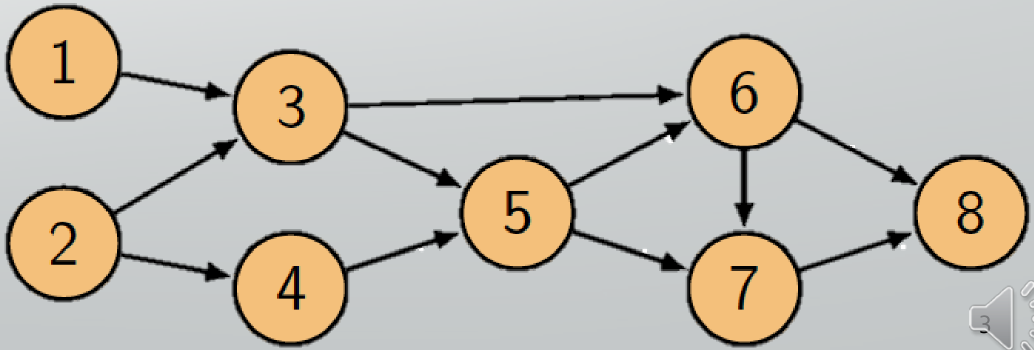
Some important definitions for combinatorial optimization: **bipartite**

A graph $G = (V, E)$ is **bipartite** if V can be **partitioned** into two sets V_1, V_2 ($V_1 \cap V_2 = \emptyset$ and $V_1 \cup V_2 = V$) such that no two vertices within a set are adjacent (connected by an edge).



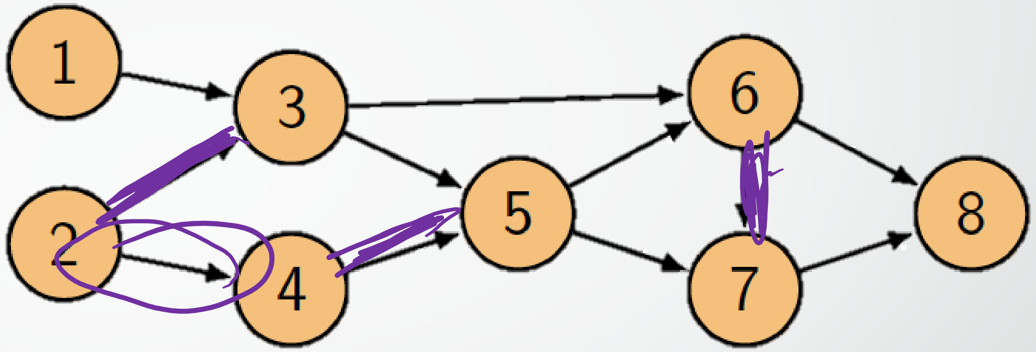
This is a bipartite graph. We can partition the vertices as:
$V_1 = \{1, 4, 6, 7\}; V_2 = \{2, 3, 5, 8\}$

Exercise: Is this graph bipartite?



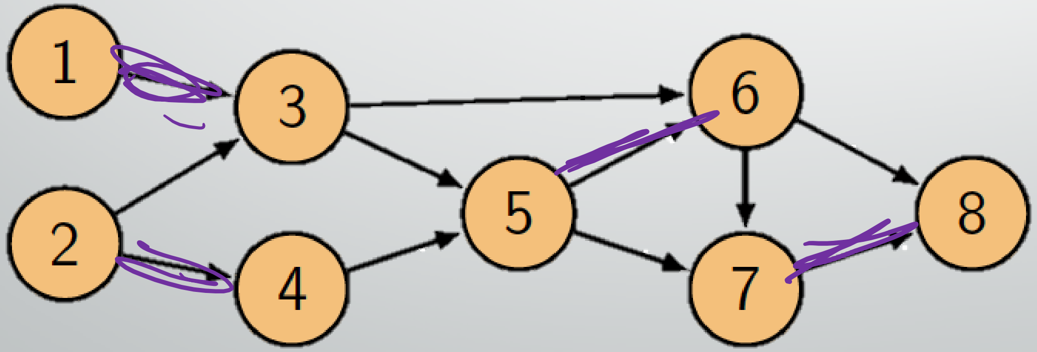
Some important definitions for combinatorial optimization: **matching, perfect matching**

A **matching** is a subset of edges $M \subseteq E$ such that for all $v \in V$, *no more than* one edge of M is **incident** upon it.



One possible matching for this graph:

$$M = \{e_{23}, e_{45}, e_{67}\}$$



A perfect matching for this graph:

$$M = \{e_{13}, e_{24}, e_{56}, e_{78}\}$$

A **perfect matching** is a matching such that for all $v \in V$, *exactly* one edge of M is incident upon it.

Assignment (Matching) Problems

- These are problems that (not surprisingly) involve *matching* one set of things to another set of things (usually creating a perfect matching)
- Let's see an example:

The story:

- A very tired mother needs to assign sandwiches to each of her 5 children. She has made 5 different kinds of sandwiches.
- Each child has indicated their preference for each sandwich by giving each a rating from 0 to 10:
  - 0 – the child hates this kind of sandwich
  - 10 – this is the child's favorite food
- The mother wishes to assign sandwiches in such a way as to maximize the overall satisfaction (the sum of the resulting preferences)

The question: How do we model this?

- Decision variables? Objective? Constraints?

Building the model: variables and objective

Data

- $I = \{1, 2, 3, 4, 5\}$ – the set of children
- $J = \{1, 2, 3, 4, 5\}$ – the set of sandwiches
- c_{ij} – the preference child i has for sandwich j

Variables

- $x_{ij} = \begin{cases} 1 & \text{if child i gets sandwich j} \\ 0 & \text{otherwise} \end{cases}$
- This is called a **binary variable**. We'll learn a lot more about these later!

Objective

- Maximize total happiness:

$$\max_{x} \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij}$$

Question: What constraints do we need?

Finding the perfect match(ing)

Constraints

- Every child gets exactly one sandwich:

$$\sum_{j \in J} x_{ij} = 1 \quad \forall i \in I$$

- Every sandwich is assigned to exactly one child:

$$\sum_{i \in I} x_{ij} = 1 \quad \forall j \in J$$

- A new one: all decision variables must be *binary*:

$$x_{ij} \in \{0,1\} \ \forall i \in I, j \in J$$

Question: Is this a linear program?

In Julia: Sandwich Matching