You want to invite your 200 closest friends to a party at your mansion. Your friends are a fractious bunch, and you have many pairs of friends who don't get along and cannot be in the same room together. Do you have enough rooms to host the party?

Each Boeing 787 Dreamliner is built from 2.3 million parts sourced from suppliers around the world. Not all 2.3m parts are put together at once—some parts are assembled from sub-parts at various facilities before being shipped elsewhere for further assembly.[1]

Given the full dependency information, what is the shortest time to manufacture a plane from start to finish?

---

[1]For example, before being sent to Washington where they are mounted in the fuselage, seats are put together in Kentucky from seatbelt made in Ohio, buckles imported from China, and cushions that are manufactured in Estonia using locally-sourced upholstery and foam from Pennsylvania.[2]

[2]Lies.

The Province of Moravia is developing a plan to electrify its 10 largest cities and towns, connecting them into a single electrical network. Given the distances required to connect each pair of municipalities directly by trunk lines, what is the shortest total amount of trunk line necessary for all 10 to be connected?

# Graphs

EECS 214

November 20, 2015

# Graph problems

Party invites Graph coloring (people are vertices, conflicts are edges, and rooms are colors)

Dreamliner Dependency graph

Electrification Minimum spanning tree

# Take-aways

- What kinds of graphs are there?
- What are *DFS* and *BFS*, and how can we implement them?

# Definitions

A *graph* is a pair $(V, E)$, where $V$ is the set of *vertices* and symmetric relation $E \subseteq V^2$ is the set of *edges*.

# Definitions

A *graph* is a pair $(V, E)$, where $V$ is the set of *vertices* and symmetric relation $E \subseteq V^2$ is the set of *edges*.

A *directed graph* is a pair $(V, E)$ where $V$ is the set of *vertices* and relation $E \subseteq V^2$ is the set of edges.

# Some useful definitions

$$Successors(v) = \{(u_s, u_d) \in \boldsymbol{E} : u_s = v\}$$
$$Predecessors(v) = \{(u_s, u_d) \in \boldsymbol{E} : u_d = v\}$$

# Graph search (basic algorithm)

**def** GraphSearch(*start*):
  $visited \leftarrow \varnothing$
  $todo \leftarrow \{ start \}$

  **while** $todo \neq \varnothing$:
    $v \leftarrow$ remove an element from $todo$
    **if** $v \notin visited$:
      Visit($v$)
      $visited \leftarrow \{v\} \cup visited$
      $todo \leftarrow \text{Successors}(v) \cup todo$

# Bread-first search

If we make *todo* a queue (FIFO), we get BFS:

**def** BFS(*start*):
    *visited* $\leftarrow \varnothing$
    *todo* $\leftarrow$ empty queue
    enqueue *start* in *todo*

    **while** *todo* $\neq \varnothing$:
        $v \leftarrow$ dequeue an element *todo*
        **if** $v \notin$ *visited*:
            Visit($v$)
            *visited* $\leftarrow \{v\} \cup$ *visited*
            enqueue Successors($v$) in *todo*

# Take-aways

- What kinds of graphs are there?
- What are *DFS* and *BFS*, and how can we implement them?