



COMP 1010/COMP 1002

Problem Solving III (FL2)



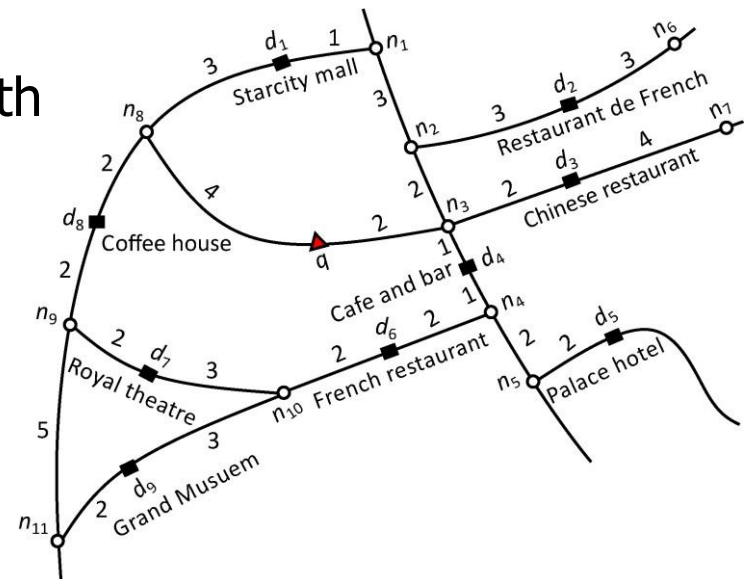
Objectives

- Understand Graph as a data abstraction
 - What is Graph
 - Why it's a great abstraction
 - How does it help
- Types of Graph
 - Understand three types of graphs through examples

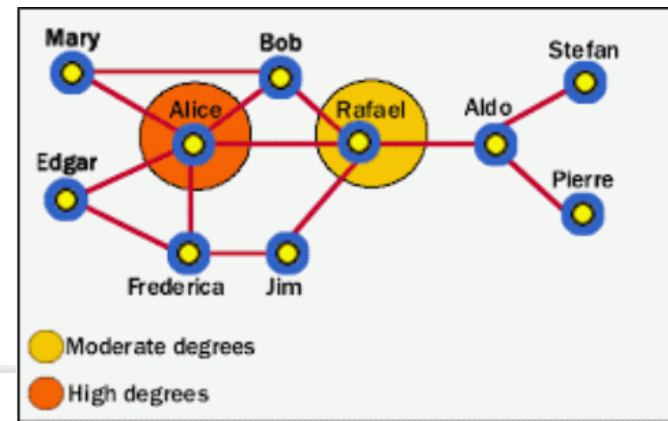
Abstraction



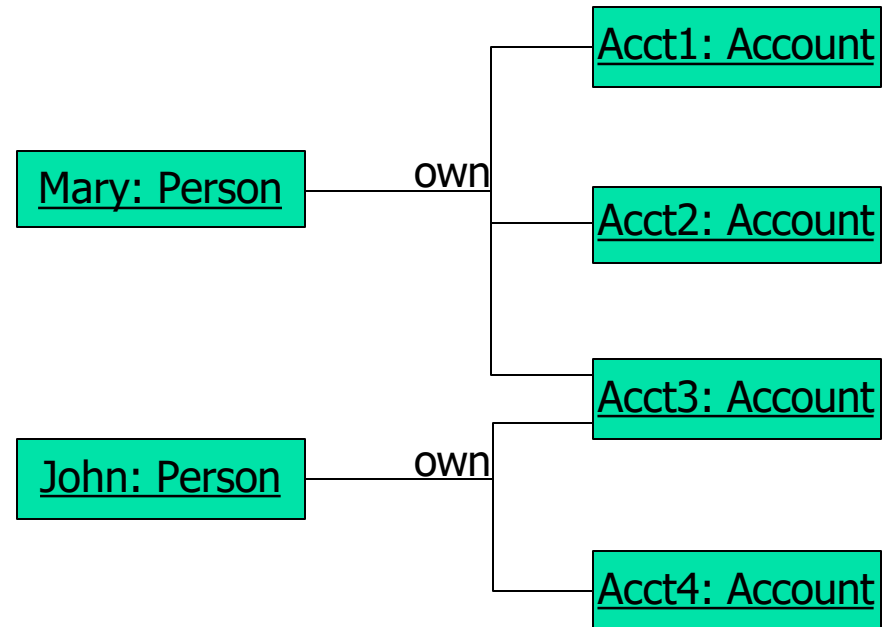
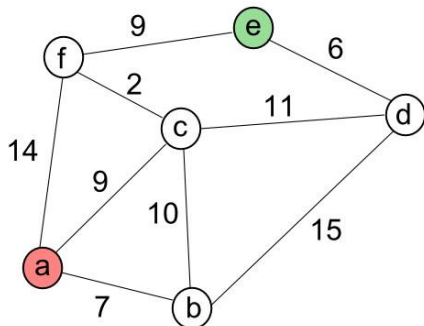
- In computer science, a **road network** is often modeled as a **graph**.
 - This is not a **graph** in mathematics.
 - This is also not to be confused with **graphics** in design.
 - This is further not to be confused with **computer graphics**.



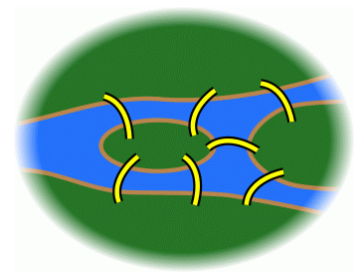
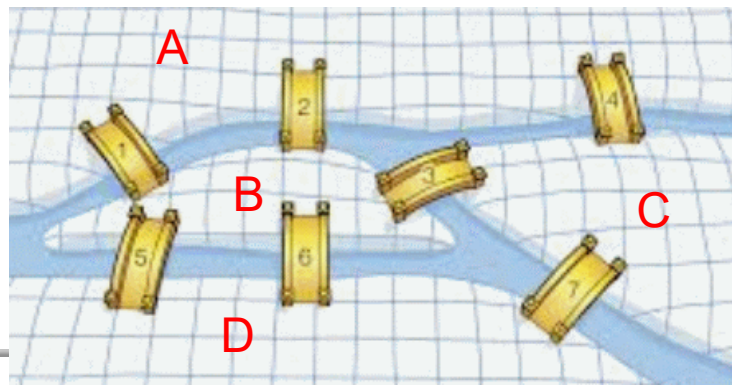
Abstraction



- Why **graphs** are so **common**?
 - They **naturally** represent **entities** (as nodes) and their **relationships** (as edges).
 - Database record
 - Object links
 - Social network
 - Any network
 - Any relationship



Example



- Seven Bridges of Königsberg
 - The problem is to find a walk through Königsberg (a city in Germany) for a tourist that would cross each of those bridges once and only once.
 - Can you do that?
 - How can you prove that there is **no solution**?
 - We model it as a **graph**.
 - Nodes represent the land block and edges represent the bridges.
 - Now the problem becomes finding a **path** that travels **each edge once and only once**.



Seven Bridges of Königsberg

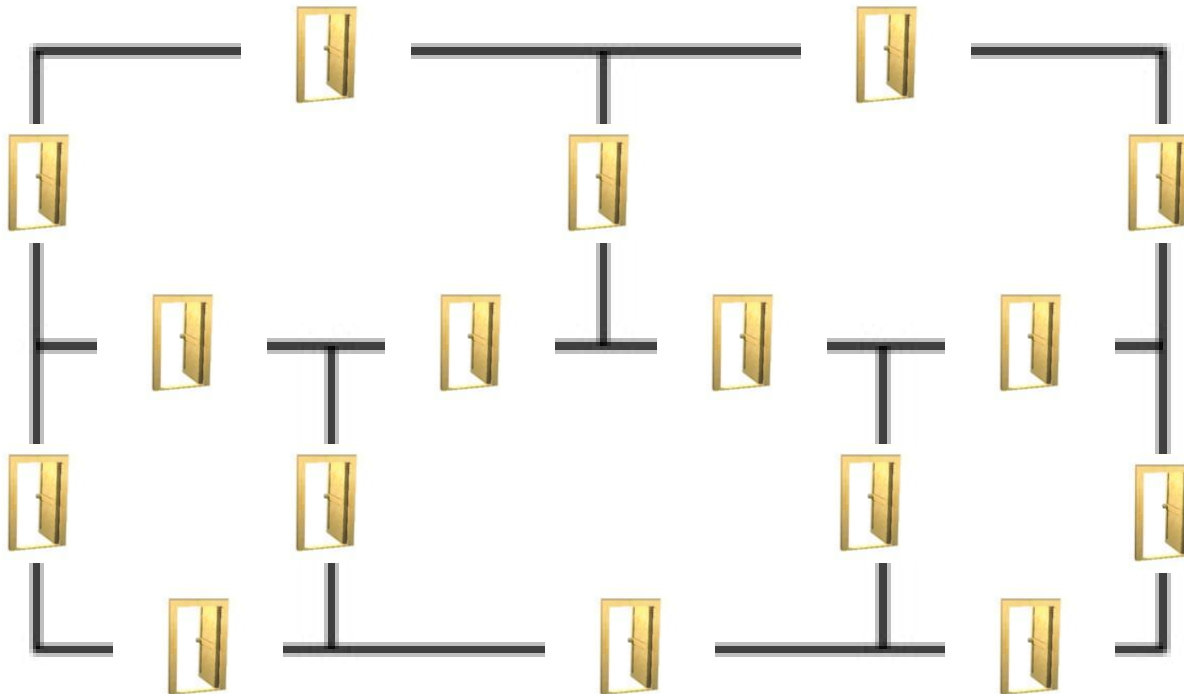
Try solving map1-map4 in the link below

<https://mathigon.org/course/graph-theory/bridges>



Example

- Close all the doors in this layout by passing through each doorway exactly once





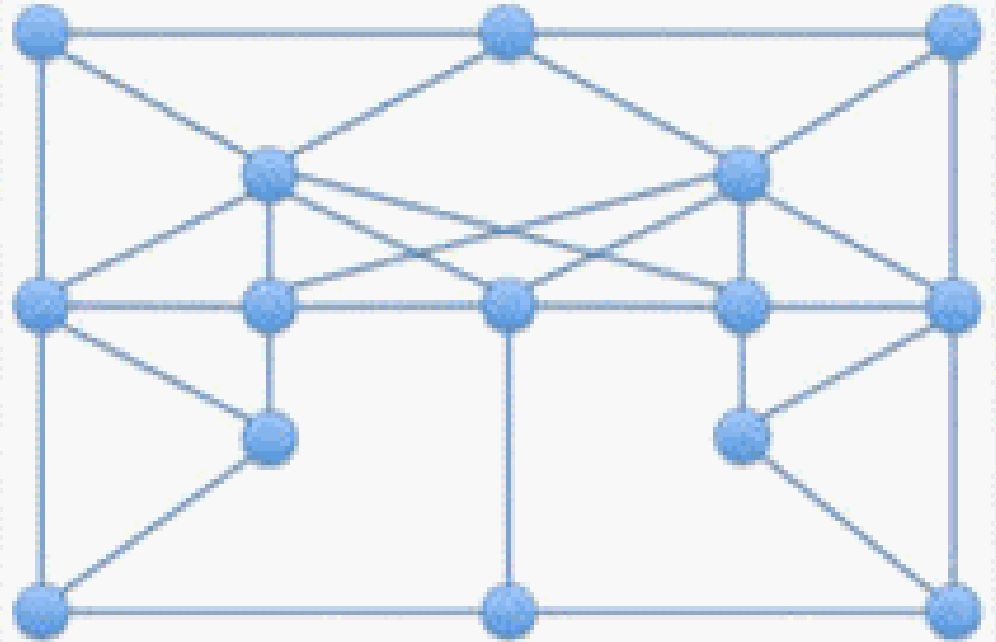
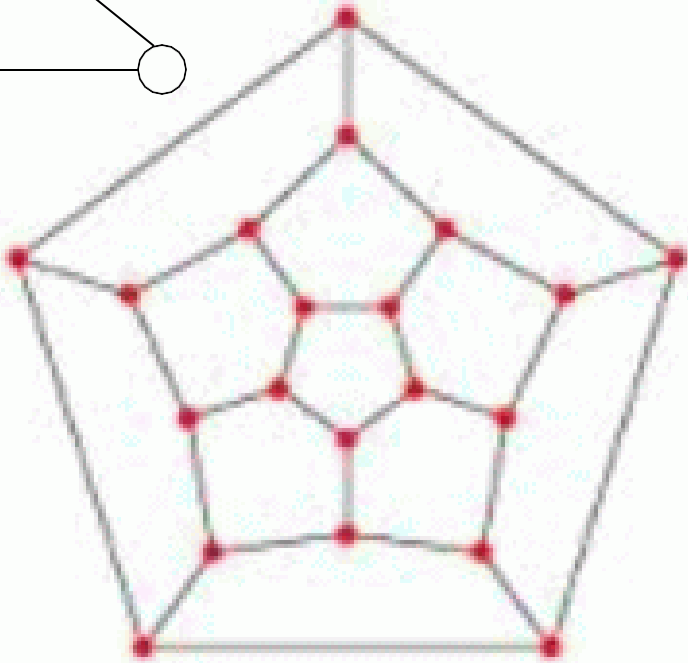
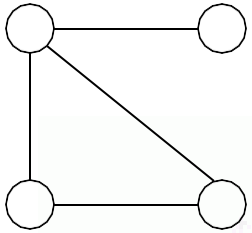
Example

- Visiting each **scenic spot** only once.
 - A related problem is to visit the scenic spots once and only once, returning to the start point (to fly in and fly out).
 - This is also a **graph**, but we are to find a path to visit **each node once and only once**, not each edge once and only once.
 - The previous 7-bridge problem to visit each edge once and only once is **easy**, but this problem to visit each node once and only once turns out to be much **harder**.

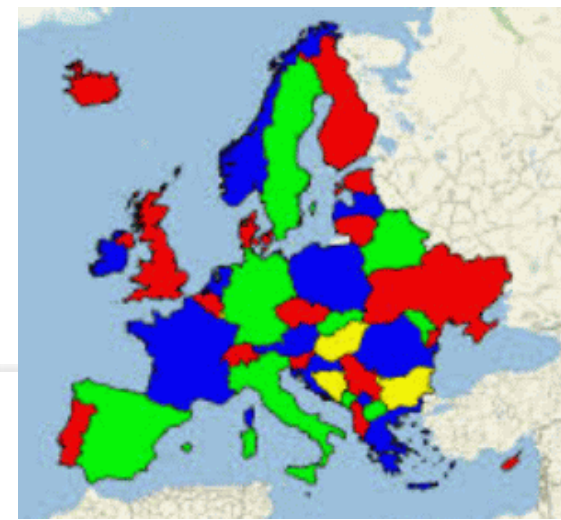


Example

- Try these



Map Coloring Problem



- Given a map
 - We are to properly color it such that neighboring countries do not share the same color.
- Abstraction
 - Convert it into a **graph**.
 - Each country is a **node**.
 - If country A shares border with country B, then there is an **edge** between A and B.
- Observation
 - An **edge** **excludes** the use of a **same color** between the two nodes.
 - Such edges are called **conflict edges** as they imply a conflict between connecting nodes (e.g. enemy in social network).



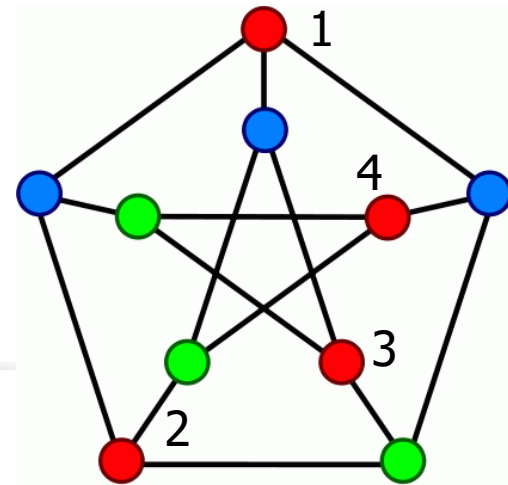
Map Coloring Problem

Try coloring map in the link below

<https://mathigon.org/course/graph-theory/map-colouring>

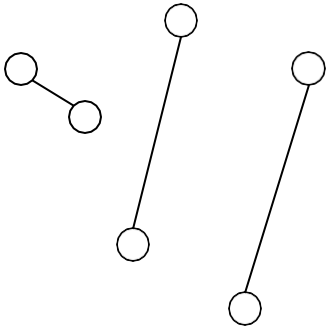


Map Coloring Problem



■ Solution

- Since nodes connected with an edge cannot be assigned the same color, **remove nodes** that have no (conflict) edges among them to be colored first.
- Find an **independent set** and remove them from the graph, removing also their **connecting edges**.
 - An independent set is a collection of nodes that have no connecting edges within the collection.
 - For example, nodes 1,2,3,4 form an independent set.
- Repeat the procedure for the remaining nodes, each set receiving a color, until nothing is left.
- Example:
 - Starting from top red color 1, try other possible nodes to be colored red, e.g. 2,3,4.
 - We remove red nodes and edges linked to them.
 - We could color 3 as blue and remaining as green

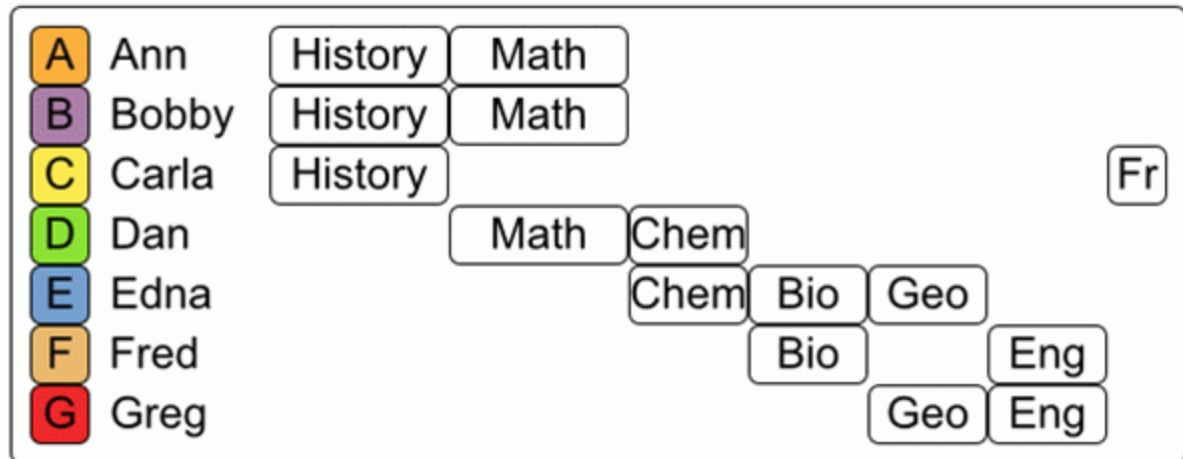


Exam Scheduling Problem

- Exam scheduling problem
 - Schedule all examinations to proper time slots.
- Constraints
 - Two exams cannot be scheduled in the same time slot, if **at least one student is taking both**.

Examination timetabling

Assign each exam a period and a room.



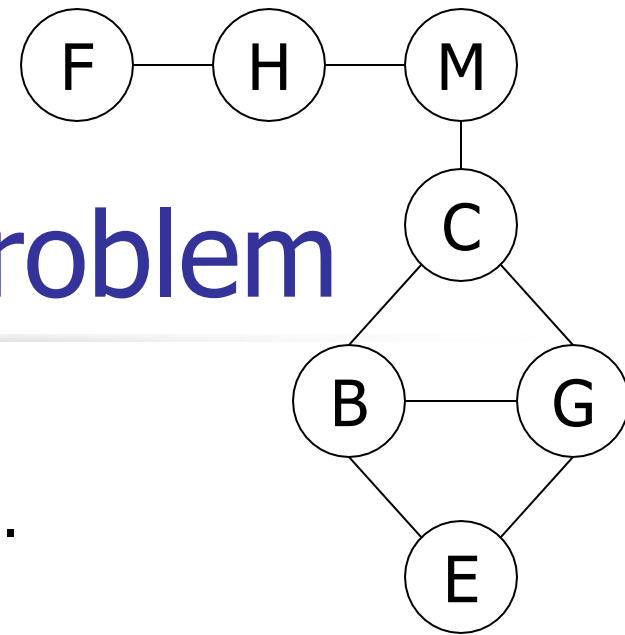


Exam Scheduling Problem

- Direct solution
 - Try like **fitting puzzle** pieces until you get an answer.
 - Make some observation about the actual data and make **human judgment** to fit the schedule.
 - Make use of some **rules** (guidelines) and ask computer to do it.
 - The first two are common approaches taken by laymen.
 - Could the third approach produce a good result?
 - It really depends on your rules.



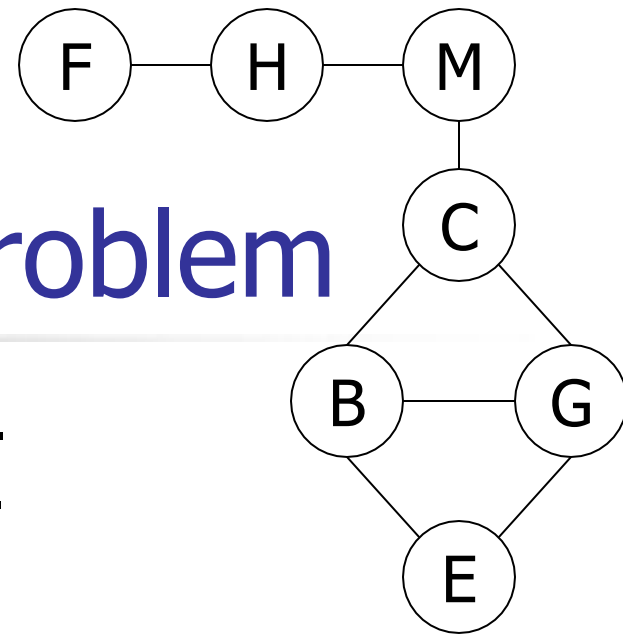
Exam Scheduling Problem



- Using data abstraction
 - Model the **constraints** using a **graph**.
- Modeling
 - Model each exam as a node.
 - Put an edge (conflict edge) between two nodes if there is at least one student taking both exams.
 - This means that the two exams cannot be scheduled together.
 - The result is a **course-conflict graph**.
- The problem is to **remove nodes** that have no edges among them until no more nodes to remove.
 - This becomes the **same problem** as **map coloring**!
 - One could "**reuse**" and adapt previous solutions.



Exam Scheduling Problem



- Round 1: remove F, then M, then B.
 - Round 2: remove H, then C, then E.
 - Round 3: remove G
-
- So F, M, B are scheduled together; H, C, E are scheduled together, and G is scheduled alone.



A Day Change Problem

- How could we model a software **button** to adjust the day for today (Monday).
 - Sunday \Rightarrow Monday \Rightarrow Tuesday \Rightarrow Wednesday \Rightarrow Thursday \Rightarrow Friday \Rightarrow Saturday \Rightarrow Sunday ...
 - Each day can be associated with different actions, e.g. subjects to attend, assignments to work on, etc.

If button pressed then

if day = Sunday then day = Monday; take Monday actions
else if day = Monday then day = Tuesday; take Tuesday actions
else if day = Tuesday then day = Wednesday; take Wednesday actions

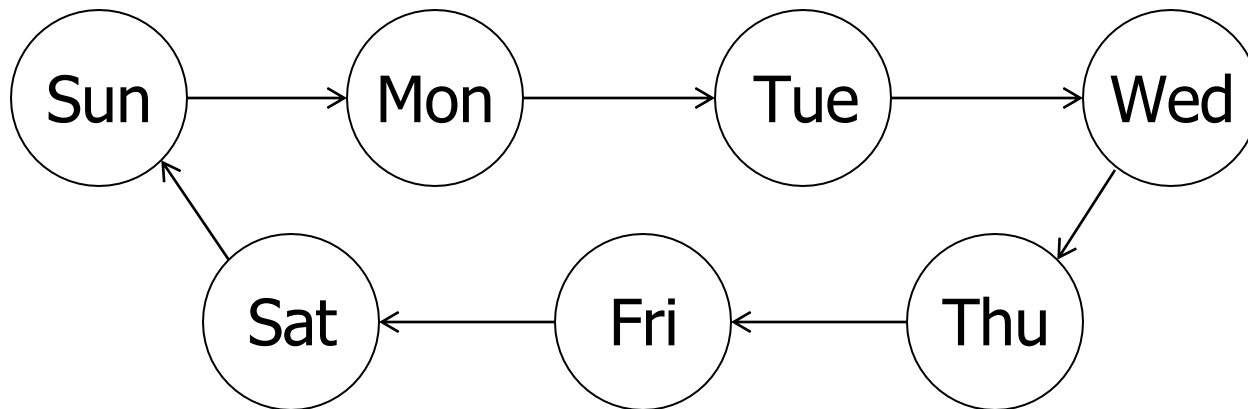
...

else if day = Saturday then day = Sunday; take Sunday actions



A Day Change Problem

- Note that the changes occur in a **cycle**.



- This can be modeled as a **graph**.
 - **Nodes** are the days.
 - **Edges** (with direction) represent the change of days.
- We call it a **state diagram** or **state transition diagram**.
 - Nodes are **states**.
 - Edges are **transitions**.

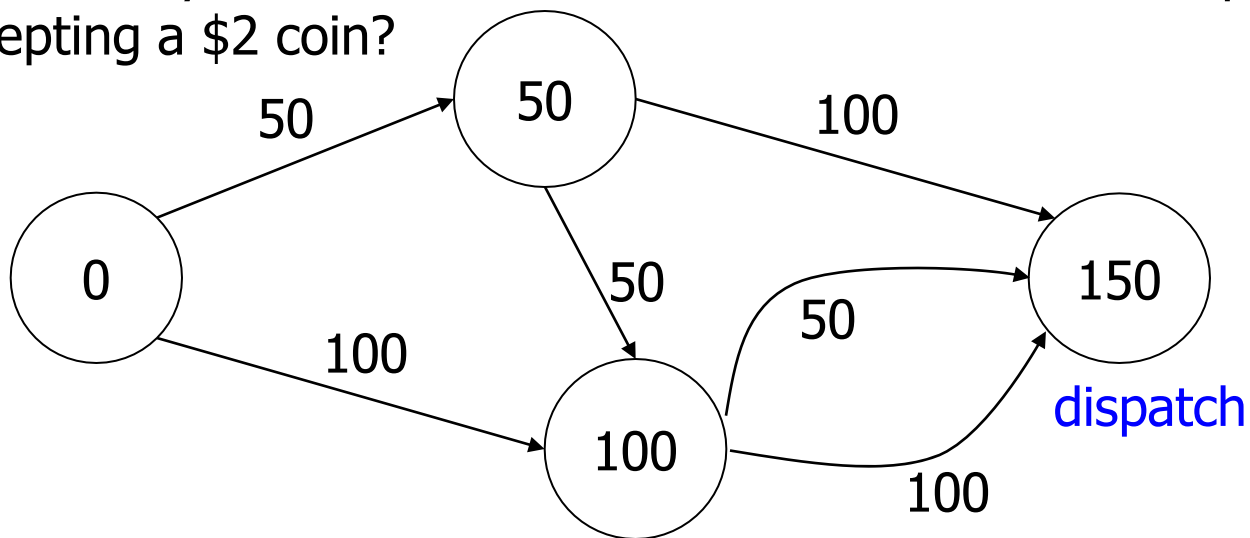


A Vendor Machine

- Consider a vendor machine selling candy costing \$1.50.
 - It accepts only \$1 and 50c coins.
- How can we model this vendor machine?
- We model the state of the machine by the amount of money paid.
 - There are 4 possible states 0, 50, 100, 150.
 - It is clear that 0 is the beginning state of the machine and 150 is the final state of the machine.
- How could it accept also \$2 coins?
- How could you deal with giving the change?

A Vendor Machine

- If we pay 50c from state 0, it will take us to state 50.
- If we pay \$1 from state 0, it will take us to state 100.
- We continue with from state 50, and from state 100.
- Note that even if you pay \$1 from state 100, it just brings you to state 150 (and dispatch the candy).
- How could you add transitions to model the machine capable of accepting a \$2 coin?



MCGW Problem



- Man, Cabbage, Goat, Wolf problem
- Problem
 - Bring the cabbage, goat and wolf from the East (right) side of the river to the West (left) side.
- Constraints
 - The man can **bring only one** of the cabbage, goat or wolf at any time.
 - The **cabbage cannot stay with the goat alone** or it will eat the cabbage.
 - The **goat cannot stay with the wolf alone** or it will eat the goat.



MCGW Problem

- How do you solve this problem?
 - In an ad hoc manner?
- A more **systematic** way:
 - Start with a picture showing all on East side of the river.
 - Draw successive pictures to show the changing situations.
 - A solution is found with a picture shows that all are on the West side of the river.
 - Trace back for the steps leading to the target situation.
- A **picture** shows a **state** of the system and a **link** between two pictures shows a **transition**.
 - The problem can be modeled as a **state transition diagram**, or a **graph**.
 - The goal is to start from a certain **start state** (all on **east**) and go to a certain **target state** (all on **west**).



MCGW Problem

- After **understanding** the problem, next is **abstraction**.
 - Recall that abstraction is a **representation** of a problem with **just enough** important details.
- Let us denote the river by “|” and MCGW for man, cabbage, goat and wolf respectively.
- At the beginning, we have “| MCGW” and our goal is “MCGW |”.
 - There are four possible moves from the beginning:
 - | MCGW -> M | CGW
 - | MCGW -> MC | GW
 - | MCGW -> MG | CW
 - | MCGW -> MW | CG
 - Any problem with the moves above?

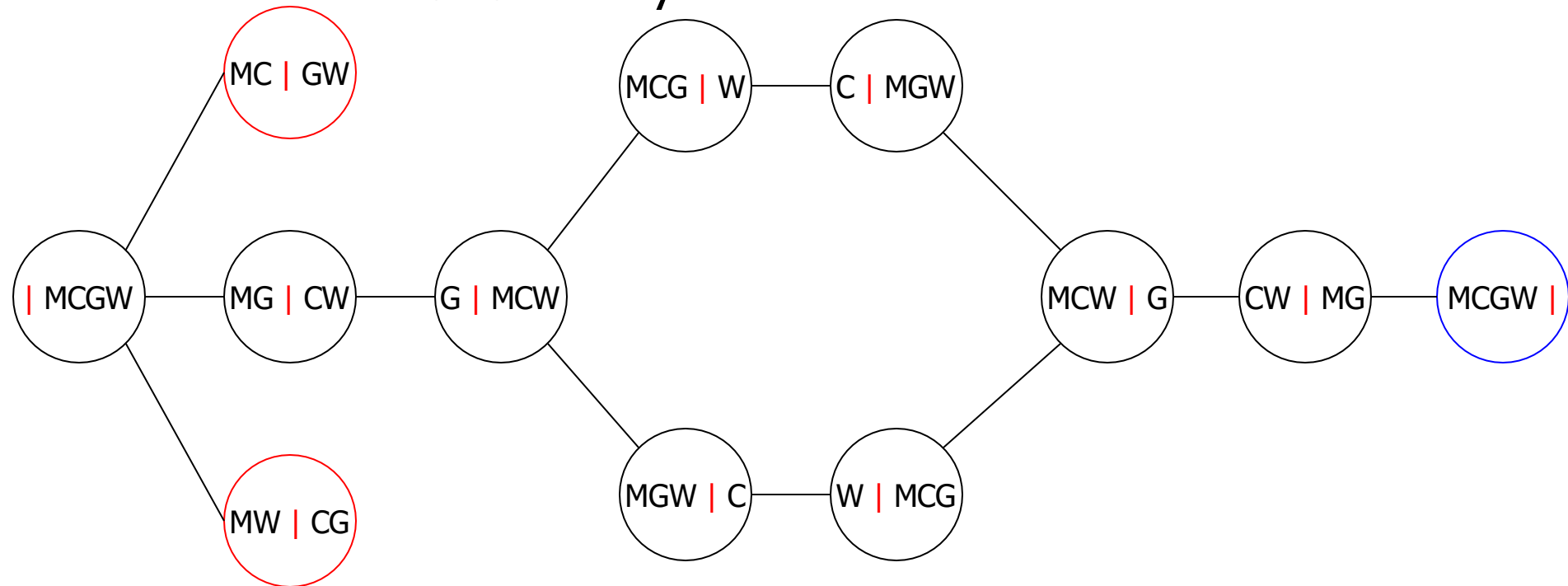


MCGW Problem

- After making the first “good” move, we have 2 possible moves:
 - | MCGW -> MG | CW -> | MCGW
 - | MCGW -> MG | CW -> G | MCW
- Now, we have 3 possible moves from the current situation:
 - G | MCW -> MG | CW
 - G | MCW -> MCG | W
 - G | MCW -> MGW | C

MCGW Problem

- Putting them together in a **graph**:
 - 7 steps, 2 possible answers.
 - What are they?





FL2 Quiz

Attempt the Quiz in the Blackboard Assessments
Section