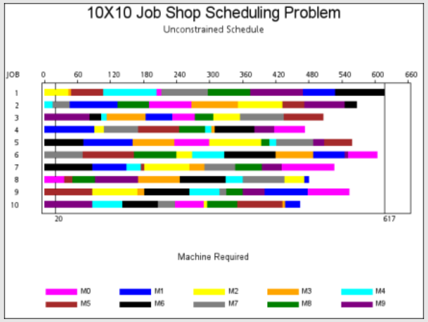
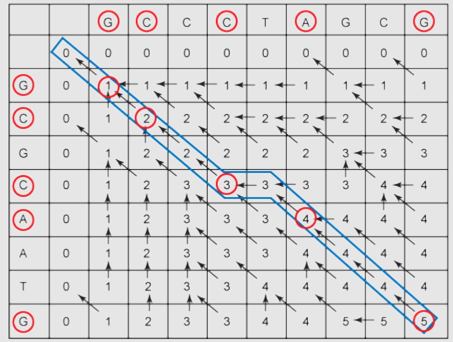
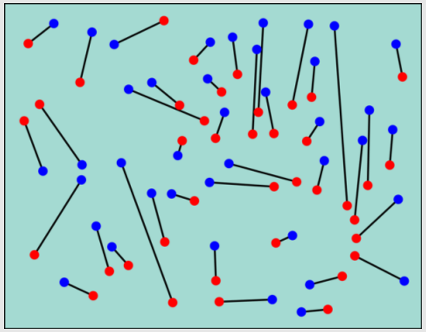
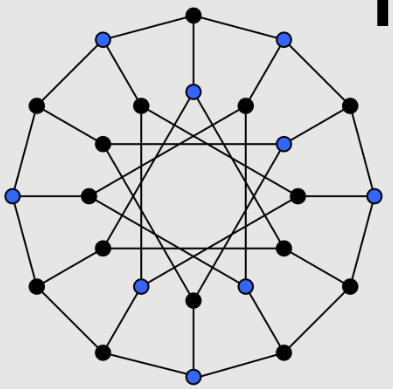
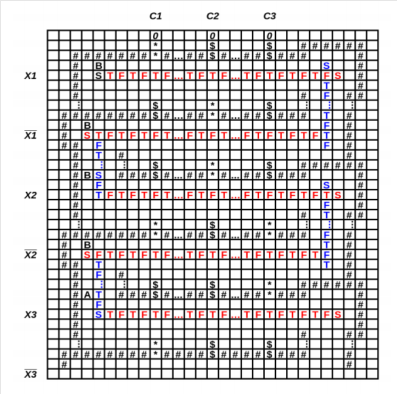






Five Problems





CSE 417

Richard Anderson

Autumn 2024, Lecture 3

Announcements

- Course website:

  //courses.cs.washington.edu/courses/cse417/24au/

- Homework Due Friday

- Office Hours:

Richard Anderson, Monday 2-3 pm, Wednesday 3-4 pm

TA Office hours will be announced soon (and will start this week)

Theory of Algorithms

- What is expertise?
- How do experts differ from novices?

Introduction of five problems

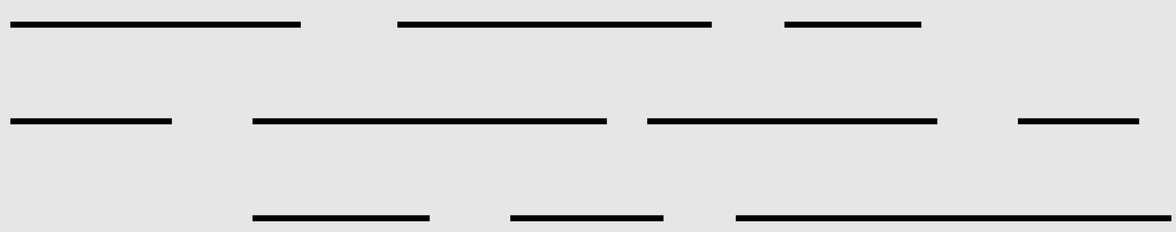
- Show the types of problems we will be considering in the class
- Examples of important types of problems
- Similar looking problems with very different characteristics
- Problems
  - Scheduling
  - Weighted Scheduling
  - Bipartite Matching
  - Maximum Independent Set
  - Competitive Facility Location

What is a problem?

- Instance
- Solution
- Constraints on solution
- Measure of value

Problem: Scheduling

- Suppose that you own a banquet hall
- You have a series of requests for use of the hall: $(s_1, f_1), (s_2, f_2), \ldots$



- Find a set of requests as large as possible with no overlap

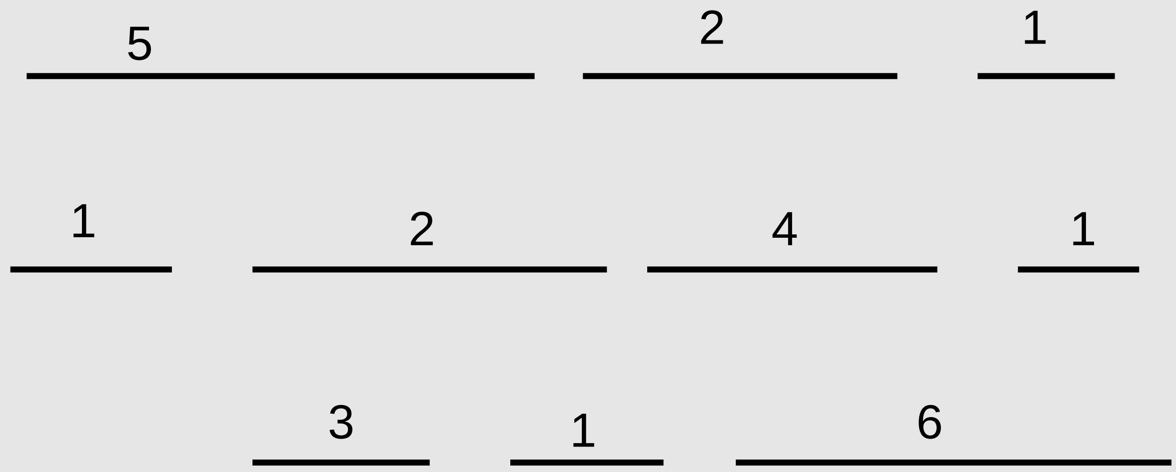
What is the largest solution?

Greedy Algorithm

- Test elements one at a time if they can be members of the solution
- If an element is not ruled out by earlier choices, add it to the solution
- Many possible choices for ordering (length, start time, end time)
- For this problem, considering the jobs by increasing end time works

Suppose we add values?

- (s_i, f_i, v_i), start time, finish time, payment
- Maximize value of elements in the solution



Greedy Algorithms

- Earliest finish time

- Maximum value

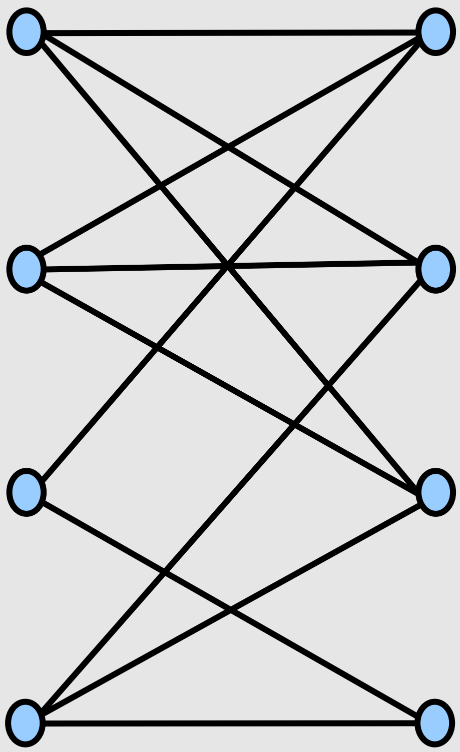
- Give counter examples to show these algorithms don’t find the maximum value solution

Dynamic Programming

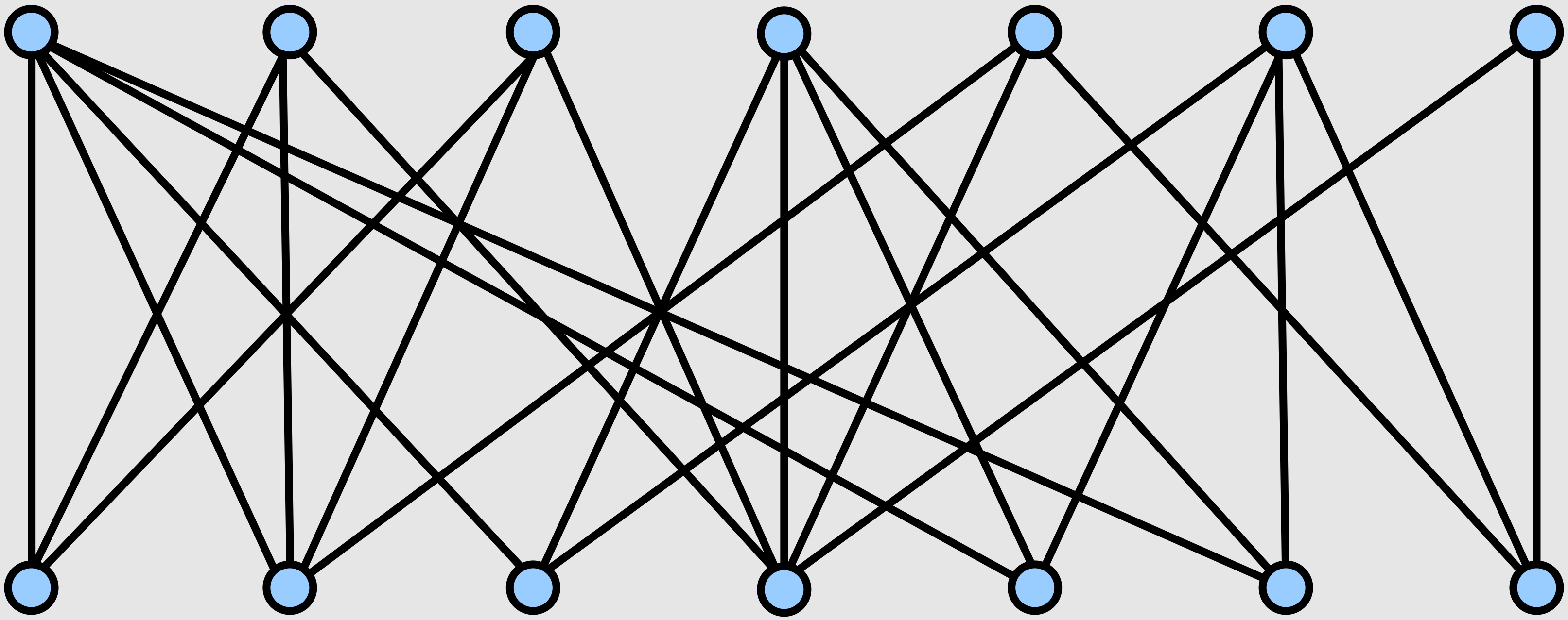
- Requests $R_1, R_2, R_3, \ldots$
- Assume requests are in increasing order of finish time ($f_1 < f_2 < f_3 \ldots$)
- Opt_i is the maximum value solution of $\{R_1, R_2, \ldots, R_i\}$ containing R_i
- $Opt_i = \text{Max}\{ j \mid f_j < s_i \}[Opt_j + v_i]$

Matching (Combinatorial Optimization)

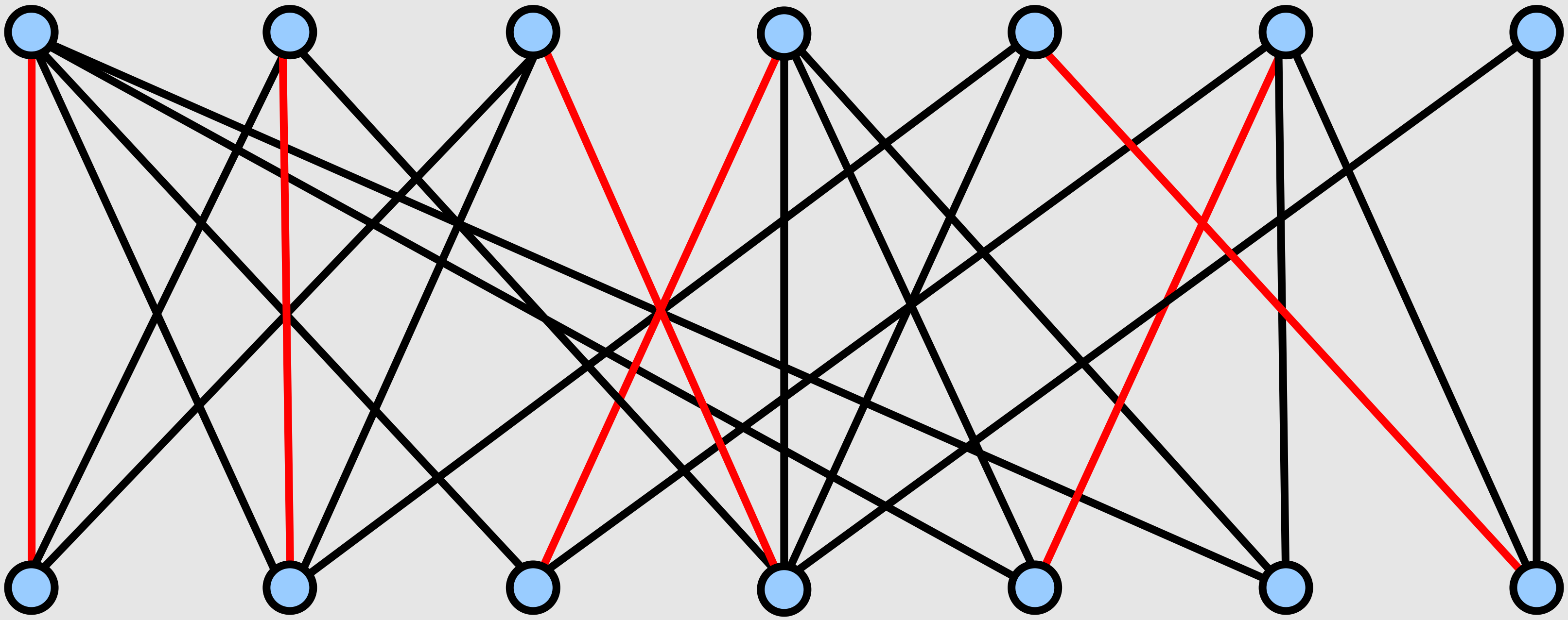
- Given a bipartite graph G=(U,V,E), find a subset of the edges M of maximum size with no common endpoints.
- Application:
  - U: Professors
  - V: Courses
  - (u,v) in E if Prof. u can teach course v



Find a maximum matching

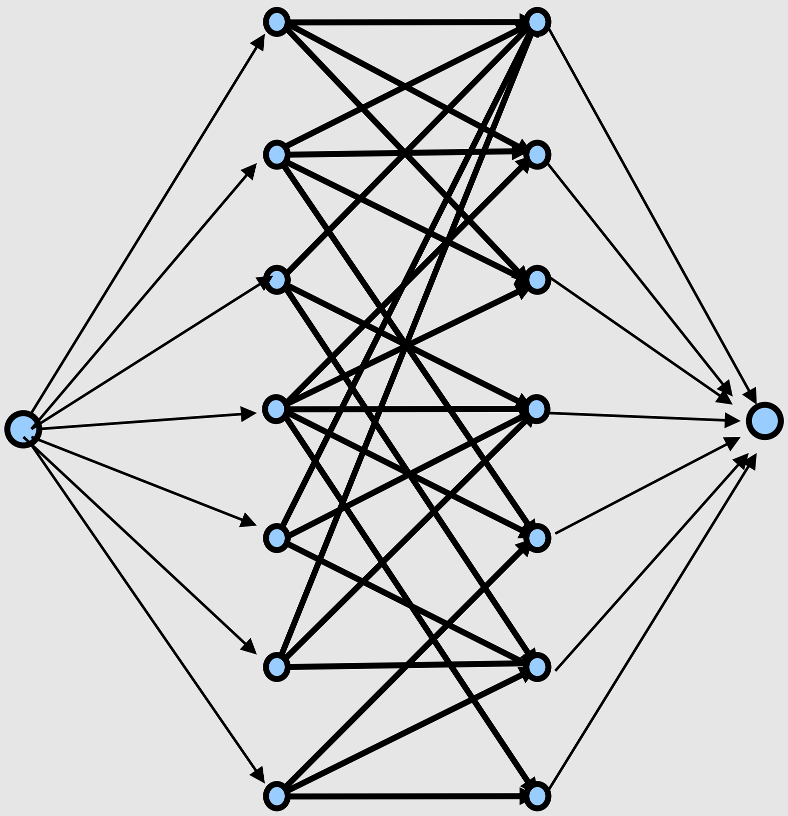


Augmenting Path Algorithm



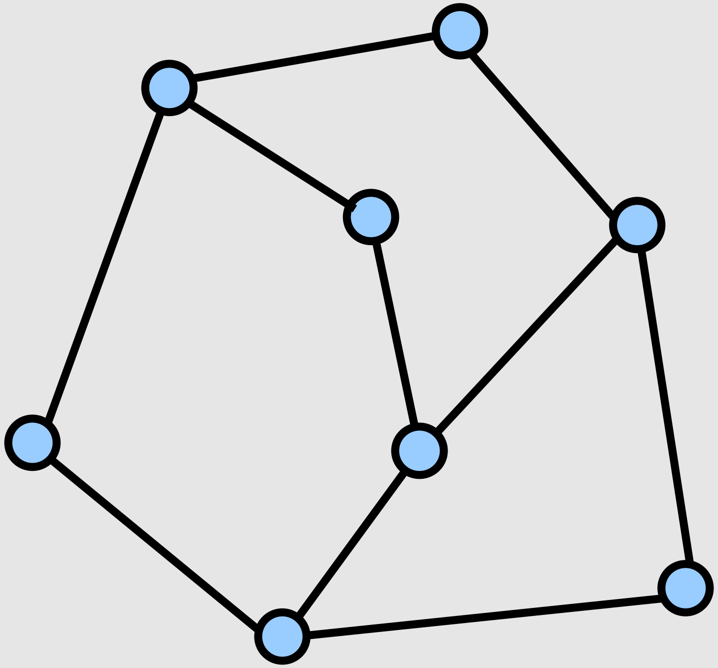
Reduction to network flow

- More general problem
- Send flow from source to sink
- Flow subject to capacities at edges
- Flow conserved at vertices
- Can solve matching as a flow problem

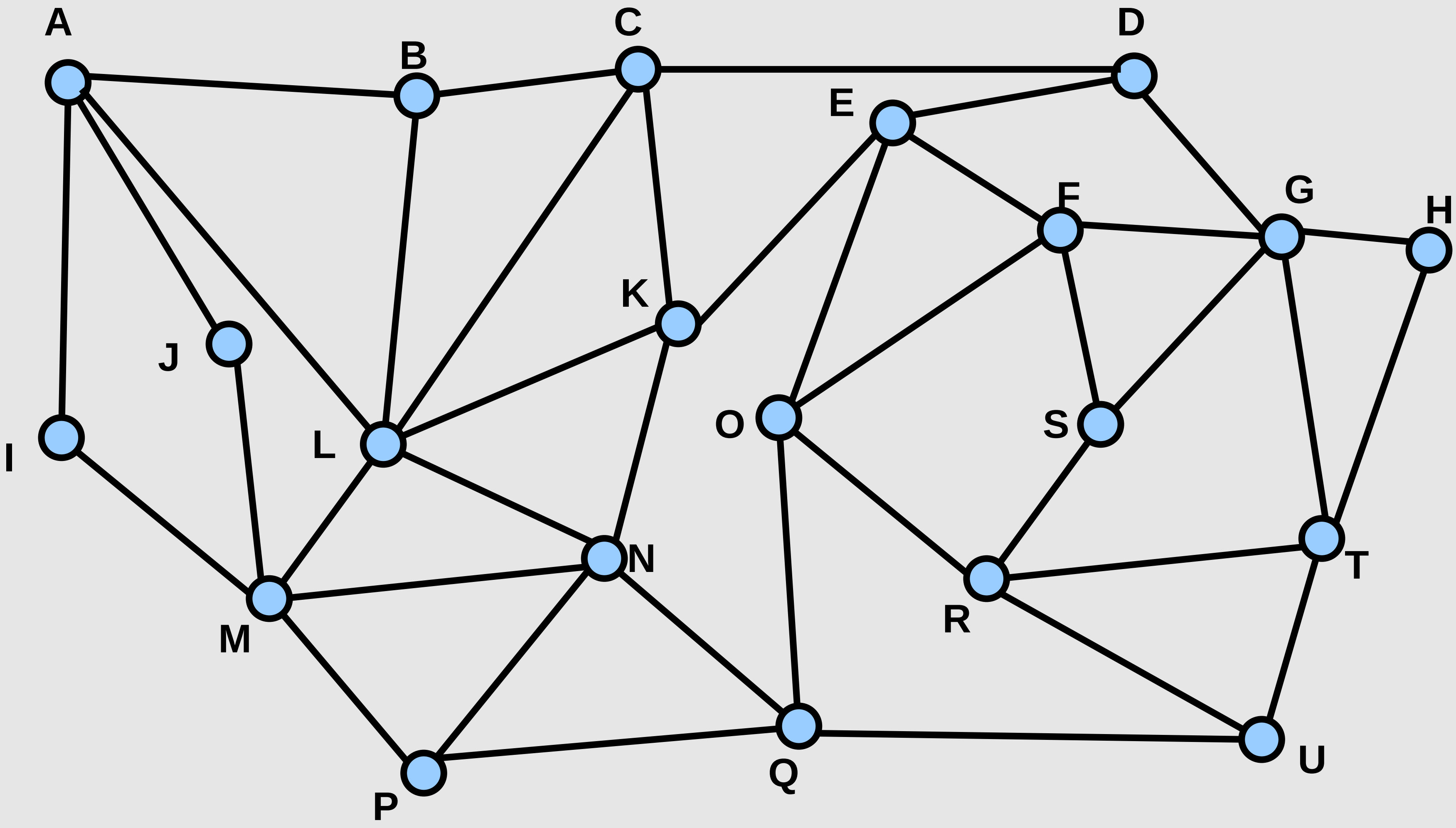


Maximum Independent Set

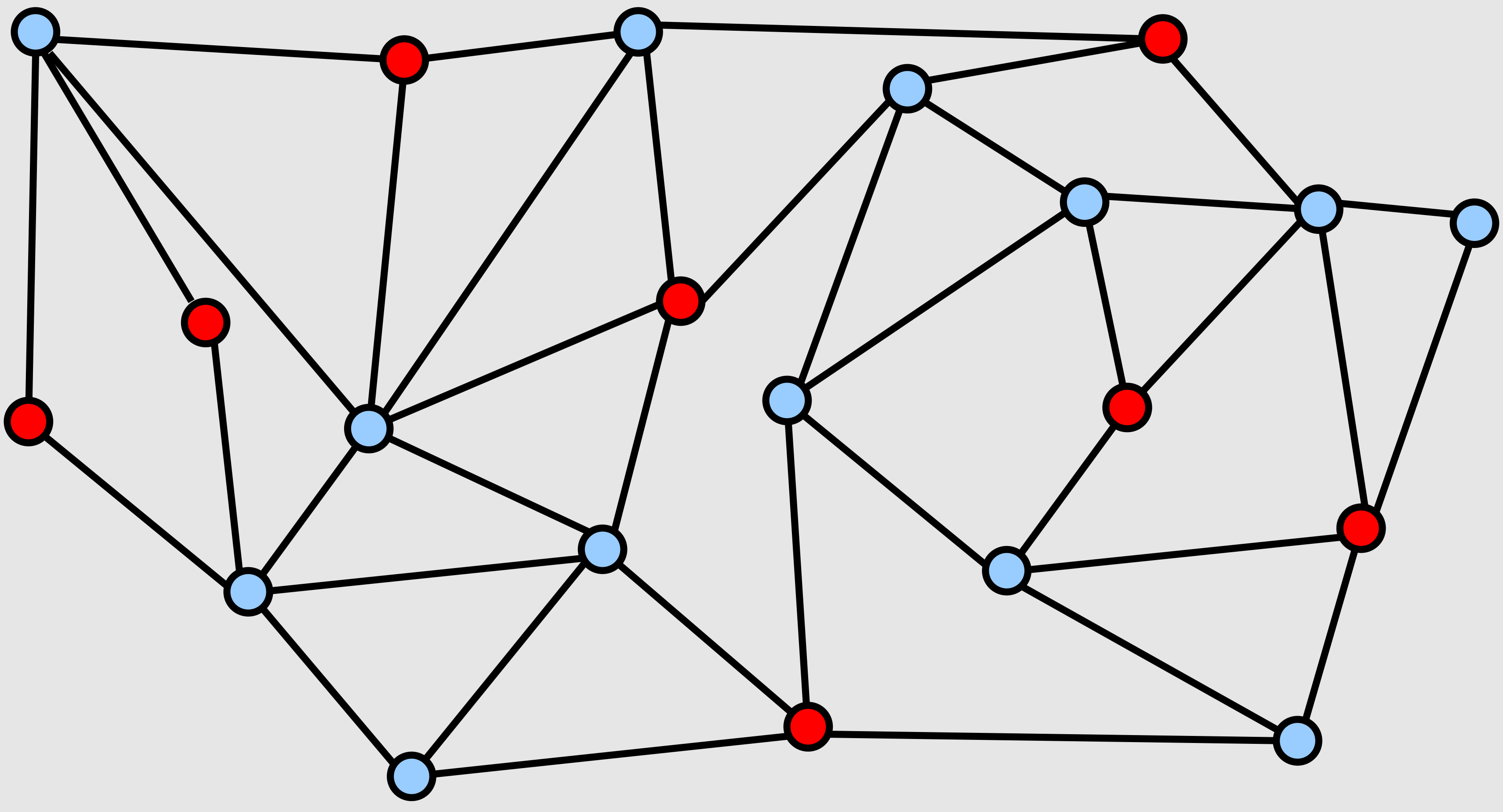
- Given an undirected graph G=(V,E), find a set I of vertices such that there are no edges between vertices of I
- Find a set I as large as possible



Find a Maximum Independent Set



Verification: Prove the graph has an independent set of size 8



Key characteristic

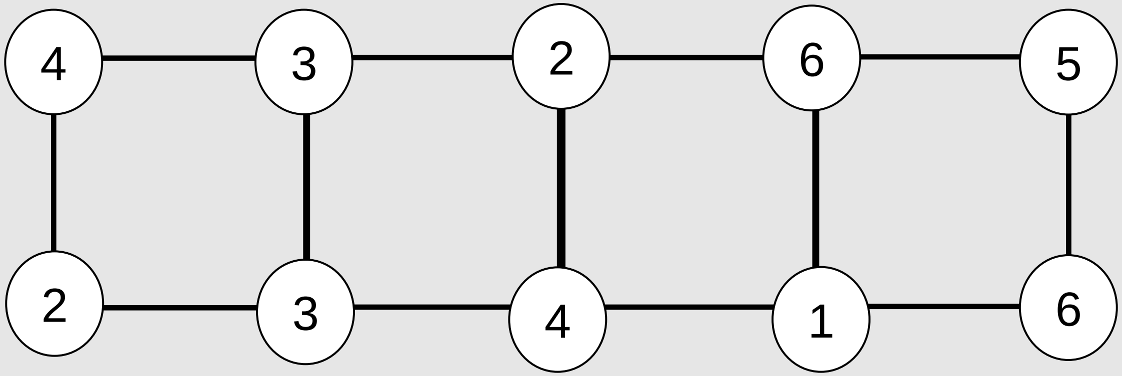
- Hard to find a solution
- Easy to verify a solution once you have one
- Other problems like this
  - Hamiltonian circuit
  - Clique
  - Subset sum
  - Graph coloring

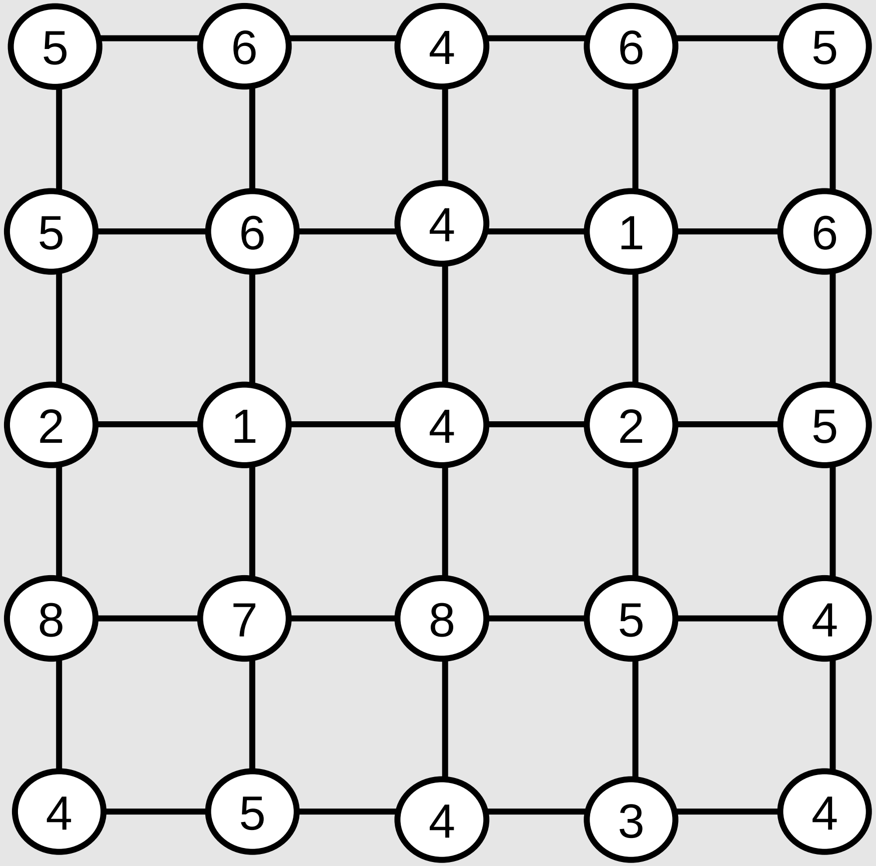
NP-Completeness

- Theory of Hard Problems
- A large number of problems are known to be equivalent
- Very elegant theory

Are there even harder problems?

- Simple game:
  - Players alternate selecting nodes in a graph
    - Score points associated with node
    - Remove nodes neighbors
  - When neither can move, player with most points wins




5 6 4 6 5
5 6 4 1 6
2 1 4 2 5
8 7 8 5 4
4 5 4 3 4

Competitive Facility Location

- Choose location for a facility
  - Value associated with placement
  - Restriction on placing facilities too close together
- Competitive
  - Different companies place facilities
    - E.g., KFC and McDonald’s

Complexity theory

- These problems are P-Space complete instead of NP-Complete
  - Appear to be much harder
  - No obvious certificate
    - G has a Maximum Independent Set of size 10
    - Player 1 wins by at least 10 points

Summary

- Scheduling
- Weighted Scheduling
- Bipartite Matching
- Maximum Independent Set
- Competitive Scheduling