Backtracking, fixed parameter tractability, planar graphs

Optimization Problems and Decision Problems

- Optimization problems we studied
  - Minimum Spanning Tree
  - Single Source Shortest Paths
  - Maximum Flow
- Decision Problems: Ask yes/no questions
- Search Problems: Decision problem + constructive answer

Minimum Spanning Tree Optimization Problem

- Input: A connected edge-weighted graph $G = (V, E)$
- Output: A minimum spanning tree $T = (V, E_T)$ for G

Spanning Tree Decision Problem

- Input: A connected edge-weighted graph $G = (V, E)$, a positive integer k
- Question: Does there exist a spanning tree $T = (V, E_T)$ for G of weight at most k?

Maximum Flow Optimization Problem

- Input: An st-flow network $G = (V, E)$
- Output: A maximum st-flow f for G

Flow
Decision Problem

- Input: An st-flow network $G = (V, E)$, a positive integer k
- Question: Does there exist an st-flow f for G with $|f| \geq k$?

Complexity Class $\mathbf{P}$

- Informal definition:
  - Contains all decision problems that can be solved in polynomial time in n, where n denotes the size of the input.
- A problem can be *solved* if there exists a algorithm that produces a correct answer to the problem for any input
- Formal definition: via Turing machines, CSC 320

Complexity Class $\mathbf{NP}$

- Informal definition:
  - Contains all decision problems for which candidate solutions can be verified in polynomial time in n, where n denotes the size of the input.
- $\mathbf{NP}$ stands for **Nondeterministic Polynomial** time
- Formal definition: via Turing machines, CSC 320

THE COMPLEXITY CLASS **NP**

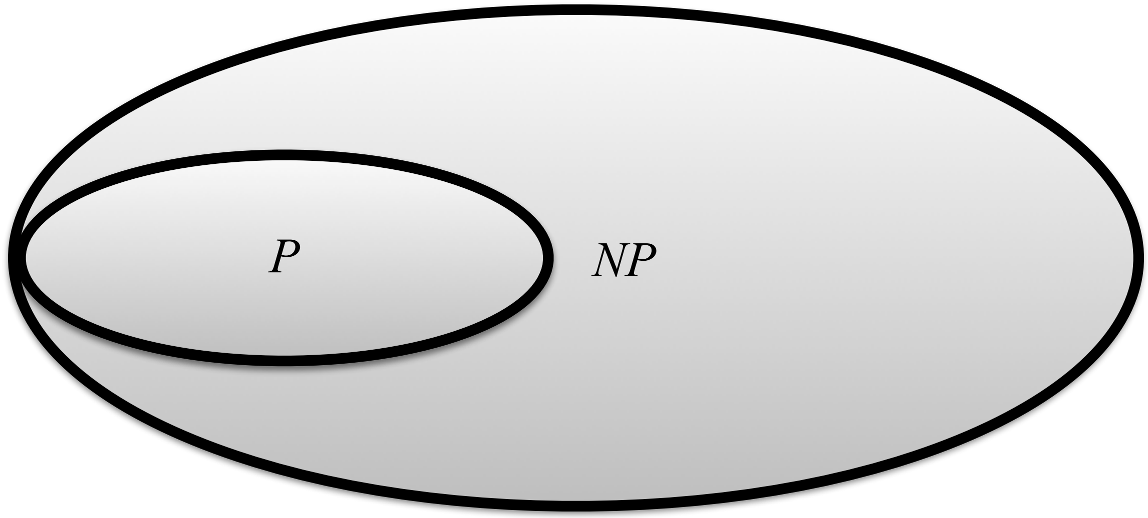
- **NP**: The class of all (decision/yes-no) problems for which candidate solutions can be verified in polynomial time

Yes!! Here is a solution. Check yourself!

Does this problem instance have a solution?

I did. You are ... (correct/ the solution is not correct).

P and NP

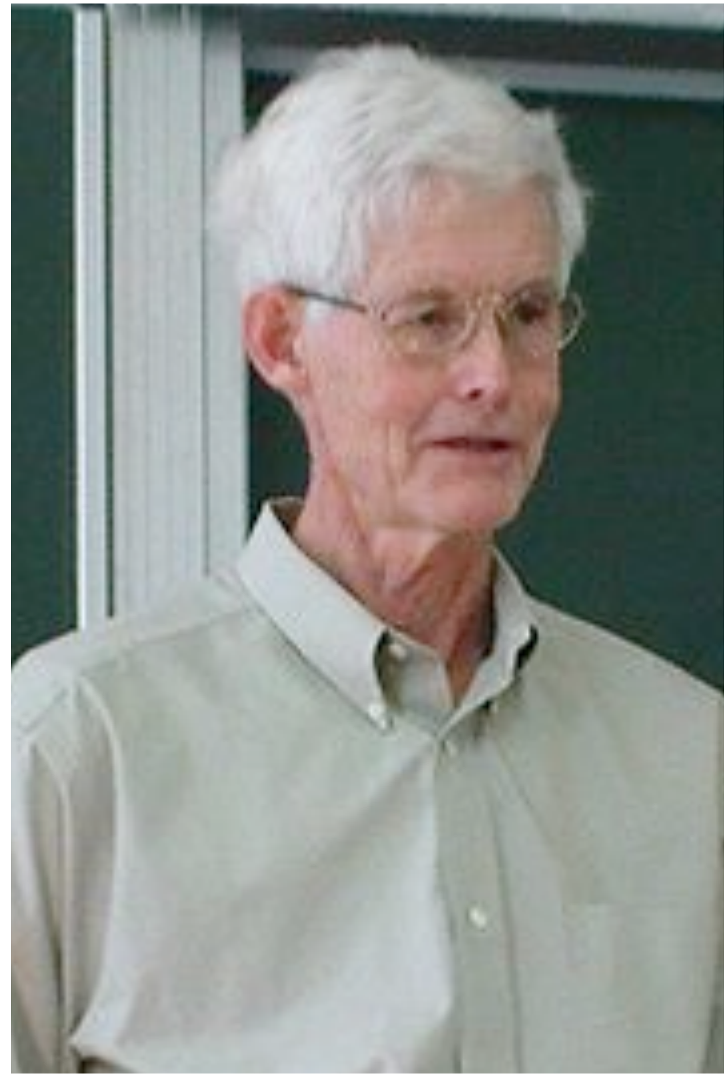


- We know: Every decision problem that is a member of the class **P** is also a member of the class **NP**

  - If a decision problem can be answered in polynomial time, also its candidate solutions (e.g., a spanning tree, an *st*-flow) can be verified in polynomial time

NP-COMPLETE PROBLEMS

- A problem in **NP** can have the property to be *NP-complete*
- There exists many **NP**-complete problems
- For none of the **NP**-complete problems a polynomial-time algorithm is known that solves it
- The property **NP**-complete was introduced by *Stephen Cook* (UofT)



NP-COMPLETE PROBLEMS ….

- are members of the class **NP** (the problems for which candidate solutions can be verified in polynomial time)

- Nobody knows whether or not they can be solved in polynomial time, that is none of the problems known to be **NP**-complete is also known to be in **P**

- the first one discovered is called Satisfiability (Stephen Cook found it); Cook's Theorem will be studied in CSC 320

IS P=NP?

- $\mathbf{NP}$-complete problems are special: If just one of them (and there are many!!) can be solved in polynomial time then all of them can be solved in polynomial time
- Thus: If just one of them is in $\mathbf{P}$, then $\mathbf{P} = \mathbf{NP}$ (worth a million dollars!)

MOST MATHEMATICIANS AND COMPUTER SCIENTISTS ...

- conjecture that **P** is **not** equals to **NP**
- One of the millenium problems of the Clay Mathematics Institute (http://www.claymath.org/millennium/P_vs_NP/)
- Prize for answering the question: US $1,000,000

SOME FAMOUS NP-COMPLETE PROBLEMS

- Vertex Cover
- Graph Coloring
- Traveling Salesman Problem
- Dominating Set
- Independent Set
- Clique
- Hamiltonian Circuit
- Eulerian Circuit
- Satisfiability

There exists a very large (growing) catalogue of **NP**-complete problems, most of them have important applications in other areas.

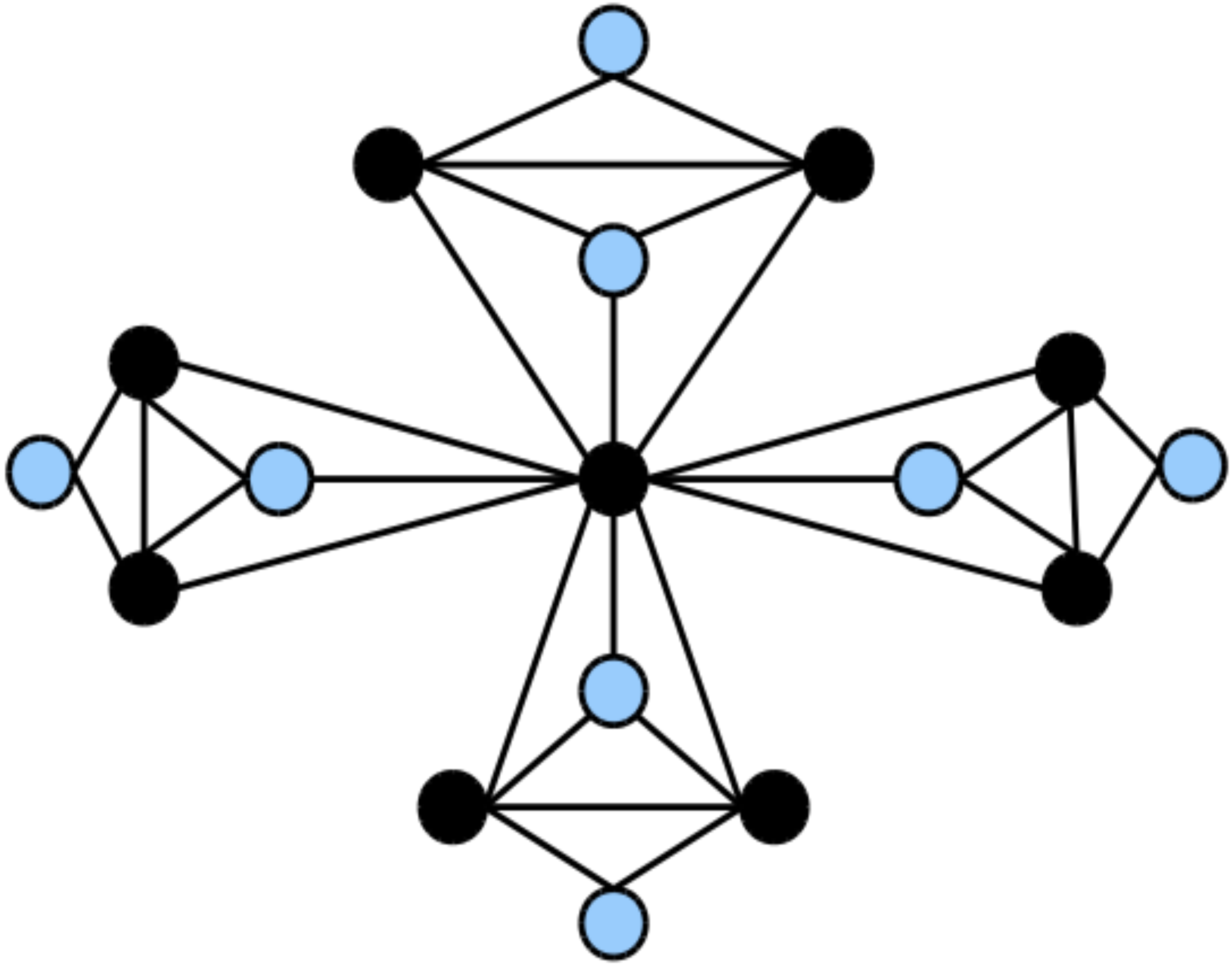
Two **NP**-complete Problems: Vertex Cover and Independent Set

- Both are graph problems
- For Vertex Cover, the goal is to determine a subset of the vertices such that each edge is incident to at least one of those
  - This subset should be small
- For Independent Set, the goal is to determine a (large) subset of vertices that does not contain adjacent pairs

Definitions

- Let $G = (V,E)$ be an undirected graph. Let $V' \subseteq V$. V' is a *vertex cover* for G if for each edge $(x,y) \in E$: $x \in V'$ or $y \in V'$.
    - We call an edge where at least one of its endpoints is in V' *covered.*
- Let $G = (V,E)$ be an undirected graph. Let $V' \subseteq V$. V' is a *independent set* for G if for each pair $x,y \in V'$: $(x,y) \notin E$.
    - We call two vertices that are not adjacent *independent*

A vertex cover (black) and an independent set (blue)



Vertex Cover Decision Problem

- Input: An undirected graph $G = (V,E)$, a positive integer k
- Question: Does there exist a vertex cover V' for G that is of size at most k, that is $|V'| \leq k$?

Minimum Vertex Cover Optimization Problem

- Input: An undirected graph $G = (V,E)$
- Output: A minimum vertex cover for G, that is a vertex cover that is of smallest possible size

Independent Set Decision Problem

- Input: An undirected graph $G = (V,E)$, a positive integer k

- Question: Does there exist a independent set V' for G that is of size at least k, that is $|V'| \geq k$?

Maximum Independent Set Optimization Problem

- Input: An undirected graph $G = (V,E)$
- Output: A maximum independent set for G, that is an independent set that is of largest possible size

A minimum vertex cover (black) and a maximum independent set (blue)