Announcements

- Exam 3 on Friday

NP-Completeness (Ch 8)

- NP-Problems
- Reductions
- NP-Completeness & NP-Hardness
- SAT
- Hamiltonian Cycle
- Zero-One Equations
- Knapsack

Brute Force Algorithms

For almost every problem we have seen there has been a (usually bad) naïve algorithm that just considers every possible answer and returns the best one.

- Is there a path from s to t in G?

- What is the longest common subsequence?

- What is the closest pair of points?

- Does G have a topological ordering?

NP

Such problems are said to be in <u>Nondeterministic Polynomial</u> time (NP).

NP-<u>Decision</u> problems ask if there is some object that satisfies a polynomial time-checkable property.

NP-<u>Optimization</u> problems ask for the object that maximizes (or minimizes) some polynomial time-computable objective.

Optimization vs. Decision

Note that these are not too different.

- Every decision problem can be phrased as an optimization problem (objective has value 1 if the object satisfies the condition and 0 otherwise).

- Every optimization problem has a decision form (can we find an example whose objective is more than x).

Examples of NP Problems

- SAT
- TSP
- Hamiltonian Cycle
- Knapsack
- Maximum Independent Set

SAT

Problem: Formula-SAT

Given a logical formula in a number of Boolean variables, is there an assignment to the variables that causes the formula to be true?

$$(x \vee y) \wedge (y \vee z) \wedge (z \vee x) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$$
$$x = \text{True}, y = \text{True}, z = \text{False}$$

$$(x \vee y) \wedge (y \vee z) \wedge (z \vee x) \wedge (\bar{x} \vee \bar{y}) \wedge (\bar{y} \vee \bar{z}) \wedge (\bar{z} \vee \bar{x})$$
No satisfying assignment.

Applications of SAT

- Circuit Design

- Logic Puzzles

- Cryptanalysis

Hamiltonian Cycle (in text as Rudruta Path)

Given an undirected graph G is there a cycle that visits every vertex exactly once?

General Knapsack

Recall knapsack has a number of items each with a weight and a value. The goal is to find the set of items whose total value is as much as possible without the total weight going exceeding some capacity.

Have algorithm that runs in polynomial time <u>in the weights</u>.

If weights are allowed to be large (written in binary), don't have a good algorithm.

Question: Decision vs. Optimization

Which of the following are NP-Decision problems?

A) SAT

B) Hamiltonian Cycle

C) General Knapsack

D) Maximum Independent Set

E) Travelling Salesman

Brute Force Search

- Every NP problem has a brute force search algorithm.

- Throughout this class we have looked at problems with algorithms that substantially improve on brute force search.

- Does every NP problem have a better-than-brute-force algorithm?

P vs. NP

$1,000,000 Question: Is P = NP?

Is it the case that every problem in NP has a polynomial time algorithm?

- If yes, every NP problem has a reasonably efficient solution.

- If not, some NP problems are fundamentally difficult

Most computer scientists believe P ≠ NP. (But proving anything is very very hard)

Hard Problems

In practice, at least <u>some</u> problems in NP appear to be hard. Despite decades of trying, people still don't know particularly good algorithms.

So if you have a problem, how do you know if it is hard or not?

- Can search for algorithms (to show it's easy).

- Can try to prove that it is hard, but this is extremely difficult.

- Can try to relate its difficulty to that of other problems.
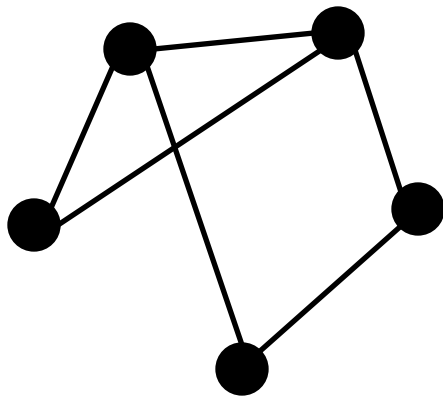
Reductions

Reductions are a method for proving that one problem is <u>at least as hard</u> as another.
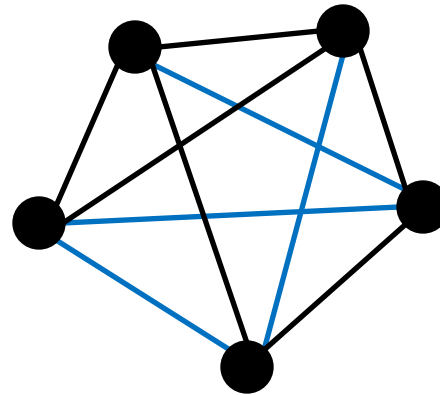
How do we do this?

We show that <u>if</u> there is an algorithm for solving A, then we can use this algorithm to solve B. Therefore, B is no harder than A.

Hamiltonian Cycle → TSP
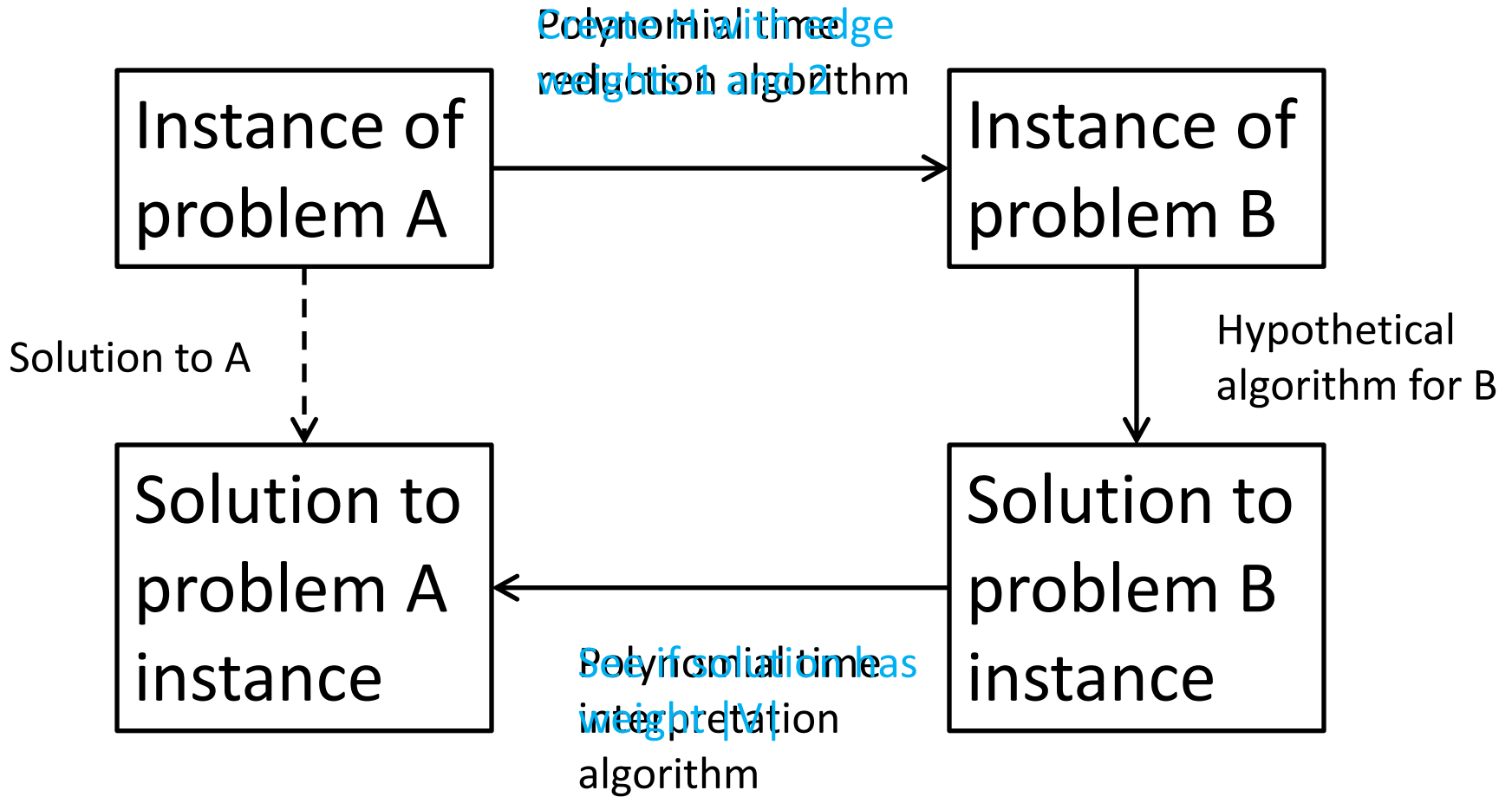
Hamiltonian
Cycle Instance

TSP Instance

Cost = 1

Cost = 2

Formally

- Given Ham. Cycle Instance G

- Create TSP Instance H
  - Edges in G are cost 1
  - Edges not in G are cost 2

- Solve TSP instance
  - Have a cycle of cost |V| in H if and only if Hamiltonian cycle in G

- Use answer to solve initial problem

If, we have an algorithm that solves TSP, we can use it to solve Ham. Cycle.

Reduction A → B

Instance of problem A

Instance of problem B

Create H with edge
~~Polynomial time~~
weights 1 and 2
~~reduction algorithm~~

Solution to A

Solution to
problem A
instance

Hypothetical
algorithm for B

Solution to
problem B
instance

See if solution has
~~Polynomial time~~
weight |V|
~~interpretation~~
algorithm

Reduction A → B

If we have algorithms for reduction and interpretation:

- Given an algorithm to solve B, we can turn it into an algorithm to solve A.

- This means that A might be <u>easier</u> to solve than B, but cannot be <u>harder</u>.