Announcements

- Exam 3 on Friday
  - In class
  - Assigned seating
  - 6 one-sided pages of notes
  - On Dynamic programming/Huffman codes/MSTs

Last Time

- NP Problems and NP completeness
  - 3SAT
  - MIS

NP

Such problems are said to be in Nondeterministic Polynomial time (NP).

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

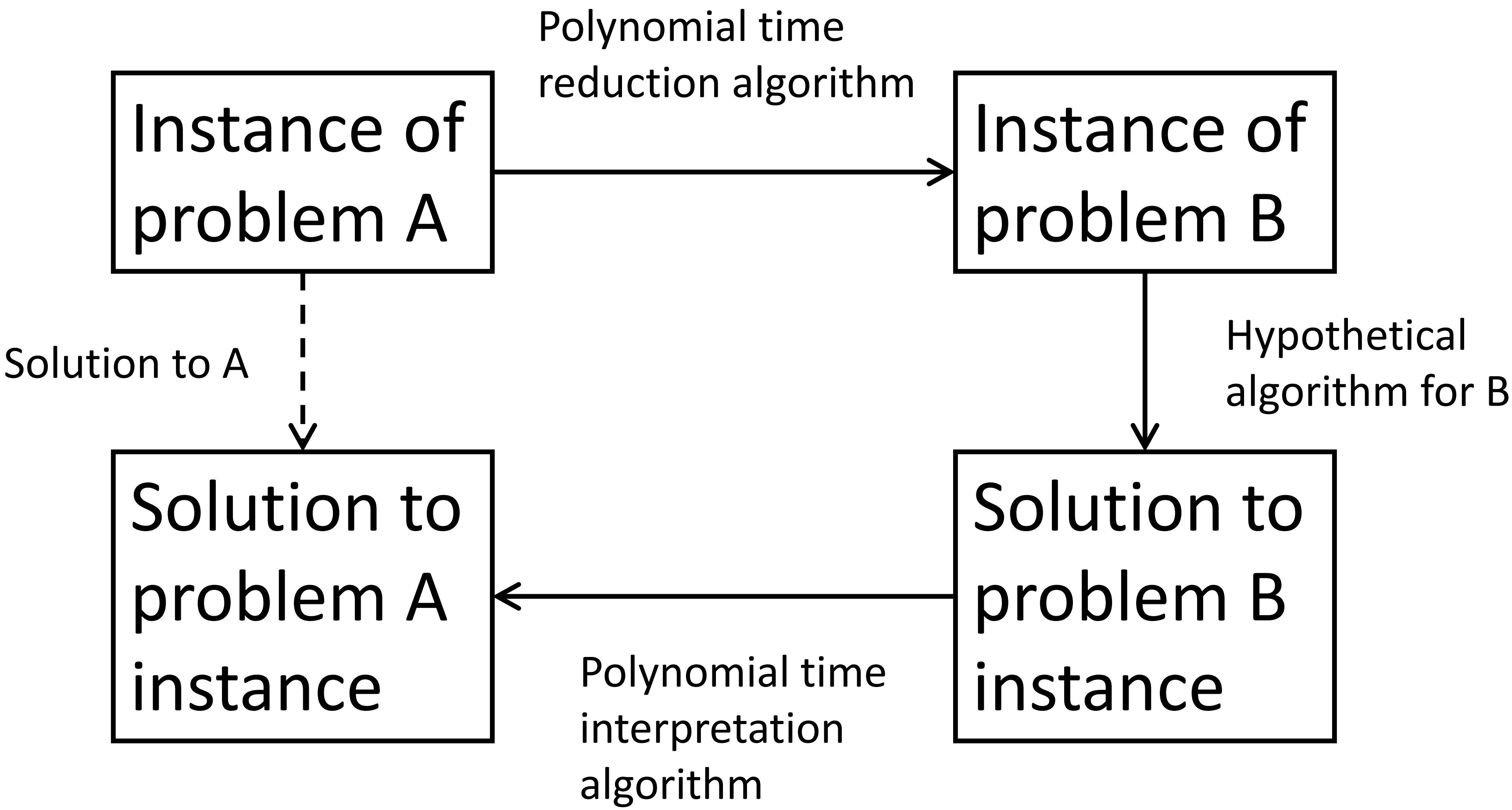
NP-Optimization problems ask for the object that maximizes (or minimizes) some polynomial time-computable objective.

Reductions

Reductions are a method for proving that one problem is at least as hard as another.

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

Reduction A → B



NP-Complete

Circuit-SAT is our first example of an NP-Complete problem. That is a problem in NP that is at least as hard as any other problem in NP.

3-SAT

3-SAT is a special case of formula-SAT where the formula is an AND of clauses and each clause is an OR of at most 3 variables or their negations.

NP-Complete!

Another Look at 3-SAT

<u>Lemma:</u> A 3-SAT instance is satisfiable if and only if it is possible to select one term from each clause without selecting both a variable and its negation.

Intermediate Problems

To prove our more complicated reductions, it will help to have the correct problem to prove reductions from.

A convenient problem is the one the book calls Zero-One Equations.

Today

- More NP-Completeness Reductions
  - Zero-One Equations
  - Knapsack
  - Hamiltonian Cycle

Zero-One Equations

Problem: Given a matrix A with only 0 and 1 as entries and b a vector of 1s, determine whether or not there is an x with 0 and 1 entries so that

$$Ax = b.$$

Zero-One Equations

Problem: Given a matrix A with only 0 and 1 as entries and b a vector of 1s, determine whether or not there is an x with 0 and 1 entries so that

$$Ax = b.$$

This problem is clearly in NP. We will show that it is NP-Complete.

Example

$$\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Example

$$\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Equivalently, do there exist x_1, x_2, $x_3 \in \{0,1\}$ so that

$x_1+x_3 = 1$

$x_1+x_2 = 1$

Example

$$\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Equivalently, do there exist x_1, x_2, $x_3 \in \{0,1\}$ so that

$x_1+x_3 = 1$

$x_1+x_2 = 1$

Generally, this is what a ZoE looks like. A bunch of sets of x_is that need to add to 1.

3-SAT → ZOE

Basic Idea:

- Use the one term from each clause formulation of 3-SAT.

3-SAT → ZOE

Basic Idea:

- Use the one term from each clause formulation of 3-SAT.
- Create one variable for each term to denote whether or not it has been selected.

3-SAT → ZOE

Basic Idea:

- Use the one term from each clause formulation of 3-SAT.
- Create one variable for each term to denote whether or not it has been selected.
- Add equations to enforce exactly one term from each clause, no contradictory terms selected.

Example

$$(x \vee y \vee z) \wedge (\bar{x} \vee y) \wedge (\bar{y} \vee \bar{x})$$

Example

$$(x \vee y \vee z) \wedge (\bar{x} \vee y) \wedge (\bar{y} \vee \bar{x})$$

x_1 x_2 x_3 x_4 x_5 x_6 x_7

Example

$$(x \vee y \vee z) \wedge (\bar{x} \vee y) \wedge (\bar{y} \vee \bar{x})$$

x_1 x_2 x_3 x_4 x_5 x_6 x_7

One term per clause:

$x_1 + x_2 + x_3 = 1$

$x_4 + x_5 = 1$

$x_6 + x_7 = 1$

Example

$$(x \lor y \lor z) \land (\bar{x} \lor y) \land (\bar{y} \lor \bar{x})$$

x_1 x_2 x_3 x_4 x_5 x_6 x_7

One term per clause:

$x_1 + x_2 + x_3 = 1$

$x_4 + x_5 = 1$

$x_6 + x_7 = 1$

No Contradictions:

$x_1 + x_4 \leq 1$

$x_1 + x_7 \leq 1$

$x_2 + x_6 \leq 1$

$x_5 + x_6 \leq 1$

Example

$$(x \vee y \vee z) \wedge (\bar{x} \vee y) \wedge (\bar{y} \vee \bar{x})$$

x_1 x_2 x_3 x_4 x_5 x_6 x_7

One term per clause:

$x_1 + x_2 + x_3 = 1$

$x_4 + x_5 = 1$

$x_6 + x_7 = 1$

No Contradictions:

$x_1 + x_4 \leq 1$

$x_1 + x_7 \leq 1$

$x_2 + x_6 \leq 1$

$x_5 + x_6 \leq 1$

Not allowed inequalities

Example

$$(x \vee y \vee z) \wedge (\bar{x} \vee y) \wedge (\bar{y} \vee \bar{x})$$

x_1 x_2 x_3 x_4 x_5 x_6 x_7

One term per clause:

$x_1 + x_2 + x_3 = 1$

$x_4 + x_5 = 1$

$x_6 + x_7 = 1$

Replace
$a + b \leq 1$ with
$a + b + c = 1$

No Contradictions:

$x_1 + x_4 \leq 1$

$x_1 + x_7 \leq 1$

$x_2 + x_6 \leq 1$

$x_5 + x_6 \leq 1$

Not allowed inequalities

Example

$$(x \vee y \vee z) \wedge (\bar{x} \vee y) \wedge (\bar{y} \vee \bar{x})$$

x_1 x_2 x_3 x_4 x_5 x_6 x_7

One term per clause:

$x_1 + x_2 + x_3 = 1$

$x_4 + x_5 = 1$

$x_6 + x_7 = 1$

No Contradictions:

$x_1 + x_4 + x_8 = 1$

$x_1 + x_7 + x_9 = 1$

$x_2 + x_6 + x_{10} = 1$

$x_5 + x_6 + x_{11} = 1$

Replace
$a + b \leq 1$ with
$a + b + c = 1$

General Construction

- Create one variable per term
- For each clause, create one equation
- For each pair of contradictory term, create an equation with those two and a new variable

Another Way of Looking at ZOE

Recall if $A = [v_1\ v_2\ v_3\ \ldots\ v_n\ ]$,

$Ax = x_1 v_1 + x_2 v_2 + x_3 v_3 + \ldots + x_n v_n$.

Another Way of Looking at ZOE

Recall if $A = [v_1\ v_2\ v_3\ \ldots\ v_n\ ]$,

$Ax = x_1\ v_1 + x_2\ v_2 + x_3\ v_3 + \ldots + x_n\ v_n$.

Example:

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Another Way of Looking at ZOE

Recall if $A = [v_1\ v_2\ v_3\ \ldots\ v_n\ ]$,

$Ax = x_1\ v_1 + x_2\ v_2 + x_3\ v_3 + \ldots + x_n\ v_n$.

Example:

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

x_1* [1 0 0 1] +

x_2* [0 0 1 1] +

x_3* [1 1 1 0]

= [1 1 1 1]

Another Way of Looking at ZOE

Recall if $A = [v_1\ v_2\ v_3\ \dots\ v_n\ ]$,

$Ax = x_1\ v_1 + x_2\ v_2 + x_3\ v_3 + \dots + x_n\ v_n$.

Example:

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

```
x1*[  1  0  0  1  ]  +
x2*[  0  0  1  1  ]  +
x3*[  1  1  1  0  ]
----------------
=   [ 1  1  1  1  ]
```

What if we treated these as numbers rather than vectors?

Another Way of Looking at ZOE

Recall if $A = [v_1\ v_2\ v_3\ \ldots\ v_n]$,

$Ax = x_1 v_1 + x_2 v_2 + x_3 v_3 + \ldots + x_n v_n$.

Example:

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

```
x1*    1  0  0  1      +
x2*    0  0  1  1      +
x3*    1  1  1  0
----------------
=       1  1  1  1
```

What if we treated these as numbers rather than vectors?

Reduction

If the numbers are represented in a large enough base that carrying is impossible, we have a solution to the vector equation if and only if we have a solution to the number equation.

Subset Sum

Problem: Given a set S of numbers and a target number C, is there a subset $T \subseteq S$ whose elements sum to C.

Subset Sum

Problem: Given a set S of numbers and a target number C, is there a subset $T \subseteq S$ whose elements sum to C.

Alternatively: Can we find $x_y \in \{0,1\}$ so that

$$\sum_{y \in S} x_y y = C.$$

Subset Sum

Problem: Given a set S of numbers and a target number C, is there a subset $T \subseteq S$ whose elements sum to C.

Alternatively: Can we find $x_y \in \{0,1\}$ so that

$$\sum_{y \in S} x_y y = C.$$

Reduction: ZOE → Subset Sum.

Knapsack

Subset Sum is pretty closely related to (non-repeated) knapsack.

Knapsack

Subset Sum is pretty closely related to (non-repeated) knapsack.

Subset Sum wants to find a set of values so that the weights equal the capacity.

Knapsack

Subset Sum is pretty closely related to (non-repeated) knapsack.

Subset Sum wants to find a set of values so that the weights equal the capacity.

Knapsack wants to find a set of values so that the weights are at most the capacity and the value is large.

Subset Sum → Knapsack

- Create Knapsack problem where for each item Value(item) = Weight(item).

Subset Sum → Knapsack

- Create Knapsack problem where for each item Value(item) = Weight(item).
- Maximizing value is the same as maximizing weight (without going over capacity).

Subset Sum → Knapsack

- Create Knapsack problem where for each item Value(item) = Weight(item).
- Maximizing value is the same as maximizing weight (without going over capacity).
- We can achieve value = capacity if and only if there is a subset of the items with total weight equal to capacity.

Knapsack is NP-Hard

3-SAT → ZOE → Subset Sum → Knapsack

Knapsack is NP-Hard

3-SAT → ZOE → Subset Sum → Knapsack

Wait. Didn't we have a polynomial time DP for knapsack?

Knapsack is NP-Hard

3-SAT → ZOE → Subset Sum → Knapsack

Wait. Didn’t we have a polynomial time DP for knapsack?

Our algorithm was polynomial in the total weight, which in this case is exponential.

One Final Reduction

The last reduction we are going to show is ZOE → Hamiltonian Cycle. This will show that both Hamiltonian Cycle and TSP are NP-Complete/Hard.