CS 154

Lecture 16: More Examples of NP-Complete Problems – and coNP

CS 154

Final Exam:

March 19, 7pm-10pm

Hewlett 200

One double-sided letter-sized sheet of notes

Subset Sum

Given: Set $S = \{a_1, \ldots, a_n\}$ of positive integers
positive integer t

Is there an $S' \subseteq \{1,\ldots,n\}$ such that $t = \sum_{i \in S'} a_i$?

SUBSET-SUM $= \{(S, t) \mid \exists S' \subseteq S \text{ s.t. } t = \sum_{I\,:\,a\_i \in S'} a_i \}$

Theorem (cs161): There is a $O(n \cdot t)$ time algorithm for solving SUBSET-SUM.

But t can be specified in (log t) bits… this isn't an algorithm that runs in polytime in the input!

VC $\leq_P$ SUBSET-SUM

Want to reduce a *graph* to a *set of numbers*

Given (G, k), let $E = \{e_0,\ldots,e_{m-1}\}$ and $V = \{1,\ldots,n\}$

The subset sum instance (S, t) will have $|S| = n+m$

"Edge numbers":

For every $e_j \in E$, put $b_j = 4^j$ in S

"Node numbers":

For every $i \in V$, put $a_i = 4^m + \sum_{j\,:\,i \in e_j} 4^j$ in S

Set the target number: $t = k \cdot 4^m + \sum_{j=0}^{m-1} (2 \cdot 4^j)$

For every $e_j \in E$, put $b_j = 4^j$ in S

For every $i \in V$, put $a_i = 4^m + \sum_{j\,:\,i \in e_j} 4^j$ in S

Set $t = k \cdot 4^m + \sum_{j=0}^{m-1} (2 \cdot 4^j)$

Claim: If $(G,k) \in$ VC then $(S,t) \in$ SUBSET-SUM

Suppose $C \subseteq V$ is a VC with k vertices.

Let $S' = \{a_i : i \in C\} \cup \{b_j : |e_j \cap C| = 1\}$

S' = the *node numbers* corresponding to nodes in C, plus the *edge numbers* corresponding to edges covered *only once* by C.

Claim: The sum of all numbers in S' equals t!

Think of the numbers as being in "base 4"…
as vectors with m+1 components

For every $e_j \in E$, put $b_j = 4^j$ in S

For every $i \in V$, put $a_i = 4^m + \sum_{j\,:\,i \in e_j} 4^j$ in S

Set $t = k \cdot 4^m + \sum_{j=0}^{m-1} (2 \cdot 4^j)$

Claim: If $(S,t) \in$ SUBSET-SUM then $(G,k) \in$ VC

Suppose $C \subseteq V$ and $F \subseteq E$ satisfy

$$\sum_{i \in C} a_i + \sum_{e_j \in F} b_j = t$$

Claim: C is a vertex cover of size k.

Proof: Subtract out the b_j numbers from the above sum.

What remains is a sum of the form:

$$\sum_{i \in C} a_i = k \cdot 4^m + \sum_{j=0}^{m-1} (c_j \cdot 4^j)$$

where each $c_j > 0$. But c_j = number of nodes in C covering e_j

This implies C is a vertex cover!

The Knapsack Problem

Given: $S = \{(p_1,c_1)\ldots, (p_n,c_n)\}$ of pairs of positive integers

a cost budget C

a profit target P

Is there an $S' \subseteq \{1,\ldots,n\}$ such that

$(\sum_{i \in S'} p_i) \geq P$ **and** $(\sum_{i \in S'} c_i) \leq C$ **?**

Define KNAPSACK = {(S, C, P) | the answer is yes}

A classic economics problem!

Theorem: KNAPSACK is NP-complete

KNAPSACK is NP-complete

KNAPSACK is in NP?

Theorem: SUBSET-SUM $\leq_P$ KNAPSACK

Proof: Given an instance $(S = \{a_1,\ldots,a_n\}, t)$ of SUBSET-SUM, create a KNAPSACK instance:

For all i, set $(p_i, c_i) := (a_i, a_i)$

Define $T = \{(p_1, c_1),\ldots, (p_n, c_n)\}$

Define $C := P := t$

Then, $(S,t) \in$ SUBSET-SUM $\Leftrightarrow$ $(T,C,P) \in$ KNAPSACK

Subset of S that sums to t =

Solution to the Knapsack instance!

The Partition Problem

Given: Set $S = \{a_1, \ldots, a_n\}$ of positive integers

Is there an $S' \subseteq S$ such that $(\sum_{a\_i \in S'} a_i) = (\sum_{a\_i \in S-S'} a_i)$?

(Formally, PARTITION is the set of all S such that the answer to this question is yes.)

In other words, is there a way to partition S into two parts, with equal sum in both parts?

A problem in fair division

Theorem: PARTITION is NP-complete

PARTITION is NP-complete

(1) PARTITION is in NP

(2) SUBSET-SUM $\leq_P$ PARTITION

Given: Set $S = \{a_1, \ldots, a_n\}$ of positive integers
positive integer t

Output $T := \{a_1, \ldots, a_n, 2A-t, A+t\}$, where $A := \sum_i a_i$

Claim: $(S,t) \in$ SUBSET-SUM $\Leftrightarrow T \in$ PARTITION

Given: Set $S = \{a_1,\ldots, a_n\}$ of positive integers
positive integer t

Output $T := \{a_1,\ldots, a_n, 2A-t, A+t\}$, where $A := \sum_i a_i$

Claim: $(S,t) \in$ SUBSET-SUM $\Leftrightarrow T \in$ PARTITION

What's the sum of all numbers in T? **4A**

Therefore: $T \in$ PARTITION
$\Leftrightarrow$ There is a $T' \subseteq T$ that sums to 2A.

Proof of $(S,t) \in$ SUBSET-SUM $\Rightarrow T \in$ PARTITION:

If $(S,t) \in$ SUBSET-SUM, let $S' \subseteq S$ sum to t.
Then $S' \cup \{2A-t\} \subseteq T$ sums to 2A, so $T \in$ PARTITION

Given: Set $S = \{a_1,\dots, a_n\}$ of positive integers
positive integer t

Output $T := \{a_1,\dots, a_n, 2A-t, A+t\}$, where $A := \sum_i a_i$

Claim: $(S,t) \in$ SUBSET-SUM $\Leftrightarrow T \in$ PARTITION

$T \in$ PARTITION $\Leftrightarrow$ There is a $T' \subseteq T$ that sums to $2A$.

Proof of $T \in$ PARTITION $\Rightarrow (S,t) \in$ SUBSET-SUM

If $T \in$ PARTITION, let $T' \subseteq T$ be a subset that sums to $2A$.

Observation: Exactly *one* of $\{2A-t, A+t\}$ is in T'.

If $(2A-t) \in T'$, then $T' - \{2A-t\}$ sums to t.

But $T' - \{2A-t\}$ is a subset of S! So $(S,t) \in$ SUBSET-SUM

If $(A+t) \in T'$, then $(T - T') - \{2A-t\}$ sums to $(2A - (2A-t)) = t$

Note that $(T - T') - \{2A-t\}$ is a subset of S.
Therefore $(S,t) \in$ SUBSET-SUM

The Bin Packing Problem

Given: Set $S = \{a_1, \ldots, a_n\}$ of positive integers,
a bin capacity B, and a target integer K.
Can we partition S into K subsets such that each subset sums to at most B?

Is there a way to pack the items of S into K bins, with each bin having capacity B?

Ubiquitous in shipping and optimization

Theorem: BIN PACKING is NP-complete

BIN PACKING is NP-complete

BIN PACKING is in NP?

Theorem: PARTITION $\leq_P$ BIN PACKING

Proof: Given an instance $S = \{a_1,\ldots, a_n\}$ of PARTITION, create an instance of BIN PACKING with:

$$S = \{a_1,\ldots, a_n\}$$
$$B = (\textstyle\sum_i a_i)/2$$
$$k = 2$$

Then, $S \in$ PARTITION $\Leftrightarrow$ $(S,B,k) \in$ BIN PACKING:

Partition of S into two equal sums = Solution to the Bin Packing instance!

Two Problems

Let G denote a graph, and s and t denote nodes.

SHORTEST PATH

= {(G, s, t, k) |

G has a simple path of length < k from s to t }

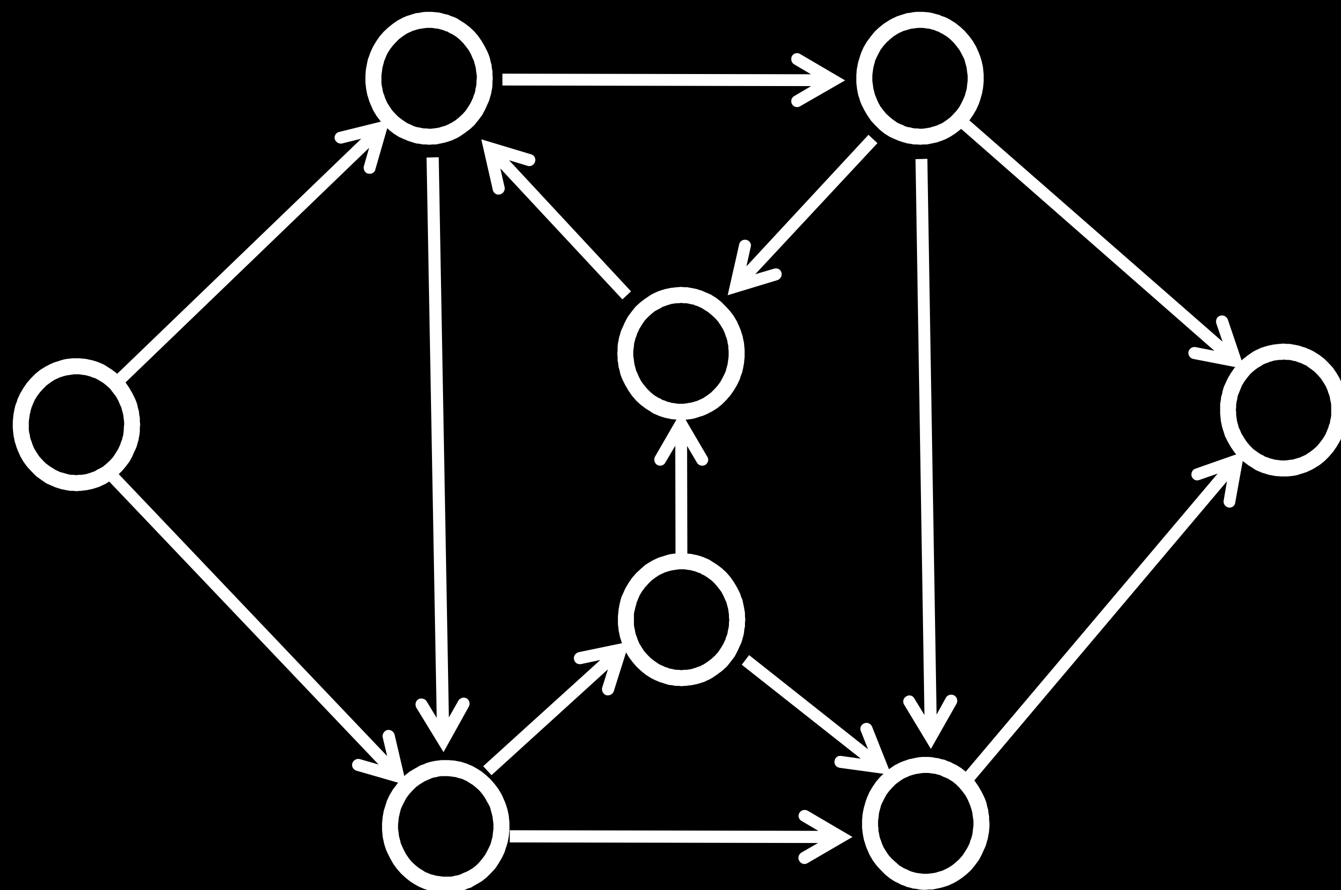
LONGEST PATH

= {(G, s, t, k) |

G has a simple path of length > k from s to t }

Are either of these in P? Are both of them?

Hamiltonian Path



HAMPATH = { (G,s,t) | G is an directed graph with a Hamiltonian path from s to t}

Theorem: HAMPATH is NP-Complete

(1) HAMPATH $\in$ NP

(2) 3SAT $\leq_P$ HAMPATH

See Sipser for the proof

HAMPATH $\leq_P$ LONGEST-PATH

LONGEST-PATH
= {(G, s, t, k) |
G has a simple path of length > k from s to t }

Can reduce HAMPATH to LONGEST-PATH by observing:

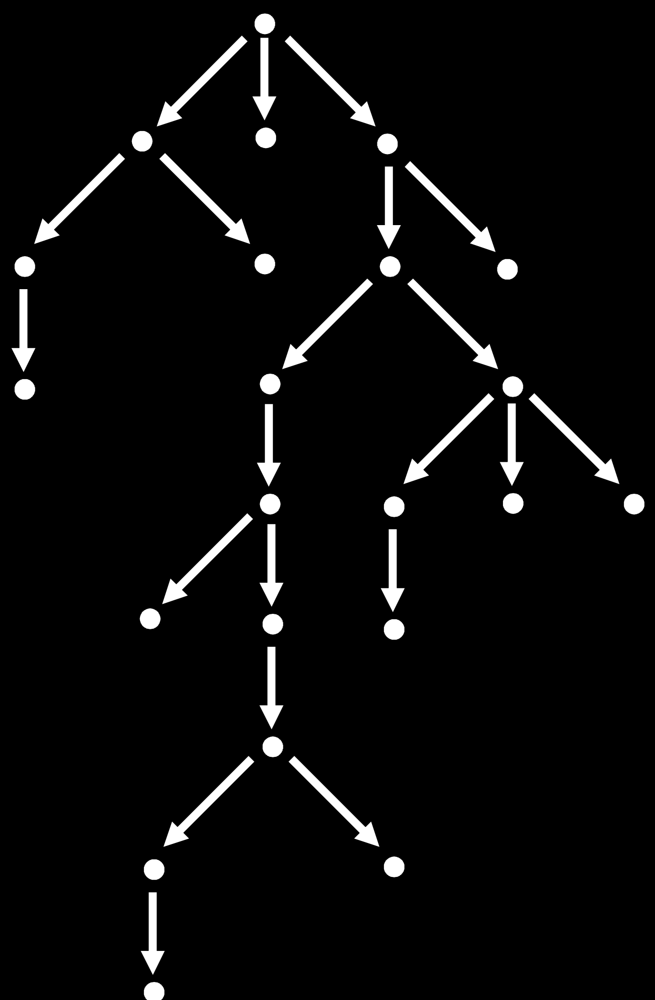
(G, s, t) $\in$ HAMPATH
$\Leftrightarrow$ (G, s, t, |V|) $\in$ LONGEST-PATH

Therefore LONGEST-PATH is NP-hard.

coNP and Friends

Definition: coNP = $\{ L \mid \neg L \in NP \}$

What does a coNP computation look like?



A *co-nondeterministic* machine has multiple computation paths, and has the following behavior:

- **the machine accepts if *all paths reach* accept state**
- **the machine rejects if *at least one path* reaches reject state**

Is P $\subseteq$ coNP?

Yes!

L $\in$ P implies that $\neg$L $\in$ P
(hence $\neg$L $\in$ NP)

In general, *deterministic* complexity classes are closed under complement

Is NP = coNP?

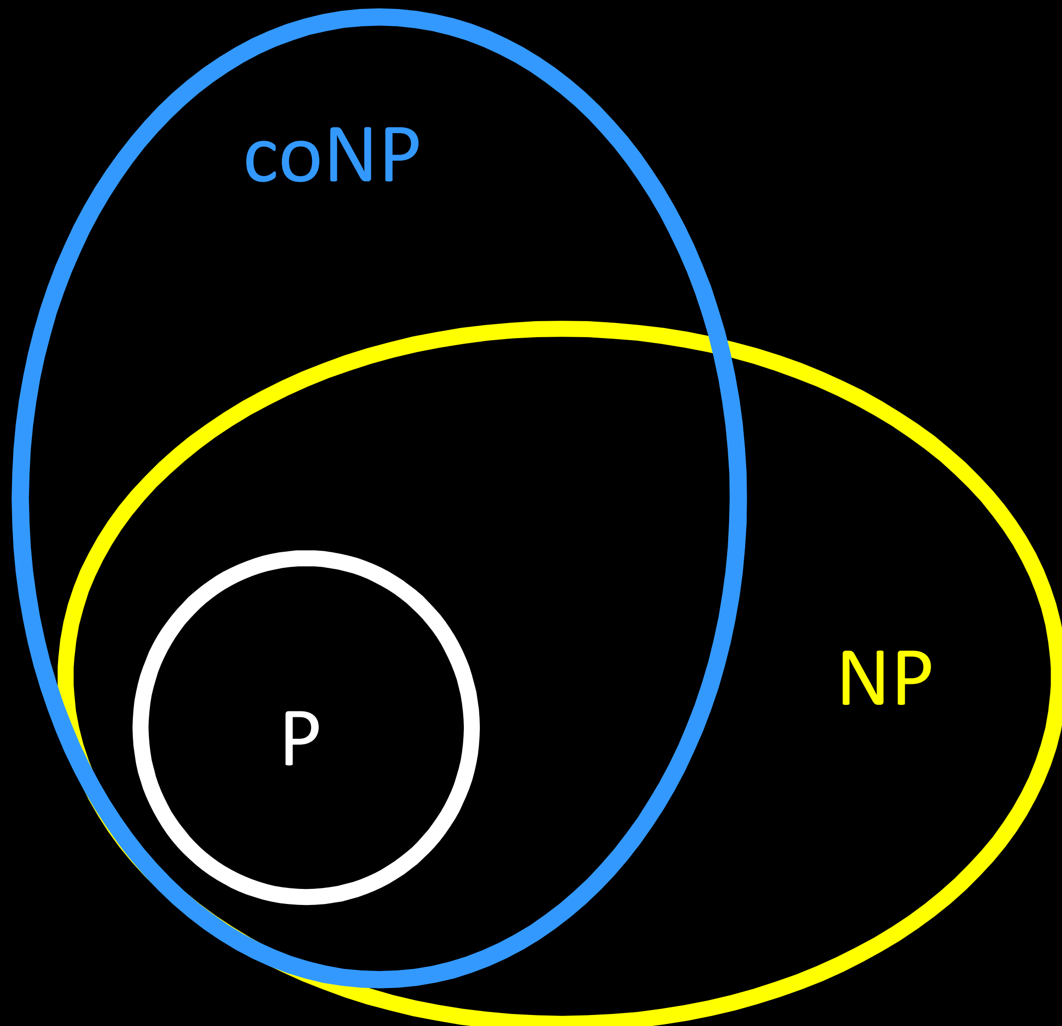
Nobody knows! It is believed that NP $\neq$ coNP

Could we define something similar for P?

Definition: A $\in$ coP if and only if $\neg$A $\in$ P

P = coP

since a deterministic decision algorithm for $\neg$A can be used to decide A by just flipping accept/reject states


coNP
NP
P

Definition: A language B is coNP-complete if

1. $B \in$ coNP

2. For every A in coNP, there is a polynomial-time reduction from A to B

(B is coNP-hard)

UNSAT = { ϕ | ϕ is a Boolean formula and *no* variable assignment satisfies ϕ }

Theorem: UNSAT is coNP-complete

Proof: UNSAT $\in$ coNP because $\neg$UNSAT $\approx$ SAT

(2) UNSAT is coNP-hard:

Let A $\in$ coNP. We show A $\leq_P$ UNSAT

On input w, transform w into a formula ϕ using Cook-Levin via an **NP machine for $\neg$A**

$w \in \neg A \Rightarrow \phi \in$ SAT

$\boxed{w \notin A \Rightarrow \phi \notin \text{UNSAT}}$

$w \notin \neg A \Rightarrow \phi \notin$ SAT

$\boxed{w \in A \Rightarrow \phi \in \text{UNSAT}}$

TAUT = { ϕ | ϕ is a Boolean formula and every variable assignment satisfies ϕ }

= $\{\phi \mid \neg\phi \in$ UNSAT$\}$

TAUT is coNP-complete

(1) TAUT $\in$ coNP, since $\neg$TAUT $\in$ NP

(2) TAUT is coNP-hard:

We show UNSAT $\leq_P$ TAUT:

Given formula ϕ, output $\neg\phi$

Is P = NP $\cap$ coNP?

THIS IS AN OPEN QUESTION!