CompSci 162
Spring 2023 Lecture 24:
Subset Sum is $\mathcal{NP}$-complete

Definition and Start

Problem Statement: given a set S of numbers and a target T, does a subset of S add up to T?

Example: $\langle\{4, 11, 16, 21, 27\}, 25\rangle \in$ SUBSET-SUM

Prove that Subset Sum is $\mathcal{NP}$-complete.

Start: it is in $\mathcal{NP}$

do not forget!

Plan for the Proof

- We know 3-Sat is $\mathcal{NP}$-complete.
- We want to show Subset Sum is also.

Make solver for 3-Sat that uses Subset Sum

- Must be polynomial time (not including time for Subset Sum)
- Create set S and target T
- Step 1: any TVA possible by interpretation
- Step 2: Make a TVA that satisfies each clause
  - First we did if $\oplus$
  - Then we will fix for $\vee$

Start of Reduction

Use SUBSET SUM to get a truth value assignment on n variables

	x_1	x_2	x_3	x_4	x_5	C_1
v_1	1	0	0	0	0	0
v_1'	1	0	0	0	0	0
v_2	0	1	0	0	0	1
v_2'	0	1	0	0	0	0
v_3	0	0	1	0	0	1
v_3'	0	0	1	0	0	0
v_4	0	0	0	1	0	1
v_4'	0	0	0	1	0	0
v_5	0	0	0	0	1	0
v_5'	0	0	0	0	1	0
T	1	1	1	1	1	1

- Keep v_i means $X_i = \text{T}$
- Keep v_i' means $X_i = \text{F}$
- Cannot keep both

We want a satisfying assignment

$$\phi = (x_2 \vee x_3 \vee x_4)(\overline{x_2} \vee x_3 \vee \overline{x_4})(\overline{x_1} \vee x_3 \vee x_5)\dots$$

Suppose instead of $\vee$ we wanted exactly one

	x_1	$\dots$	x_5	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
v_1	1	$\dots$	0	0	0	0					
v_1'	1	$\dots$	0	0	0	1					
v_2	0	$\dots$	0	1	0	0					
v_2'	0	$\dots$	0	0	1	0					
v_3	0	$\dots$	0	1	1	1					
v_3'	0	$\dots$	0	0	0	0					
v_4	0	$\dots$	0	1	0	0					
v_4'	0	$\dots$	0	0	1	0					
v_5	0	$\dots$	1	0	0	1					
v_5'	0	$\dots$	1	0	0	0					
T	1	1	1	1	1	1	1	1	1	1	1

How to convert to $\vee$?

- Previous setup makes each clause 1T and 2F
- We want any non-zero T
- Idea 1: each column target in C_i is $1, 2$, or 3
  - If any of these are satisfied, done
  - Problem: that's 3^k calls to SUBSET SUM
- Idea 2: Introduce some "slack variables"
  - Effect: variable target

But it is ∨

added #s

$$\phi = (x_2 \vee x_3 \vee x_4)(\overline{x_2} \vee x_3 \vee \overline{x_4})(\overline{x_1} \vee x_3 \vee x_5)\ldots$$

	x_1	...	x_5	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
s_1	0	0	0	1	0	0	0	0	0	0	0
s_1'	0	0	0	2	0	0	0	0	0	0	0
s_2	0	0	0	0							
s_2'	0	0	0	0							
s_3	0	0	0	0							
s_3'	0	0	0	0							
s_4	0	0	0	0							
s_4'	0	0	0	0							
T	1	1	1	4							

T is $n+k$ digits long

Putting it together...

Remember, our algorithm is:

- We are given an instance of 3-SAT
- Create the $2n$ "boolean variables" v_i, v_i'
- Create $2k$ "clause variables" s_i s_i'
- Create a value T
- Call any correct SUBSET SUM algorithm

Could this produce *false negatives* or *false positives*?

Running time

- You maybe saw $\mathcal{O}(nT)$ for Subset Sum
  - n : size of vector of numbers S
  - T : **value** of target number
- We started with 3-SAT, n variables, k clauses
  - What is $|S|$? $2n + 2k$
  - What is $\approx$ value of T? $\approx 10^{n+k}$
- What is the running time of our 3-SAT solver?

$$\mathcal{O}\left((n+k) \cdot 10^{n+k}\right)$$

Problems with Subset Sum

You might have seen $\mathcal{O}(nT)$ time algorithm.

- Suppose T is `std::uint8_t`, 1 second timing.
- How long with `std::uint16_t`? 4.2 minutes
- How long with `std::uint32_t`? ≈ 192 days
- How long with `std::uint64_t`? ≈ 4B x 192 days

  ≈ 2B years