Subset-Sum and Knapsack

☆ For vertex cover, see previous notes.

SUBSUM = $\{\langle S, t\rangle \mid S \subseteq \mathbb{N}, t \in \mathbb{N}$, $\exists$ a selection of elements of S which sum to $t\}$.

We have previously shown many SAT-like problems are NP-complete. Really, we have shown these are all variations of SAT. Using a transformation of SAT to a graph, we were able to show INDSET was NP-complete. Using INDSET, once we had one graph problem, the next two became quite easy. Today we prove that subset sum is NP-complete. This will allow us to prove many more such problems to be NP-complete. (~~Do~~ you notice that Subsetsum seems kind of like knapsack.) First, some intuition on addition. How can we relate boolean logic with addition? If we add powers of two notice how it looks like as a bit string. $2^5 + 2^3 = 100000 + 1000 = 101000$. The output is really like a boolean or of the bits. We will set our numbers up to avoid any carry operations, but we will use each digit of our sum value t independently.

If $t = 1$ and our set contains $(y=1, z=1)$, we can choose only one of y or z, but not both. We use this to ensure we don't choose both $x_i, \overline{x_i}$, and neither must have exactly one.

First we prove SUBSETSUM $\in$ NP. Our polytime verifier takes as input problem = "(S, t)" and a set $v_1 \ldots v_k \in S$. It just adds them to confirm that $\sum v_i = t$.

assume $x \vee \bar{x}$ not appear in clause

2

$$(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3)$$

	x_1	x_2	x_3	C_1	C_2	C_3	
v_1	1	0	0	1	0	0	1040
$\bar{v}_1$	1	0	0	0	1	1	1029
v_2	0	1	0	1	1	0	276
$\bar{v}_2$	0	1	0	0	0	1	257
v_3	0	0	1	1	0	1	81
$\bar{v}_3$	0	0	1	0	1	0	68
s_1	0	0	0	1	0	0	16
s_1'	0	0	0	1	0	0	16
s_2	0	0	0	0	1	0	4
s_2'	0	0	0	0	1	0	4
s_3	0	0	0	0	0	1	1
s_3'	0	0	0	0	0	1	1
t	1	1	1	3	3	3	1407

create v_i, v_i' for each x_i the first bits of $v_i, \bar{v}_i$ are set so that you can only choose exactly one of them the right bits of each of $v_i, \bar{v}_i$ are set as if they satisfy that specific clause. We set t to be the first bit all 1. This only enforces we choose only one of $x_i, \bar{x}_i$, and if it is chosen. We have the rightmost digits be set to be the number of clauses, $k=3$ we add a few slack variables so our sum is exact.

$\phi \in 3SAT \Rightarrow \langle S, t \rangle \in SUBSETSUM$ by construction

$\phi \notin 3SAT$, no selection of the variables will satisfy so sum of the right digits can be chosen to equal 1 (+ slack) so $\langle S, t \rangle \notin SUBSETSUM$

This transformation takes polynomial time. A few for loops and bit shifts. We conclude $3SAT \leq_p SUBSETSUM$. Since $SUBSETSUM \in NP$, $SUBSETSUM$ is NP-complete

Subset sum looks suspiciously close to knapsack. In fact, we can prove knapsack is NP-complete. Let us first formalize it as a decision problem.

$$KNAPSACK = \{\langle I, V, W\rangle \mid I = \text{set of items } (v_i, w_i)$$
$$\sum v_i \geq V$$
$$\sum w_i \leq W$$
$$\exists \text{ a selection } X \text{ of } I \text{ s.t.}$$

KNAPSACK $\in$ NP. Our polytime verifier takes as input a problem $\langle I, V, W\rangle$, and a witness, $X \subset I$. Computes sums and checks inequalities against V, W in polytime.

Now we perform a reduction. If $\langle S, t\rangle \in$ SUBSETSUM where $S = \{s_1, s_2 \ldots s_n\}$ set $I = \{(s_1, s_1), (s_2, s_2), \ldots (s_n, s_n)\}$
That is, set $w_i = s_i$ and $v_i = s_i$. Also set $W = V = t$.

$$\langle S, t\rangle \rightarrow \langle S \times S, t, t\rangle$$

If $\langle S, t\rangle \in$ SUBSETSUM, $\exists$ a selection of its ~~items~~ numbers to sum to t, so then $\exists$ a selection of weights to sum $\leq t$, and values $\geq t$.

$$\sum s_i = t \Rightarrow \sum w_i \leq t \text{ and } \sum v_i \geq t$$

If $\langle S, t\rangle \notin$ SUBSETSUM, no selection $\sum s_i = t$ for every selection. So $\forall$ selections, either $\sum s_i > t$ or $\sum s_i < t$. If $\sum s_i > t$, then $\sum w_i > W$. If $\sum s_i < t$ then $\sum v_i < V$. Either way, $\langle S \times S, t, t\rangle \notin$ KNAPSACK.

We conclude SUBSETSUM $\leq_p$ KNAPSACK.

Since KNAPSACK $\in$ NP, we conclude knapsack is NP-complete.