The **Subset Sum** problem is as follows: given \( n \) non-negative integers \( w_1, \ldots, w_n \) and a target sum \( W \), the question is to decide if there is a subset \( I \subset \{1, \ldots, n\} \) such that \( \sum_{i \in I} w_i = W \). This is a very special case of the **Knapsack** problem: In the **Knapsack** problem, items also have values \( v_i \), and the problem was to maximize \( \sum_{i \in I} v_i \) subject to \( \sum_{i \in I} w_i \leq W \). If we set \( v_i = w_i \) for all \( i \), **Subset Sum** is a special case of the **Knapsack** problem that we discussed when considering dynamic programming. In that section, we gave an algorithm for the problem that runs in time \( O(nW) \). This algorithm works well when \( W \) isn’t too large, but we note that this algorithm is not a polynomial time algorithm. To write down an integer \( W \), we only need \( \log W \) digits. In is natural to assume that all \( w_i \leq W \), and so the input length is \( (n + 1) \log W \), and the running time of \( O(nW) \) is not polynomial in this input length.

In this handout we show that, in fact, **Subset Sum** is NP-complete. First we show that **Subset Sum** is in NP.

**Claim 1.** **Subset Sum** is in NP.

**Proof.** Given a proposed set \( I \), all we have to test if indeed \( \sum_{i \in I} w_i = W \). Adding up at most \( n \) numbers, each of size \( W \) takes \( O(n \log W) \) time, linear in the input size. \( \square \)

To establish that **Subset Sum** is NP-complete we will prove that it is at least as hard as \( \text{SAT} \).

**Theorem 1.** \( \text{SAT} \leq \text{Subset Sum} \).

**Proof.** To prove the claim we need to consider a formula \( \Phi \), an input to \( \text{SAT} \), and transform it into an equivalent input to **Subset Sum**. Assume \( \Phi \) has \( n \) variables \( x_1, \ldots, x_n \), and \( m \) clauses \( c_1, \ldots, c_m \), where clause \( c_j \) has \( k_j \) literals.

We will define our **Subset Sum** problem using a very large base \( B \), so will write numbers as \( \sum_{j=0}^{m} a_j B^j \), and we set the base \( B \) as \( B = 2 \max_j k_j \), which will make sure that additions among our numbers will never cause a carry.

Written in base \( B \) the digits \( i = 1, \ldots, n \) will correspond to the \( n \) variables \( x_1, \ldots, x_n \), and the goal of these digits will be to make sure that we set each variable to either true or false (and not both). We’ll have two numbers \( w_i \) and \( w_{i+n} \) corresponding to the variable \( x_i \) being set true or false, and digit \( i \) will make sure that we use one of \( w_i \) and \( w_{i+n} \) in any solution. To so this, we set the \( i \)th digits of \( W, w_i \) and \( w_{i+n} \) to be 1, and set this digit in all other numbers to be 0.

The next \( m \) digits will correspond to the \( m \) clauses, and the goal digit \( n + j \) is to make sure that the \( j \)th clause is satisfied by our setting of the variables.

The target value will be \( W = \sum_{i=1}^{n} B^i + \sum_{j=1}^{m} k_j B^{n+j} \).

We start by defining \( 2n \) numbers, for of each of the literals \( x_i \) and \( \overline{x}_i \). The digits \( 1, \ldots, n \) will make sure that any subset that sums to \( W \) will use only exactly 1 of the two numbers \( x_i \) and \( \overline{x}_i \), and the the next \( m \) digits will aim to guarantee that each clause is satisfied. We will need a few additional numbers that we’ll define later.

The number corresponding to literal \( x_i \) is as follows \( w_i = B^i + \sum_{j: x_i \in c_j} B^{n+j} \), while the number corresponding to literal \( \overline{x}_i \) is \( w_{i+n} = B^i + \sum_{j: \overline{x}_i \in c_j} B^{n+j} \). If we add a set of \( n \) numbers
corresponding to a satisfying truth assignment for $\Phi$, we get a sum of the form $\sum_{i=1}^{n} B_i + \sum_{j=1}^{m} b_j B^{n+j}$ where $b_j$ is the number of literals true in clause $c_j$. Since this was a satisfying assignment, we must have $b_j \geq 1$.

As a final detail, we will add $k_j - 1$ copies of the number $B^{n+j}$ for all clauses $c_j$. This now defined our subset sum problem, with target $W$ and the $2n + \sum_{j=1}^{m} (k_j + j - 1)$ numbers defined, suing these additional numbers will allow us to exactly reach $W$.

To prove that this a valid reduction, we need to establish two claims below establishing the if and the only if direction of the proof respectively.

Claim 2. If the SAT problem defined by formula $\Phi$ is solvable, than the Subset Sum problem we just defined with $2n - m + \sum_{j=1}^{m} k_j$ numbers is also solvable.

Proof. Suppose we have a satisfying assignment for the formula $\Phi$, first consider adding the numbers that correspond to the true literals. We used exactly one of $w_i$ and $w_{i+n}$ so will have 1 in the $i$th digit, and get a sum that is of the form $\sum_{i=1}^{n} B_i + \sum_{j=1}^{m} a_j B^{n+j}$.

Further, will have $1 \leq a_i \leq k_i$, where $a_i$ is at least 1, as the assignment satisfied the formula, so at least one of the numbers added has a 1 in the $(n+j)$th digit, and at most $k_j$ as even adding all numbers at most $k_j$ of them has a 1 in the $(n+j)$th digit. In particular, with $B > k_j$, there will be no carries.

To make this sum to exactly $W$, we add $k_j - a_j$ copies of the number $B^{n+j}$ we added at the end of the construction.

Next we need to prove the other direction:

Claim 3. If the Subset Sum problem we just defined with $2n - m + \sum_{j=1}^{m} k_j$ numbers is solvable, than the Subset Sum defined by formula $\Phi$ is solvable.

Proof. First notice that for any subset we may add, there will never be a carry in any digit. To see why, note that all numbers to be summed have all digits 0 or 1; for digit $i = 1, \ldots, n$ we have two numbers with a 1 in that digit $w_i$ and $w_{i+n}$; the 0th digit is always 0; and for the $n+j$th digit we have exactly $2 \max_j k_j - 1$ numbers that have a 1 on that digits: $k_j$ corresponding to the $k_j$ literals in the clause, and $k_j - 1$ extra numbers $B^{n+j}$ we added at the end. So even is we add all of the numbers, we cannot cause a carry in any of the digits!

Based on the above observation about not having any carries, to get the number $W$, we need to find a subset $I$ that has exactly the right number of 1’s in every digit. First focus on digits $1, \ldots, n$. This digit in $W$ is a 1, and the two numbers that have a 1 in this digit are $w_i$ and $w_{i+n}$, to to sum to $W$, we must use exactly one of these, let $I' \subseteq I$ corresponding to the literals. This
shows that the selected numbers among the first $2n$ of them correspond to a truth assignment of the variables $x_1, \ldots, x_n$.

Finally, we need to show that this truth assignment satisfied the formula $\Phi$. Consider the sum $W' = \sum_{j \in I'} w_j$, just adding the subset $I$ that corresponds to variables. Note that $W' = \sum_{i=1}^n B_i + \sum_{j=1}^m a'_j B^{n+j}$ with $a'_j \leq k_j$. We need to show that that $a'_j \geq 1$ which will prove that we have a satisfying assignment. Recall that the subset $I$ sums to exactly $W$. To be able to extend $I'$ with a subset of the additional numbers to sum to $W$, we must have $a'_j \geq 1$ as there are only $k_j - 1$ copies of $B^{n+j}$. \qed