20190122
Supplemental (Optional) Notes

We have seen that Partition reduces to Subset Sum very easily, using the "special case reduces to general case" approach. Our goal here is to show that Subset Sum reduces to Partition.

To show this, we need to find a polynomial-time, answer-preserving transformation that will turn any instance of Subset Sum into an instance of Partition.

Let's think about what we start with: an instance of Subset Sum, i.e. a set of integers S and a target integer k. We want to transform this into an instance of Partition in such a way that the Yes/No answer is preserved. Suppose the answer to the instance of SS is "Yes" - that means there **IS** a subset that sums to k. If we let t be the total sum of the set, then the other subset (i.e. all the elements not in the set that sums to k) must sum to t-k. We need to "do something" to the original set of integers S so that we have a new set that can be partitioned into two equal sets.

It turns out that this is very easy, and there are many ways to do it. We'll look at just one. Consider modifying S by adding two new **very large** values to it – so large that each of them is greater than the sum of all values in S. Call these new values X and Y, and call the modified set S'. Treat S' as an instance of Partition. If the answer to this Partition instance is "Yes", then S' can be split into two subsets with equal sums. Because X and Y are both very large, the new value X is in one of these subsets (call it P_X), and the new value Y is in the other (call it P_Y). Can we choose X and Y so that the other values in X's partition sum to k?

We need $$\sum_{P_X} = X + k = \sum_{P_Y} = Y + t - k$$

and from this we can see that we need $Y = X + 2k - t$

Even requiring that X and Y are very large with respect to the other values, we see that there are infinitely many X, Y pairs that satisfy this. One such pair is

$X = 100t - k$ and $Y = 99t + k$,

but we could just as easily use $X = 500t$ and $Y = 499t + 2k$ etc.

With X and Y defined and added to S to give S′, we can show that S′ has a partition if and only if S has a subset that sums to k.

We need to formalize the transformation and show that it is answer-preserving.

Transformation: Given instance (S,k) of Subset Sum,
compute t = total sum of S
compute X = 100*t – k and Y = 99*t + k
create S' = S ∪ {X, Y} (i.e. add the elements X and Y to S)
(note that the total sum of S' is 200*t)

This transformation clearly requires no more than O(n) time, where n is the size of S

Now we show the transformation is answer-preserving:

1. Suppose the answer to Subset Sum(S,k) is “Yes”. Then S contains a subset Q that sums exactly to k. Q is also a subset of S'. Including X in Q gives a subset of S′ that sums to 100*t. The other elements of S′ must also sum to 100*t (because the total sum of S' is 200*t), so the answer to Partition(S') is “Yes”.

2. Suppose the answer to Partition(S') is “Yes”. Then S' can be split into two subsets, each summing to 100*t (because the total sum of S' is 200*t). X and Y cannot both be in the same subset, so X must be in a subset with values from S that sum to k. Thus the answer to Subset Sum(S,k) is “Yes”.

Therefore Subset Sum $\propto$ Partition.

This is an example of transforming a general case into a specific case – but as mentioned before, we can't always do this.

A brief note on proving the "answer-preserving" property of a reduction. Suppose we are reducing an instance of problem A to an instance of problem B. We need to show that

(1) if the answer for A was Yes then the answer for B will be Yes

AND

(2) if the answer for A is No then the answer for B will be No.

The first part is usually pretty easy, but the second part may be harder because it may be difficult to describe the properties of an instance with a No answer. So instead of this, we often replace (2) by this logically equivalent statement:

(2') if the answer for B is Yes then the answer for A must have been Yes

This is what we did in the proof that Subset Sum reduces to Partition.