4. Given a set of n binary integers S = $\{a_1, ..., a_n\}$ and a target binary integer t, find the subset R $\subseteq$ S that sums up to t. Given, a black box that tells us if there is a subset R $\subseteq$ S that sums up to t.

   Intuition: We can remove items from S until its sum eventually equals the subset that sums to t.

   Proof by construction:

   TM M = "on input $\langle$S, t $\rangle$where S is a set of binary integers and t is an integer target

   0. Run oracle on $\langle$S, t $\rangle$. If oracle rejects, reject.

   1. repeat for every number in S:

      2. remove the number from S, write to tape 2.

      3. run the black box on updated S.

      4. If oracle accepts, erase tape 2 and move to next number on tape 1. If oracle rejects, write the number back to the tape 1 from tape 2, and erase tape 2, move to next number on tape 1.

   Validation:

   Our machine works because we remove numbers that are unnecessary to sum to integer t. Since we go through every number on tape 1 and remove every one that is unnecessary to achieve the target sum, we will eventually end with a set where every number is necessary to sum to t.

   This machine clearly runs in polynomial time:

   step O is only based on Oracle which runs in polynomial time.

   step 1 runs in O(n) since does something once for every item in the list.

   step 2. removing a single number should be O(1)

   step 3. runs in polynomial time given by the problem

   step 4. runs in O(1) as we are using the results of oracle and checking the results.

   Step 3 runs based on step 1. So we have O(n) * O(n^k) which is O(n^{k+1}) therefore is still polynomial.

   Since our highest degree is O(n^{k+1}) which is polynomial we know that this algorithm is *in* P.