(Sol) HW11 More complexity

KT 8.*, and problem solving in general

1. (Credit: This is slightly modified from Problem 34.2 in CLRS book.) Bonnie and Clyde.
   Bonnie and Clyde have just robbed a bank. They have a big bag of money and want to divide it up. For each of the following scenarios, either give a polynomial-time algorithm, or prove that the problem is NP-Complete. The input in each case is a list of n items in the bag, along with the value of each. You may assume that the total is even.

   a. [3 points] There are n coins, but only two different denominations: Some coins are worth x dollars, and some are worth y dollars. They wish to divide the money exactly evenly.

      Answer: There is a polynomial-time algorithm for this scenario. If there are even number of x-dollar coins and y-dollar coins, then the solution is trivial. Otherwise, we can use the algorithm below:

      Without loss of generality, let Bonnie be the one that takes more x-dollar coins than the other. There is a solution if and only if the amount Bonnie gets more in x-dollar coins is equal to that Clyde gets more in y-dollar coins. Let T be the amount Bonnie gets. She takes a coins with value x and b coins with value y, so $T = ax + by$. $(T - a * x)$ as to be a multiple of y, and there are at least $(T - a * x)/y$ y-dollar coins available for Clyde after Bonnie takes b y-dollar coins. We just need to check all possible values of a.

      Our function Divide will take 4 parameters, the value and number of each coin type, and return the number of x-dollar coins the number of y-dollar coins that Bonnie should take. (Clyde takes the remainder of the coins.)

      Example of “No Solution” case: {1,1,4,4,4}

```
Divide(x, numx, y, numy)
  T = (x*numx + y*numy)/2
  for i = 1 to numx
    if (T −i*x) % y == 0 and (T −i*x) / y <= numy/2
```

```
        return i, (T − i*x) / y
    return “No Soluton”
```

b. [3 points] There are n coins, with an arbitrary number of different denominations, but each denomination is a non-negative integer power of 2, i.e., the possible denominations are 1 dollar, 2 dollars, 4 dollars, etc. They wish to divide the money evenly.

   Answer: In this instance, we can evenly divide the money in polynomial time. Since the larger denominations are always divisible by the smaller denominations, the difference in the larger denominations can be made up using the smaller denominations as long as there are enough coins in smaller denominations. We sort the coins in the decreasing order and keep adding the coins to the side that has less sum.

```
//L is the list of all n coins
CanMatch(L)
  B ← {}
  C ← {}
  Bsum ← 0
  Csum ← 0
  Sort L in the decreasing order
  for each element e in L:
    if Bsum < Csum then {
      append e to C
      Csum ← Csum + e
    } else {
      append e to B
      Bsum ← Bsum + e
    }
  if Bsum != Csum
    return “No match”
  else
    return B, C
  //you can return only B for Bonnie, and
  //let Clyde take the remaining coins, too. In such case,
  //there is no need to maintain C.
```

c. [3 points] There are n cashier's checks. They wish to divide the checks so that they each get the exact same amount of money.
   Hint: The subset-sum equality problem: Given a set of numbers S and target t, is there a subset of numbers in S whose sum is equal to t? This decision

problem is NP-Complete.
You may wish to show that any subset-sum equality problem can be reduced into a subset sum problem where the target t is equal to 1/2 of the sum of all elements in the set.

Answer: This one is NP-Complete. Proof by reduction from the Subset-Sum equality problem: Given a set of numbers S and target t, is there a subset of numbers in S whose sum is equal to t? For simplicity, we will name the subset-sum equality problem where the target t is equal to 1/2 of the sum of all elements in the set as the half-subset problem.

Formally, the half-subset problem is defined as follows: given a set of numbers S' is there a subset A', where the sum of all elements in A' is equal to 1/2 of the sum of all elements in S'? (Note that when the sum of all elements in A' is equal to 1/2 of the sum of all elements in S', the sum of all elements in $S' \setminus A'$ is also equal to 1/2 of the sum of all elements in S'.)

1. NP: Given a set of numbers S' and its subset A', adding all numbers in S' and those in A' and checking if sum of S' is twice sum of A' takes $O(|S'| + |A'|)$ = $O(|S'|)$. So the half-subset problem can be verified in polynomial time, so it is in NP.

2. Construction: Given any instance of the subset-sum equality problem (S, t) , we will construct an instance of the half-subset problem $(S$'$)$. Let s be the sum of all elements in S.

   Case 1: If $t < s/2$, $S' = S \cup \{s - 2t\}$

   Case 2: Otherwise ($t \geq s/2$), $S' = S \cup \{2t - s\}$

   In both cases the construction is done in $O(1)$, if S and S' are stored in linked list or HashSet.

3. Conversion: The solution to the half-subset problem (S') is a subset A' whose elements' sum is 1/2 of the sum of all elements in S'. Without loss of generality, let A' be the subset(partition) of S' without the new element, $s - 2t$ in case 1 and $2t - s$ in case 2, and B' be S' $\setminus A'$, including the new element. Let t' be 1/2 of the sum of all elements in S'.

   Case 1: Let $B = B' \setminus \{s - 2t\}$. The sum of elements in B is $(s + (s - 2t))/2 - (s - 2t) = s - t - s + 2t = t$, so B is the solution to the original subset sum problem (S, t).

Case 2: The sum of elements in A is $(s + (2t - s))/2 = t$, so A is the solution to the original subset sum problem (S, t).

In both cases the conversion is done in $O(1)$, if A' and B' are stored in HashSet.

So the half-subset problem is in NP, and a known NP-complete problem (the subset-sum equality problem) is polytime reducible to this problem, so the half-subset problem is NP-complete.

d. [3 points] There are n checks as in part (c), but this time they are willing to accept a split in which the difference is no larger than 100 dollars.
Hint: You can reduce the problem in (c) to the problem you wish to prove is NP-Complete.

Answer: This is also NP-Complete. We will reduce the half-subset problem to this problem. For simplicity, we will name this problem as a 100-subset problem. Formally, the 100-subset problem is defined as follows: given a set of numbers S, is there a subset A such that the difference between the sum of all elements in A and the sum of all elements in $S \setminus A$ is no larger than 100.

NP: Given an instance of 100-subset problem (S) and a subset A, it takes $O(|A| + |S| - |A|)$ = $O(|S|)$ to add all the elements and compute the difference, so it takes a polynomial time to verify a solution. Thus the 100-subset problem is NP.
Construction: Let S be a set of numbers in a half-subset problem. We create a new S' as follows: Start with an empty set S'. For each $i \in S$, add $i * 101$ to S'. This takes $O(|S|)$.

Conversion: Let A' be the solution to the 100-subset problem using S'. The difference between the sum of all elements in A' and that of the elements in S' $\setminus$ A' is no greater than 100. Start with an empty set A. For each $i \in A$', add $i/101$ to A. The difference between the sum of all elements in A and that of the elements in $S \setminus A$ is no greater than 100/101, i.e. the same, so A is the solution to the half-subset problem. (We assume that all numbers in S are whole numbers. If not, we can multiply a larger number to make the difference small enough.) This takes $O(|A|) = O(|S|)$.

The 100-subset problem is NP and a known NP-complete problem is polytime reducible to the 100-subset problem, so the 100-subset problem is NP-complete.