34-2 Bonnie and Clyde

(a) Polynomial time solvable.

Assume that there m coins worth x dollars, n-m coins worth y dollars.

Then the total amount is $m*x+(n-m)*y$ dollars, denoted as S. Thus each of them wants to take $S/2$ dollars.

Note that Bonnie can either take 0 coin worth x dollars, or 1 coin worth x dollars, or 2 coins worth x dollars…, or at most m coins worth x dollars. Since there are only two , we can just check after Bonnie taking coins worth x dollars, if coins worth y dollars can make up the rest of the amount.

For example, if Bonnie takes no coins worth x dollars, then check if $S/2 \bmod y$ and if $S/(2*y) <= (n-m)$. If Bonnie takes 1 coin worth x dollars, then check if $(S/2-x) \bmod y$ and if $(S/2-x)/y <= (n-m)$. There are at most $m+1$ possibility to check, because Bonnie can take at most m coins. In each possibility, constant number of operations is needed, thus this is a polynomial algorithm.

(b) Sort all the coins by decreasing order. Give Bonnie the largest coin, then give Clyde coins until Clyde has the same amount as Bonnie. This will always happen if the rest of money is no less than their difference, because the denominations of coins are power of 2. When they have the same amount, give Bonnie the largest coin in the remaining coins, repeat the above process until all coins are gone. If after giving Bonnie one coin C, the sum of remaining coins is less than C. Then we can not find an evenly division. Otherwise, we can.

(c) NP-Complete

This problem is actually partition problem.

First, given a division, summing up the checks given to Bonnie to see if the sum is $S/2$ verifies the solution. (S is the sum of amount of all checks)

Then, we reduce Subset-Sum problem to this problem.

Assume in Subset-Sum problem, we are given a set of numbers, denoted as P.

Let S' be the sum of all numbers in P, and t be the target number.

Form the set $P' = P \bigcup \{S' - 2t\}$, P has a subset summing up to t, if and only if P' can be partitioned into two sets with equal sum. The reduction is obviously polynomial.

Since Subset-Sum problem is NPC, thus our problem is also NPC.

(d) NP-Complete

First, given a division, u just check whether the sum of checks given to Bonnie and Clyde have a difference less than 100. Polynomial time verifiable.

Then reduce the problem in (c) which is known NPC to this problem.

Without loss of generality, we assume that the basic unit of the money on the checks is 1 dollar. The problem in (c) is: given a set of checks $\{C_1, C_2 ..., C_n\}$, decide whether they can be divided evenly.

Then we construct the set of checks in problem (d) as $\{C_1 \bullet 1000, C_2 \bullet 1000, ..., C_n \bullet 1000\}$. The original set of checks can be divided evenly if and only if these multiplied checks can be divided evenly by allowing a difference no larger than 100. The transformation is apparently polynomial time.

34-3 Graph Coloring

(a) Starting from any vertex, mark this vertex in one color, and then mark all its neighbors in the other color. Do a breadth-first search to mark all the edges. Then check all edges to see whether one edge connects two vertices in the same color. If not, 2-colorable. Otherwise, not 2-colorable.

(b) Decision problem: given an undirected graph G = (V, E), if it could be colored by *k* colors such that no adjacent vertices have the same color.

If we can solve graph-coloring problem in polynomial time, we just check if *k* is larger than the minimum number of colors needed to color the given graph. If larger, then answer to the decision problem is no. Otherwise, yes.

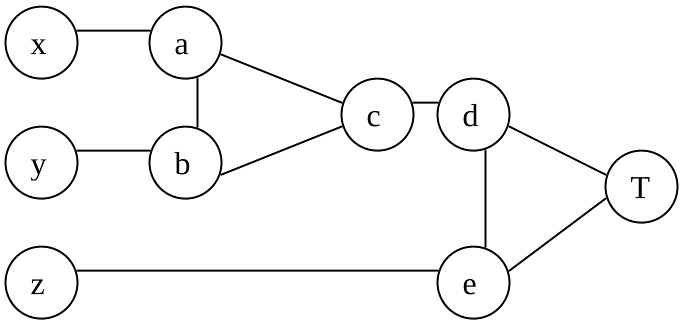
(c) First, given a coloring, we just check all edges to see if any edge connects two vertices in the same color. This is polynomial time verifiable.

Then we reduce 3-color problem to our decision problem.
Assume in the 3-color problem, we are given an undirected graph G = (V, E). We add *k-3* vertices, which are connected to each other, to G and connect all these new vertices to all existing vertices in G to form a new graph *G'*. Then, if G is 3-colorable, if and only if G' is k-colorable. The reduction is apparently polynomial.

Thus, we know if 3-color is NPC, k-colorable is also NPC.

(d) Without loss of generality, we look at variable *x*. Since *x* and $\overline{x}$ are connected, and they are both connected to RED. The same thing happens to TRUE and FALSE. Thus, in any 3-coloring, $c(x) \neq c(\overline{x})$, and either $c(x) = c(TRUE)$ or $c(x) = c(FALSE)$. Thus prove the statement.
We can color the graph containing only literal edges like this:
Give RED one color, give variable and TRUE the same color, variable negation and FALSE the same color. Obviously, no literal edge link two vertices of same color. Thus the graph containing only literal edges is always 3-colorable.

(e) First prove if widget is 3-colorable, then at least one of x, y and z is colored c(TRUE).
Use contradiction. Assume x, y and z are all colored c(FALSE). Then node c must be colored as c(FALSE). Then node d can't be c(FALSE) and also can't be c(TRUE). The same thing happens for node e. Thus d and e will have same color, because only tree colors are used. Contradiction found!



The other direction. Just look at each possibilities if at least one of x, y or z is colored c(TRUE). In all cases, the widget is 3-colorable.

(f) 3-COLOR is polynomial verifiable. Trivial.
Then reduce 3-CNF-SAT to 3-COLOR.
For a 3-CNF-SAT problem, create the graph containing only the literal edges and create one widget for each clause like in e.
Prove that if 3-CNF-SAT has a solution, if only the graph is 3-COLORable.

i) If 3-CNF-SAT has a truth assignment, then for each widget, at least one of x, y or z is TRUE. According to the proof (e), every widget is 3-COLORable. And in (d) we proved that the graph containing only the literal edges is always 3-COLORable. Thus the graph is 3-COLORable.
ii) If the graph is 3-COLORable, then in every widget, one of x, y, z has to be true. Just give the truth assignment if c(x) = c(TRUE), x = TRUE. In this assignment, 3-CNF-SAT is satisifiable.