1. (a) Let there be m clauses over n variables denoted by $x_1, \ldots, x_n$. We reduce the 2Sat problem to 3Sat by introducing a new variable x and create an instance of Sat consisting of $2m$ clauses over $n+1$ variables as follows.

   (i) For each clause $f(x_i) \vee g(x_j)$, where $f, g \in \{\text{id}, \neg\}$, we construct the two clauses $f(x_i) \vee g(x_j) \vee x$ and $f(x_i) \vee g(x_j) \vee \neg x$, taking $O(m)$ time.

   (ii) Return the result of 3Sat.

   This follows from the boolean identity $(a \vee b \vee c) \wedge (a \vee b \vee \neg c) = a \vee b$. If the RHS is satisfiable then the LHS is satisfiable for the same values of a and b and any value of c. If the LHS is satisfiable, we just delete c and the RHS is also satisfiable for the same values of a and b.

   Since this condition holds on every clause and kSat is True if all the clauses are satisfied, this reduction is correct.

   (b) Again, let there be m clauses over n variables denoted by $x_1, \ldots, x_n$. We create a graph with $2n$ vertices and $2m$ edges as follows.

   (i) Let our vertices be $x_i, \neg x_i$, representing whether or not the expression is True.

   (ii) For each clause $f(x_i) \vee g(x_j)$, where $f, g \in \{\text{id}, \neg\}$, we construct edges $\neg f(x_i) \to g(x_j)$ and $\neg g(x_j) \to f(x_i)$. Think of these as "implications". If one of the terms in the clause is False, then the other must be true.

   (iii) 2Sat is True if and only if there are no contradictions $x_i \implies \neg x_i$, which occurs in a cycle. We run a complete WFS to detect cycles, by restarting the WFS if the previous iteration did not traverse all the remaining vertices, returning True if no cycle is detected and False otherwise.

   Graph construction and WFS are both polynomial time, making the entire algorithm polynomial time.

   (c) We never assumed the existence of a polynomial time solution for 3Sat. In particular, our reduction only shows that there is a solution for 2Sat in the same approximate hardness class as 3Sat.

2. (a) Notice that SUBSETSUM on S with target T is TRUE if and only if SUBSETSUM on S with target $\sum_{s \in S} s - T$ is TRUE. Thus we assume WLOG that $T \geq \frac{1}{2} \sum_{s \in S} s$.

   We reduce SUBSETSUM on the set S with target T to PARTITION on the set $\bar{S}$ as follows.

   (i) We compute an additional integer $N = 2T - \sum_{s \in S} s$. Notice that by our assumption, $N \geq 0$.

   (ii) If $N = 0$, let $\bar{S} = S$. Else let $\bar{S} = S \cup \{N\}$.

   (iii) Return the result of PARTITION on $\bar{S}$.

   We claim the subset sum problem can be solved if and only if the partition problem can be solved.

   In the case $N = 0$, the subset sum problem is precisely the partition problem. If we get a partition, both sum to T. If we have one subset summing to T, the remaining elements also sum to T.

   In the remaining case, we now have a partition problem on a set summing to $2T$. If a subset sums to T the remaining elements union N also sum to T, solving the partition problem. If the partition problem is solvable, one of the subsets does not contain N, solving the subset sum problem.

   This is also clearly a polynomial time as we are only adjoining a single element to a set and making a polynomial time computation for N.

   (b) We compute the target $T = \frac{1}{2} \sum_{s \in S} s$. If this is not integral, then clearly PARTITION is false. We reduce the case where T is integral to SUBSETSUM as follows.

   (i) Return the result of SUBSETSUM on S with target T.

   We claim that the partition problem can be solved if and only if the subset problem can be solved. If the partition problem can be solved, then we get two subsets with sum T, either one of these solves the subset problem. If the subset problem can be solved, the remaining elements also have sum T, solving the partition problem.

   The computation of T is polynomial time, making the reduction polynomial time.

3. We reduce the standard HamiltonianPath problem on undirected graphs to PebbleClearing to show that PebbleClearing is NP-hard.

   Consider a graph $G = (V, E)$. A Hamiltonian path is a path that visits each vertex $v \in V$ precisely once. We can solve HamiltonianPath as follows.

   (i) Put one pebble on each vertex $v \in V$, taking $O(|V|)$ time.

   (ii) Adjoin an additional vertex ω, with two pebbles, to each original vertex, taking $O(|V|)$ time. Let this new graph be denoted $G' = (V', E')$.

   (iii) Return the result of PebbleClearing on G'.

   We claim G has a Hamiltonian path if and only if G' can be pebble cleared. If G has a Hamiltonian path $v_1 \to \cdots \to v_n$, we pebble clear via $\omega \to v_1, v_1 \to v_2, \ldots, v_{n-1} \to v_n, v_n \to v_{n-1}$, on v_{n-1}. Similarly, a pebble clearing of G' visits each vertex exactly once. At any given moment, there is precisely one vertex with two pebbles and a pebbling move must remove from this vertex. If we do not add the next pebble to a new vertex, no more pebbling moves can be made because every vertex has one pebble, so we must continue adding pebbles to unvisited vertices. A complete pebbling gives a Hamiltonian path because the next vertex you remove pebbles from is a neighbor of the previous. We omit ω, the guaranteed starting point, to get a Hamiltonian path on G.