The Theory of Computation
Bernard M.E. Moret
Addison-Wesley-Longman, 1998

Solution Manual
Chapter 7, Additional Problem 7.31

We consider the following cases: (i) fixed number of moves, unlimited number of pebbles (in which case re-pebbling obviously gains nothing on no re-pebbling); (ii) fixed number of pebbles, no re-pebbling; and (iii) fixed number of pebbles, re-pebbling.

1. With an unlimited number of pebbles, the problem always has answer yes (assuming no cycles in the graph and connectivity). With the number of moves limited by some constant, it is only necessary to check whether or not all of the designated vertex' ancestors can be pebbled in time. Thus we need only count the number of ancestors of the designated vertex and compare this number to the given bound. This is easily done in polynomial time. It appears rather harder to do it in logarithmic space, however, since we need to make sure that we are not counting the same ancestor twice. This may lead us to hypothesize that the problem in not in PolyL, which we might support by attempting to prove that the problem is P-complete.
2. Given a fixed number of pebbles, and with no re-pebbling allowed, the problem is clearly in NP(just list the moves and verify them; since there cannot be cycles, the total number of moves cannot exceed twice the number of vertices of the graph). The problem is in fact NP-complete (it is the register sufficiency problem of Garey & Johnson); the completeness reduction is quite tricky—see the paper "Complete register allocation problems" by R. Sethi, in *SIAM J. Comput.* 4 (1975), pp. 226–248.
3. When re-pebbling is allowed, the problem becomes PSpace-complete, a perhaps surprising fact since the game involves only one player. However, re-pebbling allows cycling, which allows an exponential number of moves. The state of the board can be represented a binary string of n (number of vertices) digits; the problem can be solved in non-deterministic polynomial space (and hence in deterministic polynomial space) by guessing successive states and verifying that

each move is legal and that the designated vertex gets pebbled after a number of moves not exceeding 2^n. We need only store two states (linear space) and one counter (also linear space). The completeness proof is quite tricky: see the article "The pebbling problem is complete in polynomial space" by J.R. Gilbert, T. Lengauer, and R.E. Tarjan, in *SIAM J. Comput.* **9** (1980), pp. 513–524.