Composite-Check(n):

composite ← false

Non-deterministically pick x and y ∈ {2, …, n-1}

If n = x * y **then** composite ← true

return composite

Visualizing the states of this algorithm Composite-Check, each state is a tuple <composite, x, y>. For this algorithm, if the correct output is **true**, then at least one of the states in which the algorithm is when it halts must correspond to a **true** output, and when the algorithm returns true, it's only because that n = x * y for x and y not being 1 or n itself, which means it is a composite number. And the algorithm only returns false if there does not exist x and y such that n = x * y, which makes it a prime number.

For the time-efficiency of this algorithm, Non-deterministically pick x and y has the same time efficiency with deterministically picking x and y, which is $\theta(1)$ in the best case. And in the worst case, it would take time $\theta(n)$, which is linear time. And reading the entire input n takes linear time in the size of s, where s is the binary form of n, which would be $\theta(|s|)$. Hence, the time efficiency of this algorithm is polynomial time.