Time complexity analysis

```
% Initial state
init_state(a). O(1)

% States O(1)
state(a).
state(b).
state(c).
state(d).
state(e).
state(f).
state(g).
state(h).
state(i).
state(j).
state(k).

% Accepting states O(1)
accept(b).
accept(c).
accept(d).
accept(e).
accept(g).
accept(h).
accept(j).

% Move O(1)
move(a, b, 0).
move(a, b, 2).
move(a, c, 1).

move(b, b, 0).
move(b, b, 2).
move(b, c, 1).

move(c, d, 0).
move(c, g, 1).
move(c, b, 2).
move(d, e, 1).
move(d, b, 0).
move(d, b, 2).

move(e, f, 1).
move(e, f, 2).
move(e, d, 0).

move(g, h, 0).
move(g, g, 1).
move(g, j, 2).

move(h, i, 1).
move(h, b, 0).
move(h, b, 2).

move(j, k, 2).
move(j, b, 0).
move(j, c, 1).

% Function to go through the list and get the final state
go_through_Digits([], State, State). O(1)

% Recursive function
O(n), where n is the length of the list.
go_through_Digits([Digit|RestofDigits], State1, State2) :-
(move(State1, NextState, Digit) -> O(1)
        NextState = NextState;
        NextState = State1),
    go_through_Digits(RestofDigits, NextState, State2).

% Verify if the list of digits leads to an accepting state
O(n), where n is the length of the list.
verify_number(Digits) :-
    init_state(StartState), O(1)
    go_through_Digits(Digits, StartState, FinalState), O(n), previous
    (accept(FinalState) -> O(1)
        writeln('yes');
        writeln('no')
    ).

% Change number to a list of digits
O(n) + O(n) -> O(2n) -> O(n)
digit_list(N, L) :-
    atom_chars(N, Chars), % O(n) Convert the atom to a list of characters
    maplist(atom_number, Chars, L). % O(n) Convert each character to its
numeric representation

O(2n) + O(n) -> O(3n) -> O(n)
check(Number) :-
    digit_list(Number, Digits), O(2n), previous
    verify_number(Digits). O(n), previous

Overall the time complexity is O(n)
```