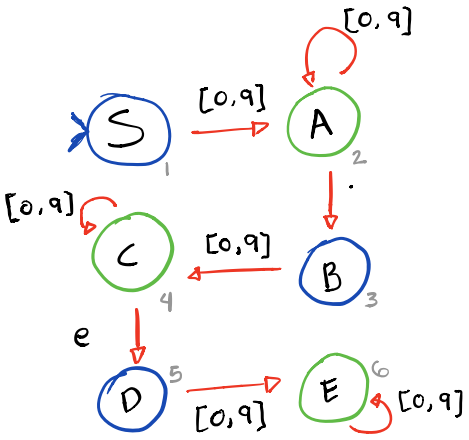
Kata 4

"Validate if a given string is numeric"

Assuming the sample input given, hexadecimal isn't part of the language that will be accepted. So, no HEX numbers allowed.

Requirements:

- ✓ Allow integers → 1, 2, 3, 4, 5, 7, 1000, 20000, ...etc
- ✓ Allow floating point → 0.1, 0.1734, $0.\overline{333}$ ... etc
- ✓ Allow scientific notation → 2e10, 2.37e-5, 2.37e 12.3 ...etc
- ✓ Allow + or - before the number → +5, -12e10, -15.36
- ✗ Reject letters at the beginning → e10, abc, x10, y12, ...etc
- ✗ Reject two+ decimal points (".") in string unless after an e. → 1.1.1, 12.3.12, 1.x.2 BUT 1.2e1.5
- ✗ Reject two+ e characters → 2e10e, e2e5 ... etc
- ✗ Decimal point only allowed if it's unique.



This problem can be solved with its associated regular automata. Which can be used to get a regex and could be easily evaluated.

Algorithm 1

```
int isNumber(char * s)
    replace whitespaces at the beginning.          1
    replace whitespaces at the end.                1
    for(int i = 0 ; i < s.length() ; i++)          n
        if ( s[i] is between 0 and 9)}
            if (getState <= 2) {                   All checks are O(1)
                setState(2)}
            else if (getState <5){setState(4);}
            else {setState(6);}}
        else if (s[i] = ".") {
            if (getState() = 2){ setState(3);}
            else return -1;  // It isn't a number.
        else if ( s[i] = "e") {
            if (getState() = 2 || getState()=4){
                setState(5);}    // Allows integers and floating point to use euler number notation.
        else
            return -1;  // Not a number.
     } //End of for cycle
     if (getState() = 2 || getState() = 4 || getState() = 6) {      1
        return 1; // It's a number
     else                                                          1
        return -1;   // Not a number.

int getState() { return state; }
void setState(int x) { state = x ; }
```

Complexity Analysis

We must check all the string, so memory and performance cost is $O(n)$. Every individual check (state check) is done in constant time. So final complexity is $O(n)$.