Solution 1 Solution 2

```swift
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

class Program {
    // O(n^3 + m) time | O(n + m) space
    func numbersInPi(_ pi: String, _ favoriteNumbers: [String]) -> Int {
        var numbersDictionary = [String: Bool]()

        for number in favoriteNumbers {
            numbersDictionary[number] = true
        }

        var cache = [Int: Int]()

        for i in stride(from: pi.count - 1, through: 0, by: -1) {
            getMinimumNumberOfSpaces(pi, numbersDictionary, &cache, i)
        }

        if cache[0] == Int(Int32.max) {
            return -1
        } else {
            return cache[0]!
        }
    }

    func getMinimumNumberOfSpaces(_ pi: String, _ numbersDictionary: [String: Bool], _ cache: inout [Int: Int], _ index: Int) -> Int {
        if index == pi.count {
            return -1
        }

        if let minimumNumberOfSpaces = cache[index] {
            return minimumNumberOfSpaces
        }

        var minimumNumberOfSpaces = Int(Int32.max)

        for i in index ..< pi.count {
            let startingIndex = pi.index(pi.startIndex, offsetBy: index)

            let endingIndex = pi.index(pi.startIndex, offsetBy: i + 1)

            let prefix = String(pi[startingIndex ..< endingIndex])

            if numbersDictionary.keys.contains(prefix) {
                let minimumNumberOfSpacesInSuffix = getMinimumNumberOfSpaces(pi, numbersDictionary, &cache, i + 1)

                minimumNumberOfSpaces = min(minimumNumberOfSpaces, minimumNumberOfSpacesInSuffix + 1)
            }
        }

        cache[index] = minimumNumberOfSpaces

        return minimumNumberOfSpaces
    }
}

```