Week 3 Exercises

Antonio Hanna

July, 21, 2024

Please complete all exercises below. You may use any library that we have covered in class UP TO THIS POINT.

For each function, show that it works, by using the provided data as a test and by feeding in some test data that you create to test your function

Add comments to your function to explain what each line is doing

1.) Write a function that takes in a string with a person's name in the form

"Sheets, Dave"

and returns a string of the form

"Dave Sheets"

Note:

-assume no middle initial ever -remove the comma -be sure there is white space between the first and last name

You will probably want to use stringr

```
library(stringr)
name_in <- "Sheets, Dave"

reorder_name <- function(name_in) {
  parts <- str_split(name_in, ", ")

  if (length(parts[[1]]) == 2) {
    last_name <- parts[[1]][1]
    first_name <- parts[[1]][2]

    formatted_name <- str_c(first_name, " ", last_name)

    return(formatted_name)
  }
  }


formatted_name <- reorder_name(name_in)
print(formatted_name)
```

```
## [1] "Dave Sheets"
```

2.) Write a function that takes in a string of values x, and returns a data frame with three columns, x, x^2 and the square root of x

```r
x= c(1,3,5,7,9,11,13)

powers_df<- function(x) {
df <- data.frame(
  reg_x = c(x),
  power_x = c(x^2),
  sqrt_x = c(sqrt(x))
)
  return(df)
}

df <- powers_df(x)
print(df)
```

```
##   reg_x power_x   sqrt_x
## 1     1       1 1.000000
## 2     3       9 1.732051
## 3     5      25 2.236068
## 4     7      49 2.645751
## 5     9      81 3.000000
## 6    11     121 3.316625
## 7    13     169 3.605551
```

3) Two Sum - Write a function named two_sum()

Given a vector of integers nums and an integer target, return indices of the two numbers such that they add up to target.

You may assume that each input would have exactly one solution, and you may not use the same element twice.

You can return the answer in any order.

Example 1:

Input: nums = [2,7,11,15], target = 9 Output: [0,1] Explanation: Because nums[0] + nums[1] == 9, we return [0, 1]. Example 2:

Input: nums = [3,2,4], target = 6 Output: [1,2] Example 3:

Input: nums = [3,3], target = 6 Output: [0,1]

Constraints:

2 <= nums.length <= 104 –109 <= nums[i] <= 109 –109 <= target <= 109 Only one valid answer exists.

Note: For the first problem I want you to use a brute force approach (loop inside a loop)

The brute force approach is simple. Loop through each element x and find if there is another value that equals to target – x

Use the function seq_along to iterate

```r
two_sum <- function(nums_vector,target) {
n <- (nums_vector)
  result <- list()
  for (i in seq_along(nums_vector)) {
  for (j  in seq_along(nums_vector)) {
    if(nums_vector[i] + nums_vector[j] == target) {
       result <- append(result,
                              list(c(i, j)))
```

```
  }
  }
  }
     return(result)
}


# Test code
nums_vector <- c(5,7,12,34,6,10,8,9)
target <- 13

z=two_sum(nums_vector,target)
print(z)
```

```
## [[1]]
## [1] 1 7
##
## [[2]]
## [1] 2 5
##
## [[3]]
## [1] 5 2
##
## [[4]]
## [1] 7 1
```

```
#expected answers
#[1] 1 7
#[1] 2 5
#[1] 5 2
```

Okay tough problem coming here! Look carefully at Jeremiah's examples of using a hash to simplify a loop by using a hash.

4) Now write the same function using hash tables.

Loop the array once to make a hash map of the value to its index. Then loop again to find if the value of target-current value is in the map.

The keys of your hash table should be each of the numbers in the nums_vector minus the target.

A simple implementation uses two iterations. In the first iteration, we add each element's value as a key and its index (array position) as a value to the hash table. Then, in the second iteration, we check if each element's complement (target – nums_vector[i]) exists in the hash table. If it does exist, we return current element's index and its complement's index. Beware that the complement must not be nums_vector[i] itself!

```
require(hash)
```

```
## Loading required package: hash
```

```
## hash-2.2.6.3 provided by Decision Patterns
```

```
library(hash)


two_sum2 <- function(nums_vector, target) {
  hash_table <- new.env(hash = TRUE, parent = emptyenv())
```

```
  results <- list()

  for (i in seq_along(nums_vector)) {
    complement <- target - nums_vector[i]

    if (exists(as.character(complement), envir = hash_table)) {
      stored_indices <- get(as.character(complement), envir = hash_table)

      for (index in stored_indices) {
        results <- c(results, list(c(index, i)))
      }
    }

    if (exists(as.character(nums_vector[i]), envir = hash_table)) {
      hash_table[[as.character(nums_vector[i])]] <- c(hash_table[[as.character(nums_vector[i])]], i)
    } else {
      hash_table[[as.character(nums_vector[i])]] <- list(i)
    }
  }

  return(results)
}




# Test code
nums_vector <- c(5,7,12,34,6,10,8,9)
target <- 15

two_sum(nums_vector,target)
```

```
## [[1]]
## [1] 1 6
##
## [[2]]
## [1] 2 7
##
## [[3]]
## [1] 5 8
##
## [[4]]
## [1] 6 1
##
## [[5]]
## [1] 7 2
##
## [[6]]
## [1] 8 5
```

```
#expected answers
#[1] 1 6
#[1] 2 7
#[1] 5 8
```

```
#[1] 61
#[1] 7 2
#[1] 8 5
```