Week 3 Exercises

Sue Susman, MEd, BSN, RN

April 2, 2023

Table of Contents

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

Exercise 1

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 <= 10^4
−10^9 <= nums[i] <= 10^9
−10^9 <= target <= 10^9
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

```
two_sum <- function(nums, target) {
 result <- list()
 n <- length(nums)
 for (i in seq_along(nums)) {
  for (j in seq_along(nums)) {
   if (i != j && nums[i] + nums[j] == target) {
    result[[length(result)+1]] <- c(i, j)
   }
  }
 }
 return(result)
}
```

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

two_sum(nums_vector, target)
## [[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
```

Exercise 2

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 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!

```
two_sum_hash <- function(nums, target) {
 hash_table <- list()
 indices <- list()

 # First iteration: populate the hash table with each element's value and index
 for (i in seq_along(nums)) {
  hash_table[[as.character(nums[i])]] <- i
 }

 # Second iteration: check if each element's complement exists in the hash table
 for (i in seq_along(nums)) {
  complement <- as.character(target - nums[i])
  if (complement %in% names(hash_table) && hash_table[[complement]] != i) {
   indices[[length(indices) + 1]] <- c(i, hash_table[[complement]])
  }
```

```
 }

 return(indices)
}

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

two_sum_hash(nums_vector, target)

#Expected answers
#[1] 10 5
#[1] 8 7
#[1] 9 6
#[1] 5 10
#[1] 7 8
#[1] 6 9

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