Week 3 Exercises

Michael Durning

March 27, 2023

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

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 <= 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

```
two_sum <- function(nums_vector, target) {

  # Check if input values are within valid range
  if (any(nums_vector < -109) || any(nums_vector > 109)) {
    stop("Invalid element in input vector nums.")
  }

  # Check if values are within range for target
  if (target < -109 || target > 109) {
    stop("Invalid target sum.")
  }

  # Brute force approach with loop within a loop
  # to find the two values that add up to target
  nums <- length(nums_vector)
  for (i in 1:(nums - 1)) {
```

```
    for (j in (i + 1):nums) {
      if (nums_vector[i] + nums_vector[j] == target) {
        return(c(i, j))
      }
    }
  }

  # Return null if no nums add up to target
  return(NULL)
}

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

two_sum(nums_vector, target)
```

```
## [1] 1 7
```

```
# How can we get expected answers, if constraints only say one answer?
```

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_vector, target) {

  # Check if values are within range for nums_vector
  if (any(nums_vector < -109) || any(nums_vector > 109)) {
    stop("Invalid element in input vector nums.")
  }

  # Check if values are within range for target
  if (target < -109 || target > 109) {
    stop("Invalid target sum.")
  }

  # Create hash table
  hash_table <- new.env(hash = TRUE, parent = emptyenv())

  # Stores the amount of numbers in nums_vector into nums
  nums <- length(nums_vector)

  # Uses hash tabel to find the complement values
  # that add up to traget then return if found.
  for (i in 1:nums) {
    complement <- target - nums_vector[i]
    if (exists(as.character(complement), envir = hash_table)) {
      return(c(hash_table[[as.character(complement)]], i))
    }
    hash_table[[as.character(nums_vector[i])]] <- i
```

```
  }

  # Return null if no nums add up to target
  return(NULL)
}

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

two_sum_hash(nums_vector, target)
```

```
## [1] 1 6
```

```
# How can we get expected answers, if constraints only say one answer?
```