Week 3 Exercises

Daniel Jackson

July 20, 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: [1,2] Explanation: Because nums[1] + nums[2] == 9, we return [1, 2]. Example 2:

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

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

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){
  # Function adds each nums_vector integer with every other integer in nums_vector
  ## and finds where the sum equals target integer, and will return corresponding
  ### indices of said integers
  for (i in 1:length(nums_vector)){
    # Head of for-loop
     for (j in (i + 1):length(nums_vector)){
       # Head of nested for-loop. j in (i + 1) ensures j starts from index directly
       ## after i, preventing duplicate pairs of printed indices
       if (!is.na(nums_vector[i]) && !is.na(nums_vector[j]) &&
          (target - nums_vector[i]) == nums_vector[j]){
         # If target minus i index of nums_vector is equal to j index of
         ## nums_vector, we then print combination of i and j.
         ### !is.na() function used to say that if those integers are not NA, proceed.
         #### I wrote this code in to avoid this error: "Error in if ((target -
         ##### nums_vector[i]) == nums_vector[j]) { : missing value where TRUE/FALSE needed"
         print(c(i,j))
        }
      }
   }
}

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

# My code returned:
two_sum(nums_vector,target)
```

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

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

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!

```r
library(hash)
```

```
## hash-2.2.6.2 provided by Decision Patterns
```

```r
two_sum = function(nums_vector,target){
h = hash()
# Create empty hash table h

  for (i in 1:length(nums_vector)){
    h[nums_vector[i]] = i
    # This code adds each element's value in nums_vector as a key and its index
    ## value i as a value in hash table h.
    complement = (target - nums_vector[i])
    # Create variable complement which is difference between target and
    ## value of i index in nums_vector.
    if (exists(as.character(complement), envir = h)){
      complement_index = get(as.character(complement), envir = h)
      print(c(complement_index, i))
      # Check to see if complement exists in hash map using exist() function. Used
      ## as.character() function to convert complement variable into character to
      ### see if that character exists in hash map h. Used get() function which will
      #### return the associated value with the compliment.
    }
  }
}

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

# My code returned:
two_sum(nums_vector,target)
```

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