Week 3 Exercises

Rebecca Hawthorne

July 13, 2023

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

```
constraint_checker <- function(nums_vector, target){
  # function to check that the vector and target are numeric and meet constraints

  if(is.numeric(nums_vector)&is.numeric(target)      #Check that all values are numeric
     &(2<= length(nums_vector)&(length(nums_vector)<= 104)   #check that values meet constraints
     & ( all(-109 <= nums_vector)) & (all(nums_vector<= 109))
     & (-109 <= target) & (target <= 109))){
      return (TRUE)
  }
  else {return (FALSE)}
}
```

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_shorter <- function(nums_vector,target){
  # a function to find the indices of two values in a vector that sum to target. This version of the fu
  #each possible pair of indices once.

  loop_work <- 0
```

```
  if(constraint_checker(nums_vector, target)){
    for(i in 1:(length(nums_vector)-1)) {
      for(j in (i+1):length(nums_vector)) {
        loop_work <- loop_work + 1

        if ((nums_vector[i] + nums_vector[j]) == target){
          cat(i, j, '\n')
        }

      }
    }
  } else{
    stop("ERROR: All values in the vector and the target must be numeric and meet given constraints.")
  }

  cat("Total work done is",loop_work)
}




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

two_sum_shorter(nums_vector,target)
```

```
## 1 7
## 2 5
## Total work done is 28
```

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

```
two_sum <- function(nums_vector,target){
  # a function to find the indices of two values in a vector that sum to target. This version of the fu
  #each possible pair of indices once.

  loop_work_longer <- 0

  if(constraint_checker(nums_vector, target)){
    for(i in 1:(length(nums_vector))) {
      for(j in 1:length(nums_vector)) {
        loop_work_longer <- loop_work_longer + 1

        if (((nums_vector[i] + nums_vector[j]) == target)&(i != j)){
          cat(i, j, '\n')
        }

      }
    }
  } else{
```

```
    stop("ERROR: All values in the vector and the target must be numeric and meet given constraints.")
  }

  cat("Total work done is",loop_work_longer)
}




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

two_sum(nums_vector,target)
```

```
## 1 7
## 2 5
## 5 2
## 7 1
## Total work done is 64
```

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

The shorter version of the code saves over half the steps! So unless you needed the code to check each pair of indices written both in ascending and descending order, I can use the shorter code to save work.

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 <- function(nums_vector,target){

   if(constraint_checker(nums_vector, target)){

    library(hash)
    complement_hash <- hash((target - nums_vector), 1:length(nums_vector))


    #check if numbers in nums_vector are equal to numbers in complement_hash and return indices
     for(i in 1:(length(nums_vector))) {
       if(nums_vector[i] %in% names(complement_hash)&
          (nums_vector[i]!=(target-nums_vector[i]))) {
         cat(i, complement_hash[[(as.character(nums_vector[i]))]], '\n')
       }


     }
```

```
}
else{
    stop("ERROR: All values in the vector and the target must be numeric and meet given constraints.")}
}
# Test code
nums_vector <- c(5,7,12,34,6,10,8,9)
target <- 15

two_sum(nums_vector,target)
```

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

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

```
#expected answers
#[1] 10  5
#[1] 8 7
#[1] 9 6
#[1]  5 10
#[1] 7 8
#[1] 6 9
```