

# Week 3 Exercises

*Benjamin Michaels*

*July 19th, 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[0] + nums[1] == 9`, we return `[1, 2]`. Example 2:

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

```
two_sum <- function(nums_vector,target)
  #your code here
  for (i in 1:(length(nums_vector) - 1 )) {
    for (j in (i+ 1):length(nums_vector)) {
      if (nums_vector[i] + nums_vector[j] == target){
        print(paste(nums_vector[i], nums_vector[j]))
      }
    }
  }

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

two_sum(nums_vector,target)
```

```
## [1] "5 8"
```

```
## [1] "7 6"
```

```
#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!*

```
library(hash)

## hash-2.2.6.2 provided by Decision Patterns

x1 <- c()
two_sum <- function(nums_vector,target){
  #your code here
  hmap <- hash()

  for(i in 1:length(nums_vector)){
    hmap[nums_vector[i]] <- nums_vector[1:length(nums_vector)]
    if (target - nums_vector[i] == target){
    }
  }
  for (j in 1: length(hmap)) {
    current <- nums_vector[j]
    x1 <- c(x1, names(hmap)[j], current)
    return(x1)
  }
}

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

two_sum(nums_vector,target)

## [1] "10" "5"
```

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