**Indicies**

**Ishaq Sajid**

Concordia University

November 6, 2022

CSC-

**Self-Assessment**

**1**. It took me just under a week to complete the assignment.

**2**. I put my entire effort into coding the programs. In this assignment, I will indeed expect an A Grade (100 on 100).

**3**. Every coding solution is correct. As a result, I would anticipate an A grade.

**4**. Learning the fundamentals of C++ produced coding simple. The main issue I encountered was in running the code. As I complete all of the weekly assignments, I am becoming more precise in detecting the errors and executing the program . The overall experience was excellent and will be best.

class Solution {

public:

vector<int> twoSum(vector<int>& nums, int target) {

vector<int> ret;

int size = nums.size();

int i,j;

for(i=0;i<size;i++) {

for(j=i+1;j<size;j++) {

if(nums[i]+nums[j] == target) {

ret.push\_back(i);

ret.push\_back(j);

return ret;

}

}

}

return ret;

}

};

Self-Evaluation:

I did this problem with a brute force solution. I first looped through the array and then looped through again. If I found two values that added up to the target, I saved the indices into a vector and returned the vector.

Time Complexity: O(n^2)

Space Complexity: O(1)

**Possible Improvements:**  
I could have saved the values into a hash map so that I can loop through once and then check if the complement exists in the hashmap.

Time Complexity: O(n)

Space Complexity: O(n)

\*/

class Solution {

public:

vector<int> twoSum(vector<int>& nums, int target) {

vector<int> ret;

int size = nums.size();

int i,j;

for(i=0;i<size;i++) {

for(j=i+1;j<size;j++) {

if(nums[i]+nums[j] == target) {

ret.push\_back(i);

ret.push\_back(j);

return ret;

}

}

}

return ret;

}

};

The above solution is a brute force algorithm one which loops thru the array twice to identify a pair of value systems that add up to the target. This has an O(n2) time complexity. We can enhance this by storing the values in a hashmap and then checking to see if the conjunction occurs in the hashmap. This would have an O(n) time complexity and an O(n) space complexity (n).

We loop and through array in the first loop and store the value systems in a hashmap. The second loop checks to see if the complement occurs in the hashmap. If it does, the indices are returned. Otherwise, we'll keep looping. Since we only loop thru the array once, the time complexity is O(n). Because we store this same values in a hashmap, the space complexity is O(n).

We could also sort the array with a sorting algorithm then use two pointers to find this same pair of values which add up to the target. Due to the sorting algorithm, this would have time complexity of O(nlogn). Because we wouldn't require any extra space, the space complexity would've been O(1).

A binary search algorithm could also be used to find this same complement of the each value in the array. Due to the binary search algorithm, this would have time complexity of O(nlogn). Because we didn't require any extra space, the space complexity would've been O(1).