Python Review / Kahoot

ITI 1120/1520

Introduction To Computing

1. Duplicate Number

Given an array of integers (nums) containing (n+1) integers, where each integer is in the range between 1 and n (inclusive)

There is one duplicate number in the array. Find that number.

Example:

Input: [1, 3, 5, 4, 3, 8]
Output: 3

because, 3 appears more than once in the input array

Option 1:

1. Copy the array into a new array (arr)
2. Sort array arr
3. Iterate through arr and check if each element is in the same position as array nums
4. Return the first element that is not in the same index

Option 2:

1. Create a new array of size n called arr2 and populate it with 0s
2. Iterate over the nums and add 1 to the same index in arr2
3. Iterate over arr2 and find the index i with a number 2
4. Using the index from step 3, return the element in index i from nums

Option 3:

1. Sort the array
2. Compare each element in the array to the element before it
3. As soon as we find a similar element, we return it

Option 4:

1. Iterate through the array nums
2. Each iteration we check if the element has the same number as the index
3. Return the element with a different number compared to its index

2. Arithmetic Progression From Sequence

Return true if an array can be rearranged to form an arithmetic progression.

A sequence is an arithmetic progression if the difference between two consecutive elements is the same

(Example: 1, 4, 7, 10... where the difference between each is 3)

Option 1:

1. Sort the array

2. Find the difference *diff* between the first and second element

3. Traverse through the sorted array, and return false if the difference between any index and its next index is not equal to diff

Option 2:

1. Find the difference *diff* between the first and second element

2. Traverse through the sorted array, and return false if the difference between any index and its next index is not equal to diff

Option 3:

1. Sort the array

2. Find the difference *diff* between the first and second element

3. Traverse through the sorted array, and return false if the difference between any index and its next index is not equal to 1

Option 4:

1. Sort the array

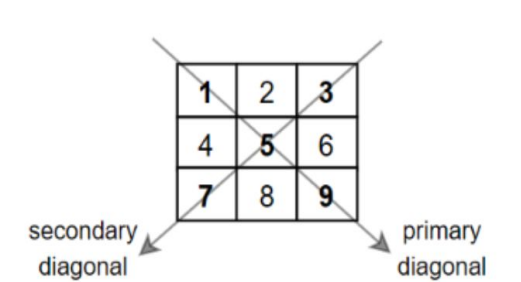
2. Take the modulus *mod* of the first element divided by the second element

3. Traverse through the sorted array, and return false if the mod of any index divided by its next index is not equal to *mod*

3. Matrix Diagonal Sum

Given a square matrix “matrix”, return the sum of the matrix diagonals.

Example:




```
Input: mat = [[1,1,1,1],
              [1,1,1,1],
              [1,1,1,1],
              [1,1,1,1]]

Output: 8
```

Option 1:

```
def diagonal_sum(matrix):
  n = len(matrix)
  total = 0
  for i in range(n):
    for j in range (n):
        If i==j:
            total+=matrix[i][j]

  If n % 2 == 1:
      total -= matrix[n//2][n//2]
  return total
```

Option 2:

```
def diagonal_sum(matrix):
  n = len(matrix)
  total = 0
  for i in range(n):
    for j in range (n):
        if i==j:
            total+=matrix[i][j]

  If n % 2 == 0:
      total -= matrix[n//2][n//2]
  return total
```

Option 3:

```
def diagonal_sum(matrix):
  n = len(matrix)
  total = 0
  for i in range(n):
    total += matrix[i][i]
    total += matrix[i][n-i-1]

  if n % 2 == 1:
    total -= matrix[n//2][n//2]
  return total
```

Option 4:

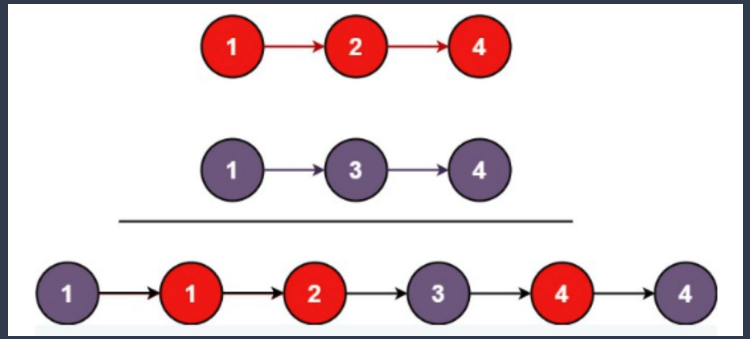
```
def diagonal_sum(matrix):
  n = len(matrix)
  total = 0
  for i in range(n):
    total += matrix[i][i]
    total += matrix[i][n-i-1]

  if n % 2 == 0:
    total -= matrix[n//2][n//2]
  return total
```

4. Merge Two Sorted Lists

Given two sorted lists, merge the lists to create one sorted list.

Example:



Option 1:

1. Create a new list of size = (length of list 1 + length of list 2)

2. Put the contents of list 2 into the new list

3. Sort using selection sort starting at index = length of list 1

Option 2:

1. Create a new list of size = (length of list 1 + length of list 2)

2. Put the contents of both lists into the new list

3. Sort using bubble sort starting at index=length of list 1

Option 3:

1. Create a new list of size = (length of list 1 + length of list 2)

2. Compare the first non-null element of each list and place the smaller one in new list, and the larger one immediately after that

3. Keep going until both lists are empty

4. Return the final array

Option 4:

1. Assign a pointer *i* for list 1 and pointer *j* for list 2

2. Compare the element at list1[i] with list2[j] and insert the lower element into a new list.

3. Increment i or j based on which index was inserted, and whether they are less than the list size

4. Return the final array once both i and j reach list size

5. Two Sum

Given an array of numbers *nums*, and an integer *target*, return two numbers from *nums* such that they add up to *target*.

Example:

Input:
nums: [3, 2, 4]
target: 6

Output:
[1,2]

because nums[1] + nums[2] = target

Option 1:

1. Create loop i that loops through each element starting at index 0

2. Create another loop j that starts at index 1

3. Check if nums[i]+nums[j]=target

4. If yes, return i and j, if not, keep going

Option 2:

1. Create loop i that loops through each element starting at index 0

2. Create another loop j that starts at index 1

3. Check if i==j. If yes, skip step 4.

4. Check if nums[i]+nums[j]=target

5. If yes, return i and j, if not, keep going

Option 3:

1. Create a loop i that loops through each element starting at index 0

2. Each time, have a variable *X* = target - nums[i]

3. Use binary search to find this *X*.

4. If it exists, return i and index of *X*. If not, keep looping.

Option 4:

1. Dictionary is initially empty. Put first element of array in it as a key with value the index.

2. Look at next element *i* and check if the result of target minus *i* is in the dictionary.

3. If match found, return the pair. If not, put it in as key *i* with value index.

4. Keep going until find solution, and return the value of key and index of element.

6. Length Of Last Word

Given a sentence string, with uppercase/lowercase letters and spaces, return the length of the last word of the sentence. (If no last word exists, return 0).

Example:

Input: “Python is fun”
Output: 3

because the word “fun” has three letters in it!

Option 1:

1. Create a length counter variable
2. Traverse the string to search for spaces
3. At each encountered space, reset counter to 0
4. At end of string, return counter

Option 2:

1. Calculate the length of the full sentence
2. Traverse the string to count the number of white spaces
3. Return the length value divided by the total number of spaces

Option 3:

1. Create a length counter variable
2. Traverse the string to search for spaces, incrementing length counter
3. At each encountered space, reset counter to 0
4. At end of string, return counter

Option 4:

1. Create a length counter variable
2. Traverse the string to search for spaces, incrementing length counter
4. At end of string, return counter

7. Single number

Given a non-empty list of integers *nums*, every element appears twice instead of one. Find that one integer.

Example:

Input: [2, 3, 5, 6, 3, 2, 6]
Output: 5

because every number other than 5 appears twice in that input array!

Option 1:

1. Iterate over all the elements in nums
2. If some number in nums is new to array, append it to list *seen*
3. If some number is already in the array, remove it from *seen*
4. Once the iteration is complete, return *seen[0]*

Option 2:

1. Iterate over all the elements in nums
2. If some number in nums is new to array, remove it from list *seen*
3. If some number is already in the array, append it to list *seen*
4. Once the iteration is complete, return *seen[0]*

Option 3:

1. Iterate over all the elements in nums
2. Check if one number is equal to the next
3. Return first number that is different from its next number

Option 4:

1. Sort the list
2. Iterate over list and check if one number is equal to the next
3. Return first number that is different from its next number and its previous number

Bonus question!!

What was the Big O of Two Sum from Question 5?

Code:

```
def two_sum_efficient(nums, target):

    pair = {}

    for i in range(len(nums)):
        if target - nums[i] in pair:
            return [pair[target - nums[i]], i]
        else:
            pair[nums[i]] = i

    return None
```

O(n^2)	O(n)
O(log n)	O(n^n)

```
import sys
sys.exit(0)
```