Algorithms – Interview Questions – Part 1

ddas.tech/algorithms-interview-questions-part-1/

October 2, 2022

Created By: Debasis Das (Oct 2022)

Table of Contents

Recursive Dictionary Search in Python

We have a dictionary and the value of each key of the dictionary can be a integer, a dictionary or a list of dictionary. Given a search key the requirement is to find the corresponding value by traversing through the entire dictionary. If a key is not found return None.

```
data_dictionary = {
    "A":1,
    "B":2,
    "C":{
        "D":4,
        "E":5
    },
    "F":[
        {"G":6,"H":7},
        {"I":8,"J":9}
    ]
}

def recursive_dict_search(search_dict,search_field):
    if type(search_dict) is dict:
        if search_field in search_dict:
            return search_dict[search_field]
        for key in search_dict:
            item = recursive_dict_search(search_dict[key],search_field)
            if item is not None:
                return item
    elif type(search_dict) is list:
        for element in search_dict:
            item = recursive_dict_search(element,search_field)
            if item is not None:
                return item
    return None

print("Value for A = ",recursive_dict_search(data_dictionary,"A"))
print("Value for B = ",recursive_dict_search(data_dictionary,"B"))
print("Value for C = ",recursive_dict_search(data_dictionary,"C"))
print("Value for D = ",recursive_dict_search(data_dictionary,"D"))
print("Value for E = ",recursive_dict_search(data_dictionary,"E"))
print("Value for F = ",recursive_dict_search(data_dictionary,"F"))
print("Value for G = ",recursive_dict_search(data_dictionary,"G"))
print("Value for H = ",recursive_dict_search(data_dictionary,"H"))
print("Value for I = ",recursive_dict_search(data_dictionary,"I"))
print("Value for J = ",recursive_dict_search(data_dictionary,"J"))

Output
Value for A =  1
Value for B =  2
Value for C =  {'D': 4, 'E': 5}
Value for D =  4
Value for E =  5
Value for F =  [{'G': 6, 'H': 7}, {'I': 8, 'J': 9}]
Value for G =  6
Value for H =  7
Value for I =  8
Value for J =  9
```

Prime Numbers

Prime Numbers are numbers which are only divisible by 1 and itself.

This problem is divided into 3 parts

- Given a number find out if the number is prime or not
- Find out the first N prime numbers
- Find Out all Prime Numbers Less than N

```
import math

# Check if a Number is a Prime Number or Not
def is_prime_number(n:int)-> bool:
    max_divisor = int(math.sqrt(n))
    is_Prime = True
    for i in range(2,max_divisor+1):
        rem = n%i
        if rem == 0:
            return False
    return is_Prime

print(f"4 is a prime number = {is_prime_number(4)}")
print(f"5 is a prime number = {is_prime_number(5)}")
print(f"7 is a prime number = {is_prime_number(7)}")
print(f"11 is a prime number = {is_prime_number(11)}")
print(f"100 is a prime number = {is_prime_number(100)}")

# Output
# 4 is a prime number = False
# 5 is a prime number = True
# 7 is a prime number = True
# 11 is a prime number = True
# 100 is a prime number = False

# Provide the List of First N Prime Numbers
def first_N_prime_numbers(n:int)->list:
    prime_numbers = [2]
    currentNumber = 2
    while len(prime_numbers) != n:
        currentNumber += 1
        if is_prime_number(currentNumber) == True:
            prime_numbers.append(currentNumber)

    return prime_numbers


print(first_N_prime_numbers(20))
#[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71]
```

```
# All Prime Numbers Less than N
def all_prime_numbers_less_than_N(n:int)->list:
    prime_numbers = []
    for i in range(2,n):
        if is_prime_number(i) == True:
            prime_numbers.append(i)

    return prime_numbers

print(all_prime_numbers_less_than_N(200))

#[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79,
83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167,
173, 179, 181, 191, 193, 197, 199]
```

FizzBuzz

- Write a Python program which iterates the integers from 1 to N.
- For multiples of three print “Fizz” instead of the number and
- For the multiples of five print “Buzz”.
- For numbers which are multiples of both three and five print “FizzBuzz”.

```
for i in range(1,31):
    if ((i%3 == 0) & (i%5 == 0)):
        print(f"{i} = FizzBuzz")
    elif i%3 == 0:
        print(f"{i} = Fizz")
    elif i%5 == 0:
        print(f"{i} = Buzz")
    else:
        print(f"{i} = {i}")
```

```
1 = 1
2 = 2
3 = Fizz
4 = 4
5 = Buzz
6 = Fizz
7 = 7
8 = 8
9 = Fizz
10 = Buzz
11 = 11
12 = Fizz
13 = 13
14 = 14
15 = FizzBuzz
16 = 16
17 = 17
18 = Fizz
19 = 19
20 = Buzz
21 = Fizz
22 = 22
23 = 23
24 = Fizz
25 = Buzz
26 = 26
27 = Fizz
28 = 28
29 = 29
30 = FizzBuzz
```

Fibonacci

- The Fibonacci numbers are the numbers in the following integer sequence.
  0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ……..
- In mathematical terms, the sequence Fn of Fibonacci numbers is defined by the recurrence relation
  Fn = Fn-1 + Fn-2

Fibonacci using a recursion approach

```
# Find the nth Fibonacci Number
def fibonacci(n):
    if n < 0:
        print("Incorrect input")
    elif ((n == 1) | (n==2)):
        return 1
    else:
        return fibonacci(n-1) + fibonacci(n-2)

print(fibonacci(10))
#55

fib_array1 = []
for i in range(1,10):
    val = fibonacci(i)
    fib_array1.append(val)


print(fib_array1)

#[1, 1, 2, 3, 5, 8, 13, 21, 34]
```

Fibonacci using a for loop

```
def fibonacci_series(n):
    fib_list = [1,1]
    for i in range(2,n):
        fib_list.append(fib_list[i-1]+fib_list[i-2])
    return fib_list
print(fibonacci_series(10))
#[1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

Factorial

Factorial of 5 is 5*4*3*2*1

Factorial of n in terms of recursion is Factorial(n) = n * Factorial(n-1)

Factorial Using a Recursion Approach

```
def factorial_recursion(n):
    if n > 1:
        return n* factorial_recursion(n-1)
    else:
        return 1

print(factorial_recursion(5))
#120
```

Factorial Using a For Loop

```
def factorial_loop(n):
    val = 1
    for i in range(n):
        val = val * (i+1)
    return val

print(factorial_loop(5))
#120
```

Armstrong Numbers

An Armstrong number of three digits is an integer such that the sum of the cubes of its digits is equal to the number itself.

For example, 371 is an Armstrong number since 3****3 + 7****3 + 1**3 = 371.

Write a program to find all Armstrong number in the range of 0 and 999

Approach 1

```
armstrong_numbers = []
for i in range (999):
    sum = 0
    for index,item in enumerate(str(i)):
        sum = sum + int(item) ** 3

    if sum == i:
        armstrong_numbers.append(i)

print(armstrong_numbers)
#[0, 1, 153, 370, 371, 407]
```

Approach 2

```
armstrong_numbers = []
for i in range(10):
    for j in range(10):
        for k in range(10):
            number = 100*i + 10*j + k
            armstrong_representation = i**3 + j**3 + k**3
            if number == armstrong_representation:
                armstrong_numbers.append(number)

print(armstrong_numbers)
#[0, 1, 153, 370, 371, 407]
```

Merge two already sorted Array in Python

```
def merge(data1, data2):
    mergedData = []
    i = 0
    j = 0
    while i < len(data1) and j < len(data2):
        if data1[i] < data2[j]:
            mergedData.append(data1[i])
            i += 1
        else:
            mergedData.append(data2[j])
            j += 1

#At this point find out what is left from list 1 and list 2 and add it to merged data
list
    while (i < len(data1)):
        mergedData.append(data1[i])
        i += 1
    while (j < len(data2)):
        mergedData.append(data2[j])
        j += 1

    return mergedData

print(merge([1,3,5,7,9,11],[2,4,6,8]))

#[1, 2, 3, 4, 5, 6, 7, 8, 9, 11]
```

Palindrome

A string is said to be palindrome if the reverse of the string is the same as string itself.
For example, “radar” is a palindrome
malayalam is a palindrome
abcd is not a palindrome

```
def isPalindrome(text)-> bool:
    is_palindrome = False
    if text == text[::-1]:
        is_palindrome = True
    return is_palindrome

print(isPalindrome("radar"))
print(isPalindrome("abcba"))
print(isPalindrome("malayalam"))
print(isPalindrome("abcd"))
```

In the below function we are comparing character at 0th index and last character , character at 1st index and last-1 index and so on

```
def isPalindrome_approach2(text)->bool:
    for i in range(0,int(len(text)/2)):
        if text[i] != text[len(text)-i-1]:
            return False
    return True
```

In the below function we are simply creating a new reverse string using python reversed function and comparing the same to the original string

```
def isPalindrome_approach3(text)->bool:
    reversed_text = ''.join(reversed(text))
    if text == reversed_text:
        return True
    else:
        return False
```

In the below function we are creating a new string in reverse by appending each character in the original string to the beginning of the newly formed string

```
def isPalindrome_approach4(text)->bool:
    temp_text = ""
    for x in text:
        temp_text = x + temp_text

    if temp_text == text:
        return True
    else:
        return False
```

In the below function we will use a recursive approach which checks the first and last character of a given string and then slices the string recursively and keeps checking the first and last character, In one occurrence where the first and last character does not match, it returns false

```
def isPalindrome_recursive(text):
    l = len(text)
    if l < 2:
        return True
    else:
        if text[0] == text[l-1]:
            return isPalindrome_recursive(text[1:l-1])
        else:
            return False
```

Word Counting Algorithm

```
text = """Lorem Ipsum is simply dummy text of the printing and typesetting industry.
Lorem Ipsum has been the industry's standard dummy text ever since the 1500s,
when an unknown printer took a galley of type and scrambled it to make a type
specimen book.
It has survived not only five centuries, but also the leap into electronic
typesetting,
remaining essentially unchanged. It was popularised in the 1960s with the release of
Letraset sheets containing Lorem Ipsum passages, and more recently with desktop
publishing
software like Aldus PageMaker including versions of Lorem Ipsum"""

word_count_dict = dict()
for word in text.split():
    if word in word_count_dict:
        word_count_dict[word] += 1
    else:
        word_count_dict[word] = 1

print(word_count_dict)
```

```
{'Lorem': 4, 'Ipsum': 4, 'is': 1, 'simply': 1, 'dummy': 2, 'text': 2, 'of': 4, 'the':
6, 'printing': 1, 'and': 3, 'typesetting': 1, 'industry.': 1, 'has': 2, 'been': 1,
"industry's": 1, 'standard': 1, 'ever': 1, 'since': 1, '1500s,': 1, 'when': 1, 'an':
1, 'unknown': 1, 'printer': 1, 'took': 1, 'a': 2, 'galley': 1, 'type': 2,
'scrambled': 1, 'it': 1, 'to': 1, 'make': 1, 'specimen': 1, 'book.': 1, 'It': 2,
'survived': 1, 'not': 1, 'only': 1, 'five': 1, 'centuries,': 1, 'but': 1, 'also': 1,
'leap': 1, 'into': 1, 'electronic': 1, 'typesetting,': 1, 'remaining': 1,
'essentially': 1, 'unchanged.': 1, 'was': 1, 'popularised': 1, 'in': 1, '1960s': 1,
'with': 2, 'release': 1, 'Letraset': 1, 'sheets': 1, 'containing': 1, 'passages,': 1,
'more': 1, 'recently': 1, 'desktop': 1, 'publishing': 1, 'software': 1, 'like': 1,
'Aldus': 1, 'PageMaker': 1, 'including': 1, 'versions': 1}
```