assignment

April 5, 2023

1 CP2410 Assignment 1 & 2

1.1 Q2

1.1.1 Fast Solution

[11]:
```
def get_primes(n):
  result = [True] * (n + 1)
  # 0 and 1 are not prime number
  result[0] = result[1] = False
  for i in range(2, int(n**(0.5))+1):
    if result[i]:
    # number i is a prime number, remove i**2+n*i
      for j in range(i**2, n+1, i):
        result[j] = False
  return [i for i in range(2, n+1) if result[i]]
```

[12]:
```
get_primes(20)
```

[12]:
```
[2, 3, 5, 7, 11, 13, 17, 19]
```

1.1.2 Slow Solution

1.2 Q1

[6]:
```
factors = {2:4, 3:2, 5:1, 7:1, 11:1, 13:1, 17:1, 19:1}
result = 1
for k, v in factors.items():
    result *= (k**v)
result
```

[6]:
```
232792560
```

1.2.1 Fast Solution

[24]:
```
def find_the_smallest_evenly_divisible(n: int):
  primes = get_primes(n)
  print(primes)
  factors = {}
```

```
  for num in primes:
    factors[num] = 1

  squart_root = int(n**0.5)
  i = 0
  while primes[i] <= squart_root:
    i += 1
  first_half_for_check = primes[:i]

  # find power for first half
  for num in first_half_for_check:
    power = 0
    while num**power < n:
      power += 1
    factors[num] = power - 1

  # for second half, th

  print(factors)
  result = 1
  for k, v in factors.items():
    result *= (k**v)
  return result
```

[25]:
```
find_the_smallest_evenly_divisible(10)
```

```
[2, 3, 5, 7]
{2: 3, 3: 2, 5: 1, 7: 1}
```

[25]: 2520

1.2.2 Slow Solution

1.3 Q3

1.3.1 Slow Solution

[41]:
```
def find_pythagorean_triplet(n: int):
    # 3, 4, 5 is the smallest pythagorean triplet
    for c in range(5, n+1):
        for b in range(4, c):
            for a in range(3, b):
                if a**2 + b**2 == c**2 and a + b + c == n:
                    print(f"a: {a}, b: {b}, c: {c}")
                    return
    print("No pythagorean triplet exist for number:", n)
```

[47]:
```
find_pythagorean_triplet(1231)
```

```
No pythagorean triplet exist for number: 1231
```

Time Complexity: O(n^3)

From the pythagorean theorem and property of right triangle, we know for all $a^{2+b}2$=c^2, a + b > c.

So if a+b+c = n, then c must < n/2.

Q3 faster version(v2), we reduce the length of checking by half for c

[44]:
```python
def find_pythagorean_triplet_v2(n: int):
    # 3, 4, 5 is the smallest pythagorean triplet
    for c in range(5, n//2 + 1):
        for b in range(4, c):
            for a in range(3, b):
                if a**2 + b**2 == c**2 and a + b + c == n:
                    print(f"a: {a}, b: {b}, c: {c}")
                    return
    print("No pythagorean triplet exist for number:", n)
```

[46]:
```python
find_pythagorean_triplet_v2(1231)
```

```
No pythagorean triplet exist for number: 1231
```

Time Complexity: O((n/2)^3), still O(n^3)