

LAPORAN TUGAS AKHIR
ALGORITMA DAN STRUKTUR DATA
LEETCODE



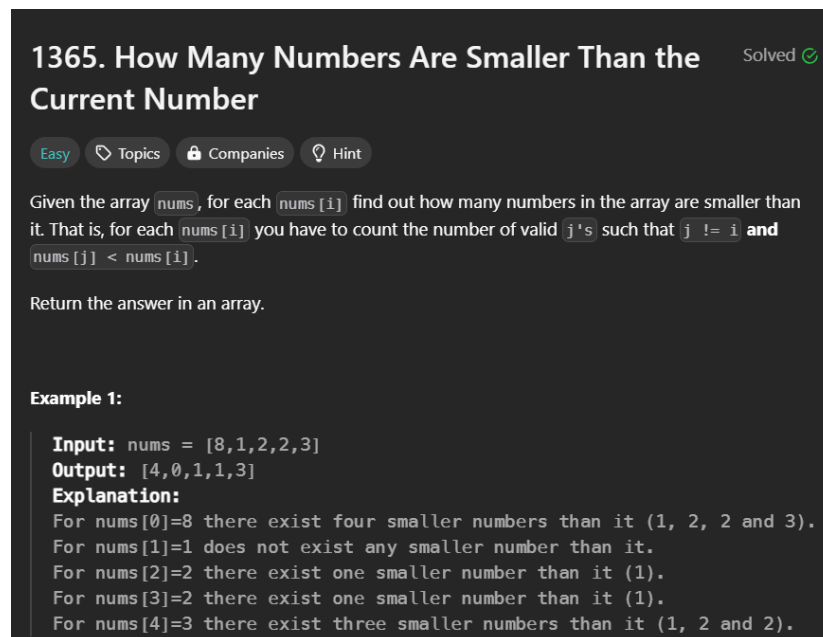
Disusun oleh:
Muhammad Aksal P
L200220269
C

TEKNIK INFORMATIKA
FAKULTAS KOMUNIKASI DAN INFORMATIKA
UNIVERSITAS MUHAMMADIYAH SURAKARTA
2023/2024

TUGAS

1. Judul masalah : *How Many Numbers Are Smaller Than the Current Number.*

Berikut merupakan *screenshot* dari masalah yang saya coba selesaikan :



gambar 1.1 screenshot masalah 1

Inti dari permasalahan ini yaitu : menghitung banyak angka yang lebih kecil dari angka sekarang.

Source Code

Berikut merupakan *screenshot* dari kode program yang digunakan sebagai solusi dari permasalahan ini :

```
1 class Solution(object):
2     def smallerNumbersThanCurrent(self, nums):
3         #hanya angka 0-100
4         arr = [0] * 101
5         for num in nums:
6             arr[num] += 1
7         acc = 0
8         for i in range(len(arr)):
9             prev = arr[i]
10            arr[i] = acc
11            acc += prev
12        return [arr[num] for num in nums]
13        print("L200220269")
```

gambar 1.2 screenshot kode masalah 1

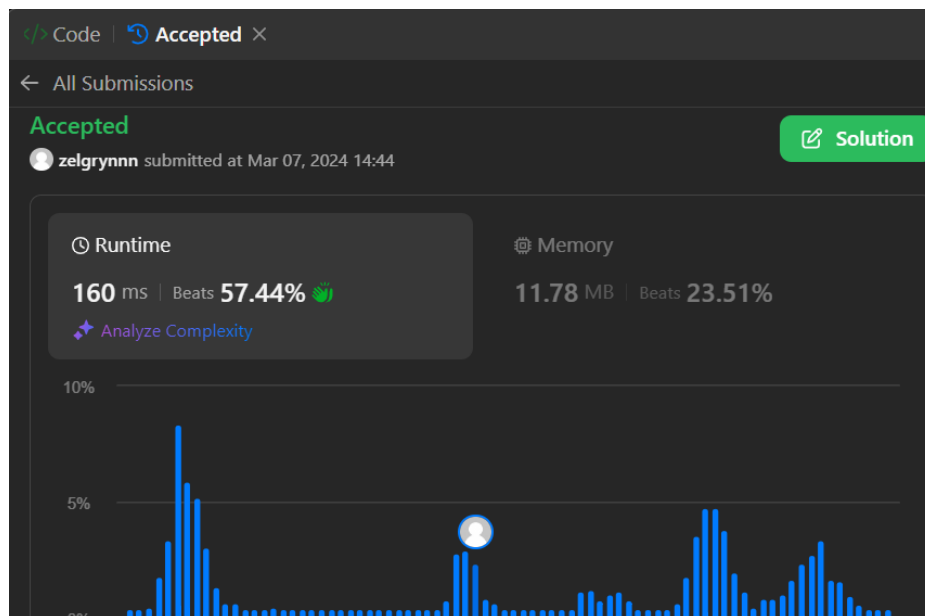
Penjelasan

Untuk menyelesaikan permasalahan ini, saya menggunakan logika solusi sebagai berikut:

- Membuat array dari 0-100
- Kemudian di loop untuk menghitung berapa kali setiap angka muncul dalam 'nums'. Setiap kali sebuah angka 'num' ditemukan, nilai pada index 'num' dalam 'arr' akan ditambahkan 1.
- Variabel 'acc' untuk menyimpan jumlah angka yang lebih kecil.
- Kemudian di loop yang berfungsi untuk akumulasi frekuensi. Nilai pada index 'i' pada 'arr' diubah menjadi jumlah semua angka yang lebih kecil daripada 'i'. Variabel 'acc' kemudian diperbarui dengan menambahkan 'prev' (nilai frekuensi asli pada index 'i' sebelum diubah).
- Mengembalikan nilai daftar baru yang berisi jumlah angka yang lebih kecil dari angka sekarang.

Bukti Accepted

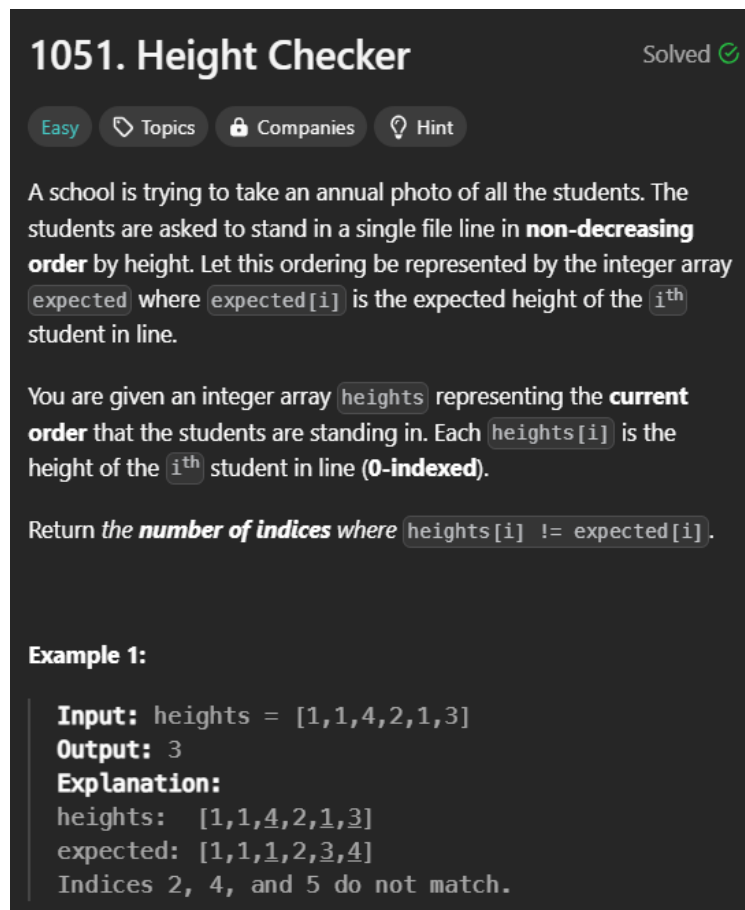
Berikut merupakan *screenshot* bukti bahwa solusi yang telah dibuat dapat berjalan dan diterima oleh sistem :



Gambar 1.3 screenshot accepted 1

2. Judul masalah : *Height Checker*.

Berikut merupakan *screenshot* dari masalah yang saya coba selesaikan :



gambar 1.1 screenshot masalah 2

Inti dari permasalahan ini yaitu : pengecekan tinggi.

Source Code

Berikut merupakan *screenshot* dari kode program yang digunakan sebagai solusi dari permasalahan ini :

```
1 class Solution(object):
2     def heightChecker(self, heights):
3         urutan_tinggi = sorted(heights)
4         return len([val for index, val in enumerate
5                     (urutan_tinggi) if urutan_tinggi[index] != heights
6                     [index]])
7         print("L200220269")
```

gambar 1.2 screenshot kode masalah 2

Penjelasan

Untuk menyelesaikan permasalahan ini, saya menggunakan logika solusi sebagai berikut:

- Variabel 'urutkan_tinggi' menyimpan nilai yang membuat salinan terurut dari daftar 'heights'.
- Melakukan iterasi melalui 'urutkan_tinggi' menggunakan **enumerate** untuk mendapatkan index dan nilai dari setiap elemen.
- Kemudian memeriksa apakah elemen pada index tertentu di 'urutkan_tinggi' berbeda dengan elemen pada index yang sama di 'heights'.
- Jika kondisinya benar, nilai elemen (val) dimasukkan ke dalam daftatr yang dihasilkan oleh list dan dikembalikan nilainya.

Bukti Accepted

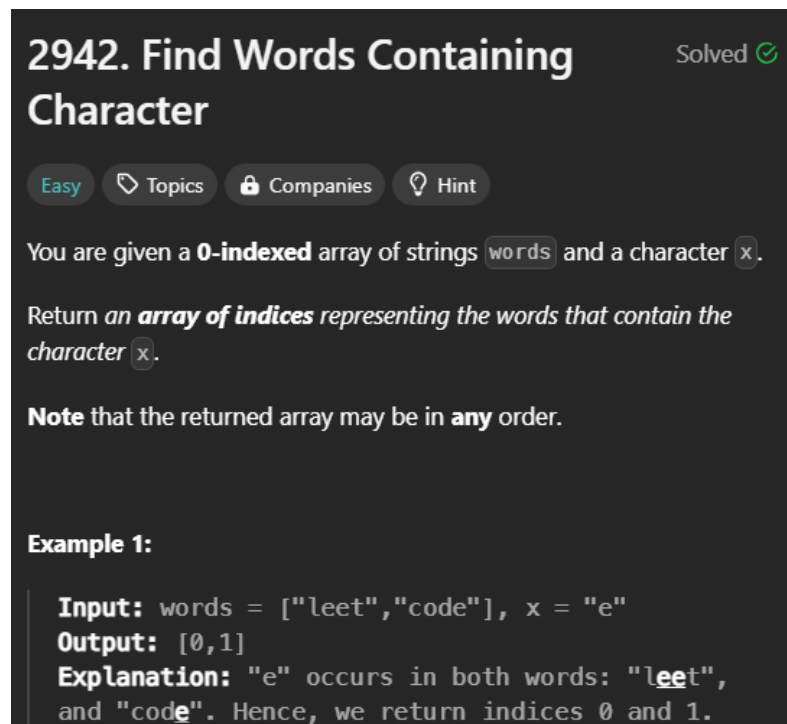
Berikut merupakan *screenshot* bukti bahwa solusi yang telah dibuat dapat berjalan dan diterima oleh sistem :



Gambar 1.3 screenshot accepted 2

3. Judul masalah : *Find Words Containing Character.*

Berikut merupakan *screenshot* dari masalah yang saya coba selesaikan :



gambar 1.1 screenshot masalah 3

Inti dari permasalahan ini yaitu : menemukan kata-kata yang mengandung karakter.

Source Code

Berikut merupakan *screenshot* dari kode program yang digunakan sebagai solusi dari permasalahan ini :

```
Python 3 Auto
1 class Solution(object):
2     def findWordsContaining(self, words, x):
3         output = []
4         index = 0
5         for i in words:
6             if x in i:
7                 output.append(index)
8                 index += 1
9         return output
10    print("L200220269")
```

gambar 1.2 screenshot kode masalah 3

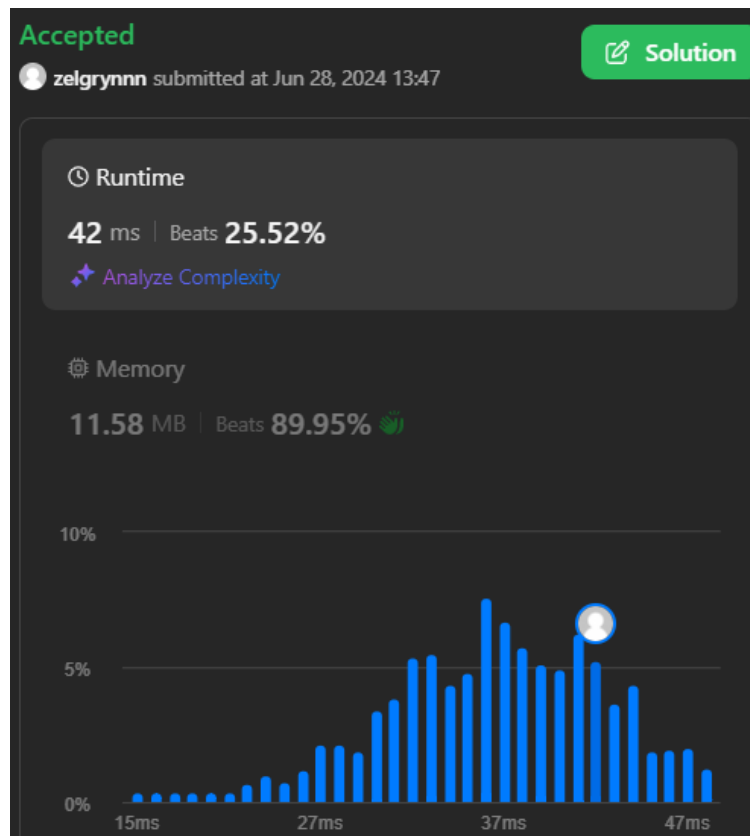
Penjelasan

Untuk menyelesaikan permasalahan ini, saya menggunakan logika solusi sebagai berikut:

- Variabel 'output' untuk menyimpan daftar kosong yang akan menampung index kata-kata yang mengandung substring 'x' dan variabel 'index' menyimpan angka 0 yang akan digunakan untuk melacak setiap posisi kata dalam 'words'.
- Kemudian diloop yang digunakan untuk iterasi setiap kata 'i' dalam 'words' dan memeriksa apakah substring 'x' ada dalam kata 'i'.
- Jika 'x' ditemukan dalam kata 'i', maka index kata tersebut ('index') ditambahkan ke dalam 'output' dan 'index' ditambahkan 1 setiap kali untuk melacak posisi kata berikutnya dalam 'words'.
- Mengembalikan 'output'.

Bukti *Accepted*

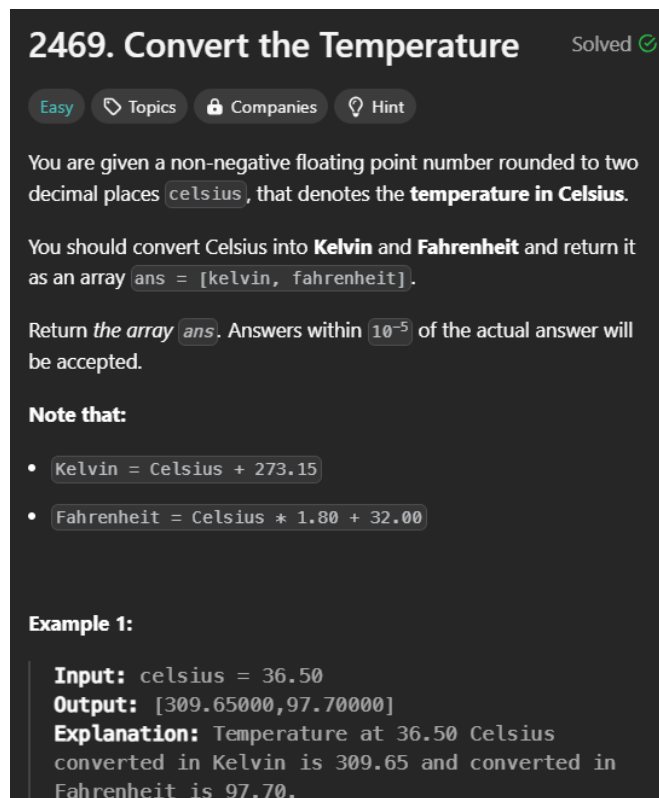
Berikut merupakan *screenshot* bukti bahwa solusi yang telah dibuat dapat berjalan dan diterima oleh sistem :



Gambar 1.3 screenshot accepted 3

4. Judul masalah : *Convert the Temperature.*

Berikut merupakan *screenshot* dari masalah yang saya coba selesaikan :



gambar 1.1 screenshot masalah 4

Inti dari permasalahan ini yaitu : mengubah temperatur dari celcius menjadi kelvin dan fahrenheit.

Source Code

Berikut merupakan *screenshot* dari kode program yang digunakan sebagai solusi dari permasalahan ini :

```
Python ▾ 🔒 Auto
1 class Solution(object):
2     def convertTemperature(self, celsius):
3         Kelvin = celsius + 273.15
4         Fahrenheit = celsius * 1.80 + 32.00
5         return (Kelvin, Fahrenheit)
6         print("L200220269")
```

gambar 1.2 screenshot kode masalah 4

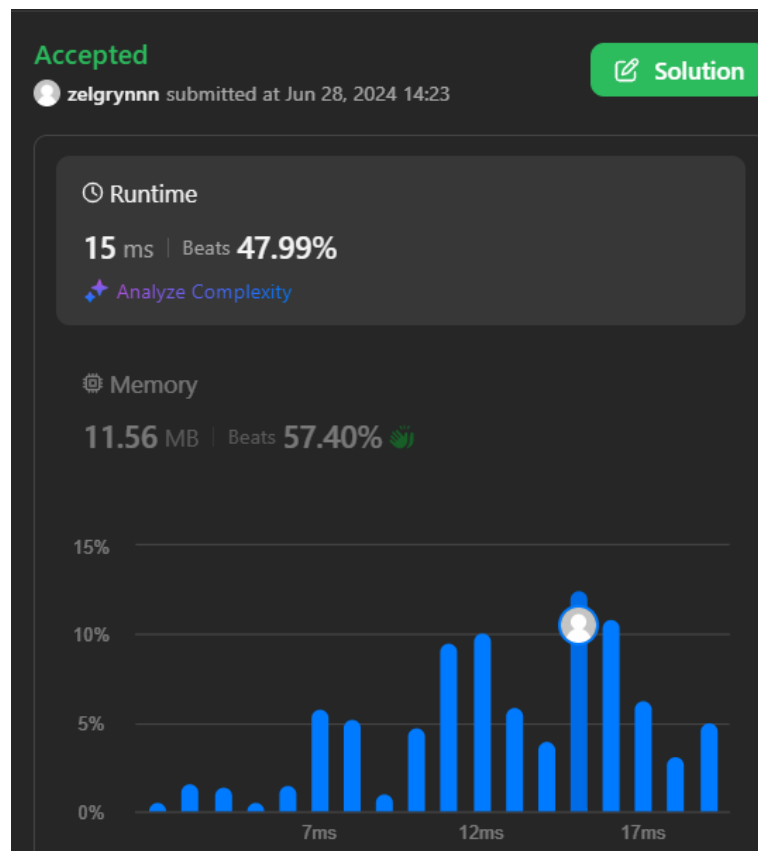
Penjelasan

Untuk menyelesaikan permasalahan ini, saya menggunakan logika solusi sebagai berikut:

- Membuat variabel Kevil yang berisi rumus untuk mengubah suhu dari celcius ke kelvin.
- Membuat variabel Fahrenheit yang berisi rumus untuk mengubah suhu dari celcius ke fahrenheit.
- Mengembalikan variabel Kelvin dan Fahrenheit.

Bukti *Accepted*

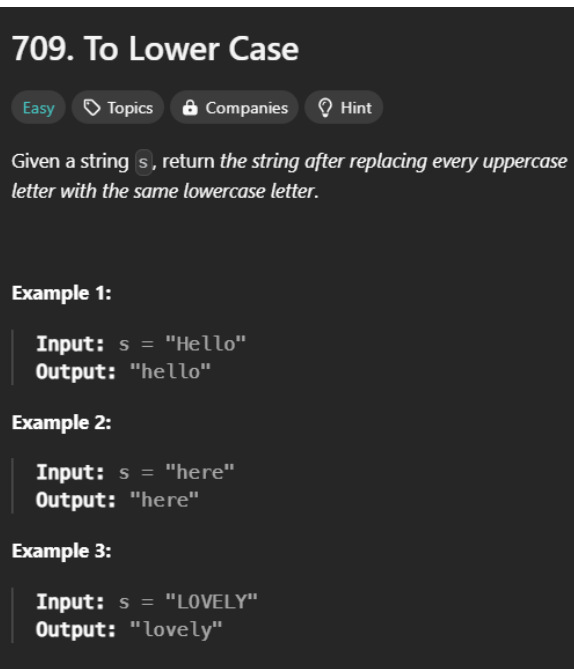
Berikut merupakan *screenshot* bukti bahwa solusi yang telah dibuat dapat berjalan dan diterima oleh sistem :



Gambar 1.3 screenshot accepted 4

5. Judul masalah : *To Lower Case*.

Berikut merupakan *screenshot* dari masalah yang saya coba selesaikan :



gambar 1.1 screenshot masalah 5

Inti dari permasalahan ini yaitu : mengubah kata menjadi huruf kecil.

Source Code

Berikut merupakan *screenshot* dari kode program yang digunakan sebagai solusi dari permasalahan ini :

```
Python 3 Auto
1 class Solution(object):
2     def toLowerCase(self, s):
3         return s.lower()
4         print("L200220269")
```

gambar 1.2 screenshot kode masalah 5

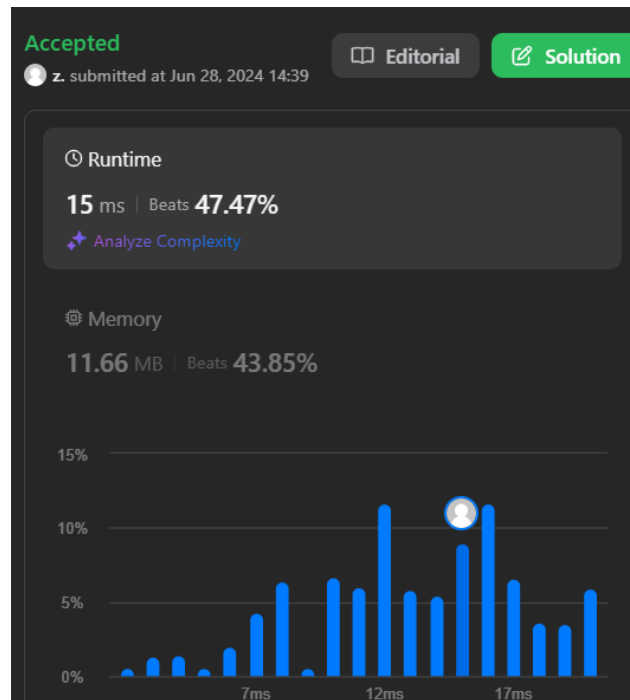
Penjelasan

Untuk menyelesaikan permasalahan ini, saya menggunakan logika solusi sebagai berikut:

- Mengembalikan nilai dari 's' yang berinputkan kata dan menjadikan semua hurufnya menjadi kecil.

Bukti Accepted

Berikut merupakan *screenshot* bukti bahwa solusi yang telah dibuat dapat berjalan dan diterima oleh sistem :



Gambar 1.3 screenshot accepted 5

6. Judul masalah : *Unique Email Addresses.*

Berikut merupakan *screenshot* dari masalah yang saya coba selesaikan :

929. Unique Email Addresses Solved ✓

Easy Topics Companies

Every **valid email** consists of a **local name** and a **domain name**, separated by the '@' sign. Besides lowercase letters, the email may contain one or more '.' or '+'.

- For example, in "alice@leetcode.com", "alice" is the **local name**, and "leetcode.com" is the **domain name**.

If you add periods '.' between some characters in the **local name** part of an email address, mail sent there will be forwarded to the same address without dots in the local name. Note that this rule **does not apply to domain names**.

- For example, "alice.z@leetcode.com" and "alicez@leetcode.com" forward to the same email address.

If you add a plus '+' in the **local name**, everything after the first plus sign **will be ignored**. This allows certain emails to be filtered. Note that this rule **does not apply to domain names**.

- For example, "m.y+name@email.com" will be forwarded to "my@email.com".

It is possible to use both of these rules at the same time.

Given an array of strings `emails` where we send one email to each `emails[i]`, return the number of different addresses that actually receive mails.

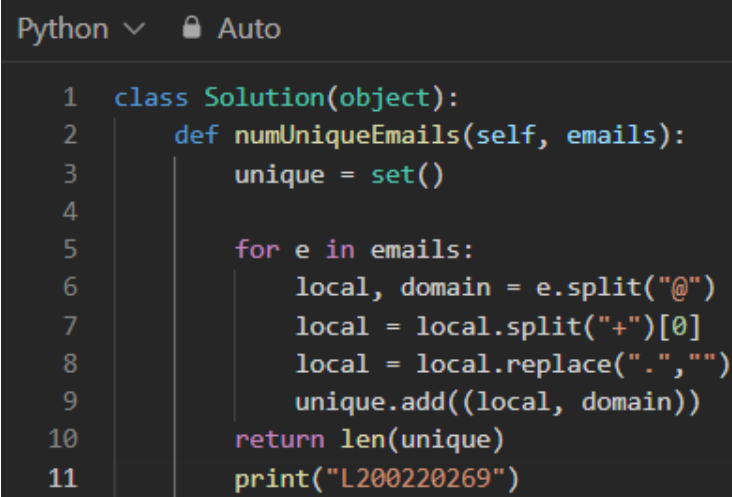
gambar 1.1 screenshot masalah 6

Inti dari permasalahan ini yaitu : setiap email terdiri dari nama lokal dan domain yang dipisahkan dengan tanda '@' dan selain huruf kecil, email dapat berisi satu atau dua

lebih tanda '.' atau '+', kemudian mengembalikan jumlah alamat email berbeda yang menerima email.

Source Code

Berikut merupakan *screenshot* dari kode program yang digunakan sebagai solusi dari permasalahan ini :



```
Python ▾ 🔒 Auto
1 class Solution(object):
2     def numUniqueEmails(self, emails):
3         unique = set()
4
5         for e in emails:
6             local, domain = e.split("@")
7             local = local.split("+")[0]
8             local = local.replace(".", "")
9             unique.add((local, domain))
10        return len(unique)
11        print("L200220269")
```

gambar 1.2 screenshot kode masalah 6

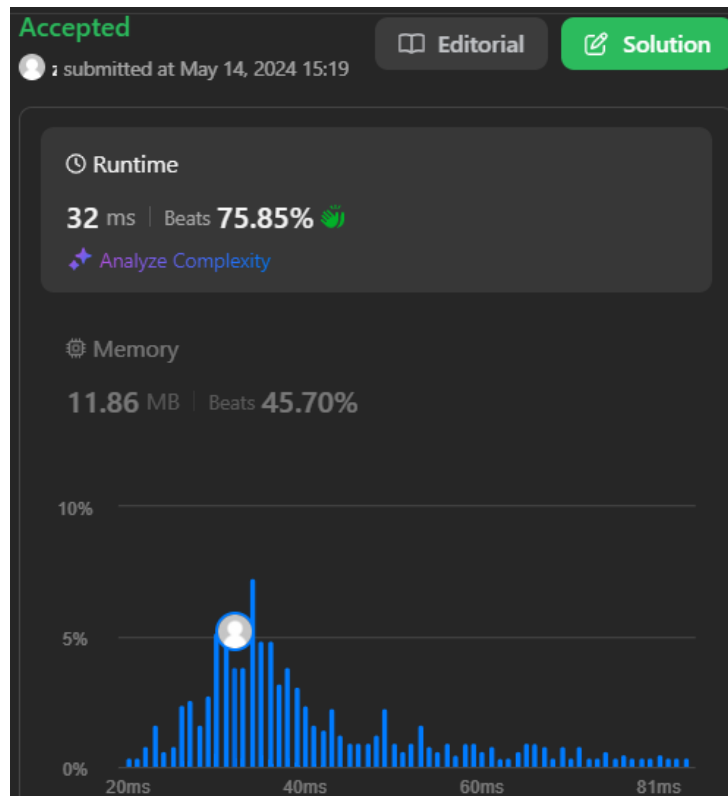
Penjelasan

Untuk menyelesaikan permasalahan ini, saya menggunakan logika solusi sebagai berikut:

- Membuat variabel 'unique' yang digunakan untuk menyimpan alamat email unik menggunakan set (digunakan untuk menghindari duplikasi).
- Melakukan looping pada 'emails', kemudian memisahkan alamat email menjadi dua bagian yaitu 'local' dan 'domain' berdasarkan '@'.
- Jika ada '+' dalam bagian 'local', maka hanya bagian sebelum '+' yang diambil. Bagian setelah '+' dan karakter '+' akan diabaikan.
- Menghapus semua '.' dalam bagian 'local'.
- Hasil dari 'local' dan 'domain' ditambahkan ke set 'unique' dan mengembalikan panjang dari set 'unique', yang merupakan jumlah alamat email unik.

Bukti Accepted

Berikut merupakan *screenshot* bukti bahwa solusi yang telah dibuat dapat berjalan dan diterima oleh sistem :



Gambar 1.3 screenshot accepted 6

7. Judul masalah : *Minimum Number of Moves to Seat Everyone.*

Berikut merupakan *screenshot* dari masalah yang saya coba selesaikan :

2037. Minimum Number of Moves to Seat Everyone Solved

Easy Topics Companies Hint

There are n **available** seats and n students **standing** in a room. You are given an array `seats` of length n , where `seats[i]` is the position of the i^{th} seat. You are also given the array `students` of length n , where `students[j]` is the position of the j^{th} student.

You may perform the following move any number of times:

- Increase or decrease the position of the i^{th} student by 1 (i.e., moving the i^{th} student from position x to $x + 1$ or $x - 1$)

Return the **minimum number of moves** required to move each student to a seat such that no two students are in the same seat.

Note that there may be **multiple** seats or students in the **same** position at the beginning.

Example 1:

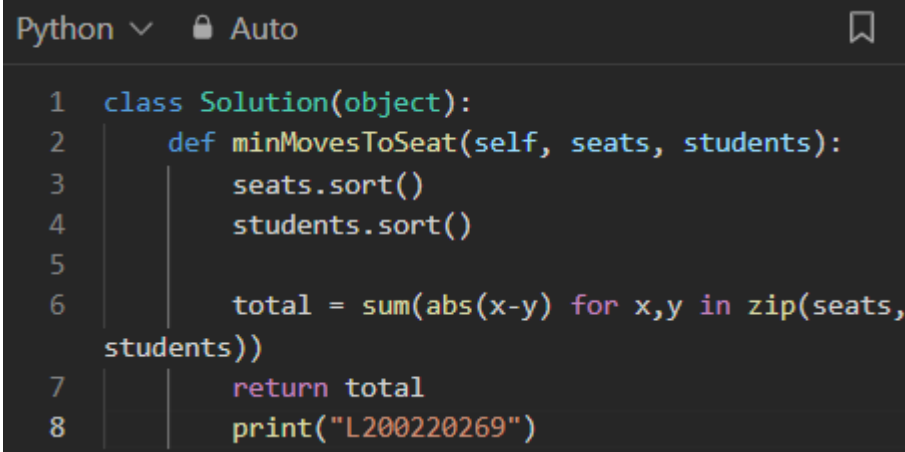
Input: `seats = [3,1,5]`, `students = [2,7,4]`
Output: 4

gambar 1.1 screenshot masalah 7

Inti dari permasalahan ini yaitu : menghitung perpindahan minimum untuk mendudukkan semua orang.

Source Code

Berikut merupakan *screenshot* dari kode program yang digunakan sebagai solusi dari permasalahan ini :



```
Python ▾ 🔒 Auto
1 class Solution(object):
2     def minMovesToSeat(self, seats, students):
3         seats.sort()
4         students.sort()
5
6         total = sum(abs(x-y) for x,y in zip(seats,
7         students))
8         return total
9         print("L200220269")
```

gambar 1.2 screenshot kode masalah 7

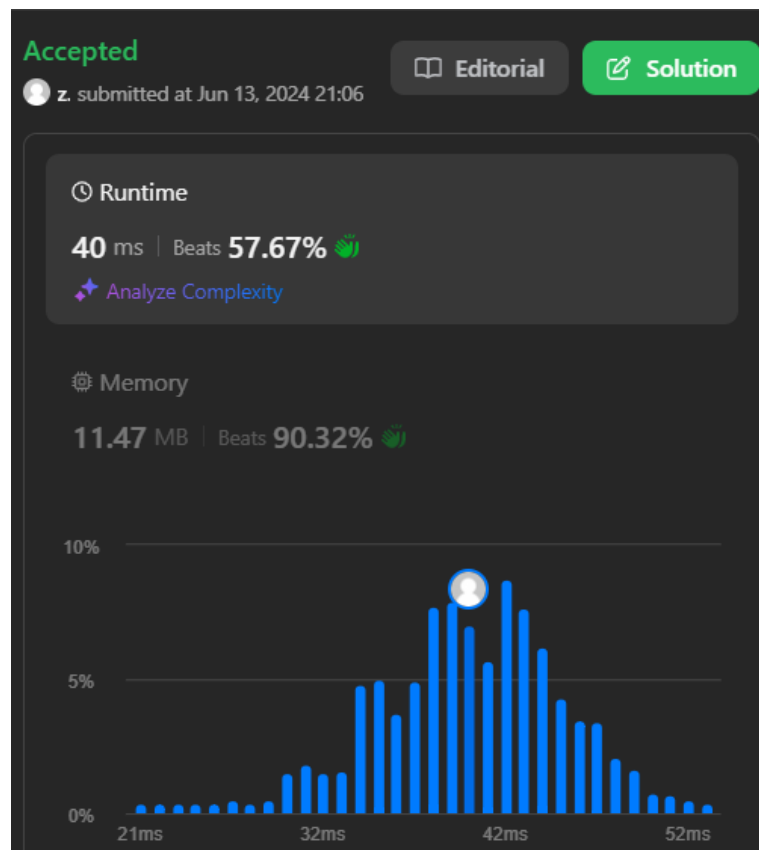
Penjelasan

Untuk menyelesaikan permasalahan ini, saya menggunakan logika solusi sebagai berikut:

- Mengurutkan 'seats' dan 'students' dari kecil ke besar.
- Membuat variabel 'total' yang menyimpan nilai dalam fungsi sum yang berisi nilai dari hasil selisih absolut antara posisi kursi dan posisi siswa yang dipasangkan dan hasil dari memasangkan elemen dari 'seats' dan 'students' satu per satu.
- Mengembalikan variabel 'total'.

Bukti Accepted

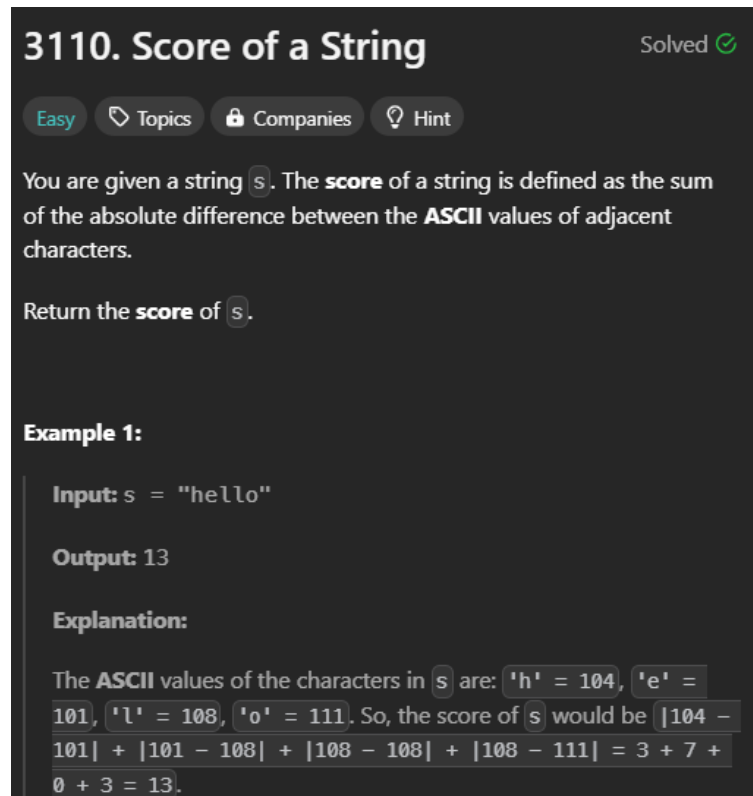
Berikut merupakan *screenshot* bukti bahwa solusi yang telah dibuat dapat berjalan dan diterima oleh sistem :



Gambar 1.3 screenshot accepted 7

8. Judul masalah : *Score of a String*.

Berikut merupakan *screenshot* dari masalah yang saya coba selesaikan :



gambar 1.1 screenshot masalah 8

Inti dari permasalahan ini yaitu : menghitung skor pada string yang diubah menjadi kode ASCII.

Source Code

Berikut merupakan *screenshot* dari kode program yang digunakan sebagai solusi dari permasalahan ini :

```
Python ▾ 🔒 Auto
1 class Solution(object):
2     def scoreOfString(self, s):
3         res = 0
4         for i in range(len(s)-1):
5             res += abs(ord(s[i]) - ord(s[i+1]))
6         return res
7         print(f"L200220269")
```

gambar 1.2 screenshot kode masalah 8

Penjelasan

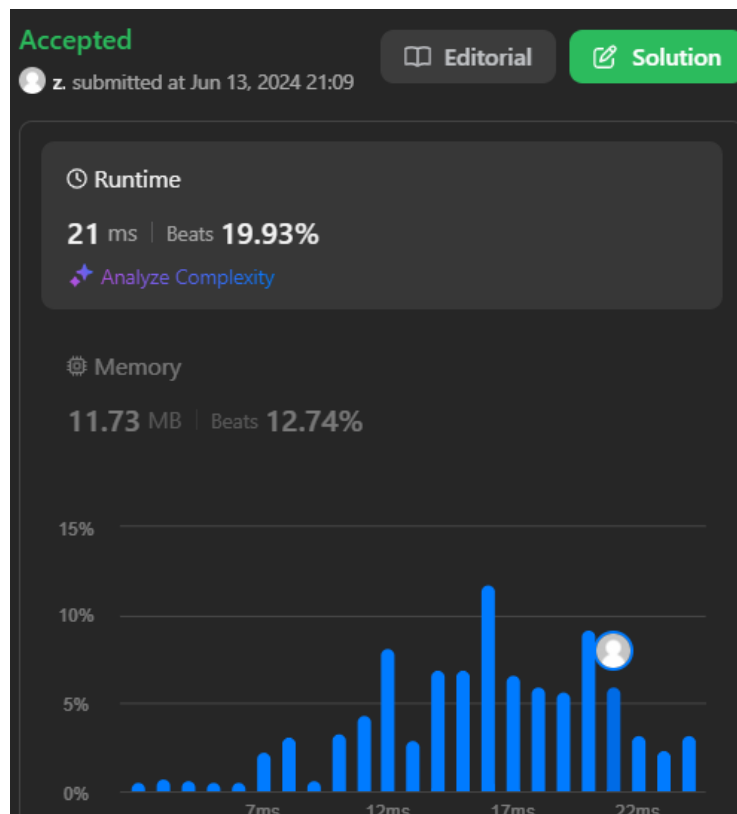
Untuk menyelesaikan permasalahan ini, saya menggunakan logika solusi sebagai berikut:

- Membuat variabel 'res' untuk menyimpan skor total.

- Melakukan perulangan melalui 's' dari index 0 hingga panjang 's' - 2. Setiap karakter pada index 'i' dan karakter berikutnya pada index 'i+1', kemudian menghitung nilai absolut dari nilai ASCII kedua karakter tersebut dan ditambahkan ke 'res'.
- Mengembalikan 'res'.

Bukti Accepted

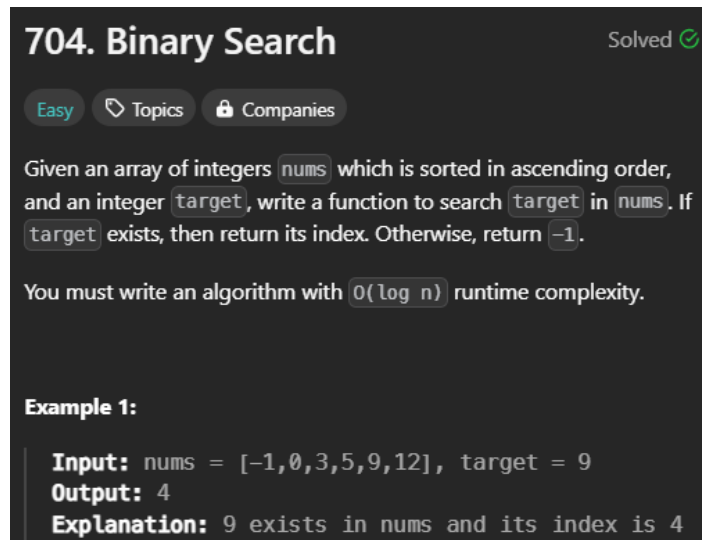
Berikut merupakan *screenshot* bukti bahwa solusi yang telah dibuat dapat berjalan dan diterima oleh sistem :



Gambar 1.3 screenshot accepted 8

9. Judul masalah : *Binary Search*.

Berikut merupakan *screenshot* dari masalah yang saya coba selesaikan :



gambar 1.1 screenshot masalah 9

Inti dari permasalahan ini yaitu : mencari angka target dan indexnya.

Source Code

Berikut merupakan *screenshot* dari kode program yang digunakan sebagai solusi dari permasalahan ini :

```
Python ▾ 🔒 Auto
1 class Solution(object):
2     def search(self, nums, target):
3         start = 0
4         end = len(nums)-1
5         while start <= end:
6             mid = (start + end)//2
7             num = nums[mid]
8             if target == num:
9                 return mid
10            elif target > num:
11                start = mid + 1
12            elif target < num:
13                end = mid - 1
14        return -1
15        print("L200220269")
```

gambar 1.2 screenshot kode masalah 9

Penjelasan

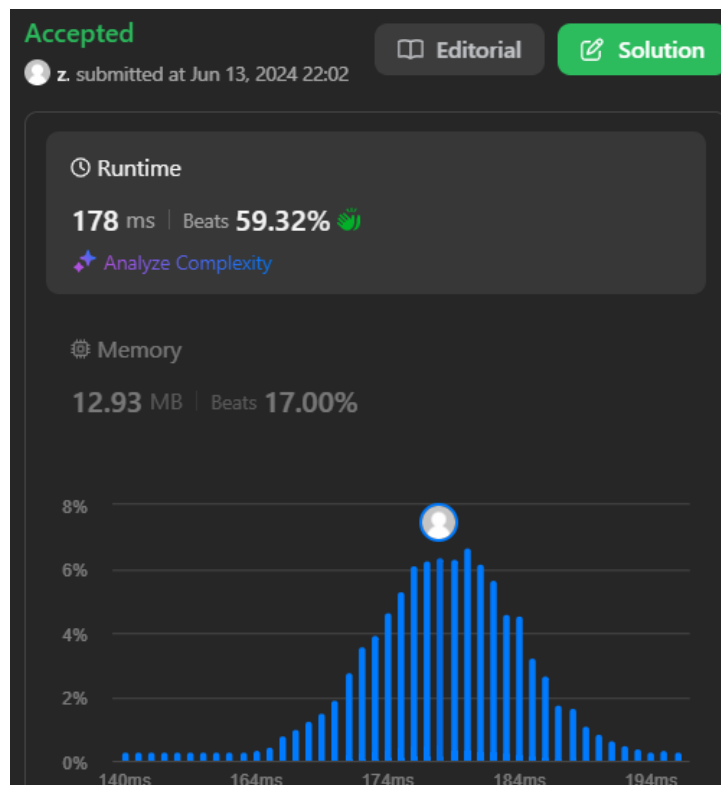
Untuk menyelesaikan permasalahan ini, saya menggunakan logika solusi sebagai berikut:

- Membuat variabel 'start' yang menyimpan nilai 0 dan variabel 'end' yang menyimpan nilai dari panjang 'nums' -1.

- Selama 'start' tidak melebihi 'end', hitung 'mid' sebagai index tengah. Kemudian bandingkan nilai 'target' dengan 'nums[mid]'.
- Jika 'target' sama dengan 'nums[mid]', kembalikan 'mid'.
- Jika 'target' lebih besar dari 'nums[mid]', perbarui 'start' menjadi 'mid' + 1 untuk mencari di bagian kanan.
- Jika 'target' lebih kecil dari 'nums[mid]', perbarui 'end' menjadi 'mid' - 1 untuk mencari di bagian kiri.
- Jika 'target' tidak ditemukan dalam daftar, kembalikan '-1'.

Bukti *Accepted*

Berikut merupakan *screenshot* bukti bahwa solusi yang telah dibuat dapat berjalan dan diterima oleh sistem :



Gambar 1.3 screenshot accepted 9

10. Judul masalah : *Maximum Depth of Binary Tree.*

Berikut merupakan *screenshot* dari masalah yang saya coba selesaikan :

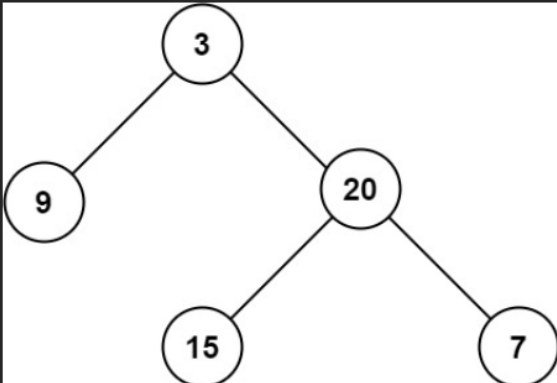
104. Maximum Depth of Binary Tree Solved

Easy Topics Companies

Given the `root` of a binary tree, return its *maximum depth*.

A binary tree's **maximum depth** is the number of nodes along the longest path from the root node down to the farthest leaf node.

Example 1:



```

graph TD
    3((3)) --- 9((9))
    3 --- 20((20))
    20 --- 15((15))
    20 --- 7((7))
  
```

Input: `root = [3,9,20,null,null,15,7]`
Output: 3

gambar 1.1 screenshot masalah 10

Inti dari permasalahan ini yaitu : mencari level kedalaman pada pohon biner.

Source Code

Berikut merupakan *screenshot* dari kode program yang digunakan sebagai solusi dari permasalahan ini :

```

Python Auto
1 # Definition for a binary tree node.
2 # class TreeNode(object):
3 #     def __init__(self, val=0, left=None, right=None):
4 #         self.val = val
5 #         self.left = left
6 #         self.right = right
7 class Solution(object):
8     def maxDepth(self, root):
9         """
10         :type root: TreeNode
11         :rtype: int
12         """
13         if not root:
14             return 0
15         l = self.maxDepth(root.left)
16         r = self.maxDepth(root.right)
17         return max(l, r) + 1
18         print("L200220269")
  
```

gambar 1.2 screenshot kode masalah 10

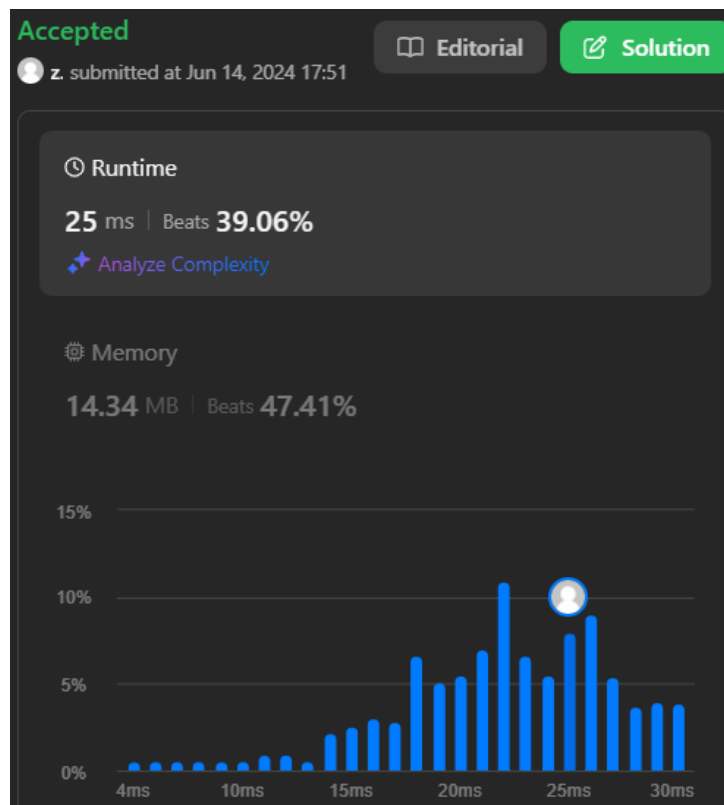
Penjelasan

Untuk menyelesaikan permasalahan ini, saya menggunakan logika solusi sebagai berikut:

- Jika 'root' adalah None (pohon kosong), maka kedalaman adalah 0.
- Menghitung maksimum dari subpohon kiri dan subpohon kanan dan menyimpannya dalam variabel 'l' dan 'r'.
- Mengembalikan nilai max dari 'l' dan 'r', ditambah 1 untuk memperhitungkan akar saat ini.

Bukti Accepted

Berikut merupakan *screenshot* bukti bahwa solusi yang telah dibuat dapat berjalan dan diterima oleh sistem :



Gambar 1.3 screenshot accepted 10

11. Judul masalah : *Destination City*.

Berikut merupakan *screenshot* dari masalah yang saya coba selesaikan :



gambar 1.1 screenshot masalah 11

Inti dari permasalahan ini yaitu : mencapai kota pada array yang diberikan dengan kembali ke kota tujuan, yaitu kota tanpa jalur keluar ke kota lain.

Source Code

Berikut merupakan *screenshot* dari kode program yang digunakan sebagai solusi dari permasalahan ini :

```
Python ▼ 🔒 Auto
1 class Solution(object):
2     def destCity(self, paths):
3         A, B = map(set, zip(*paths))
4         return (B-A).pop()
5         print("L200220269")
```

gambar 1.2 screenshot kode masalah 11

Penjelasan

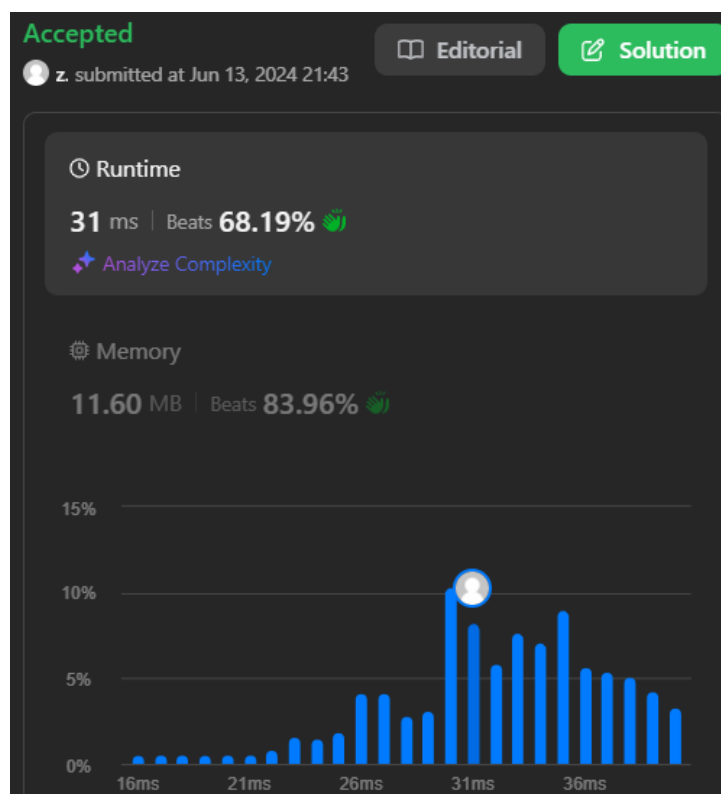
Untuk menyelesaikan permasalahan ini, saya menggunakan logika solusi sebagai berikut:

- 'zip(*paths))' akan mengurai daftar 'paths' ke dalam dua tuple terpisah: satu semua kota awal dan satu lagi untuk semua kota tujuan,

- `'map(set,zip(*paths))'` kemudian mengubah kedua tuple ini menjadi 2 set 'A' dan 'B'. Set 'A' berisi semua kota awal, dan set 'B' berisi semua kota tujuan.
- Kemudian menghitung selisih set, yaitu semua elemen dalam 'B' yang tidak dalam 'A'. Ini memberikan kita kota yang hanya muncul sebagai kota tujuan dan `'pop()'` mengambil satu elemen dari hasil set (karena kita hanya tahu hanya ada satu kota tujuan akhir, ini akan menjadi kota yang kita cari).

Bukti *Accepted*

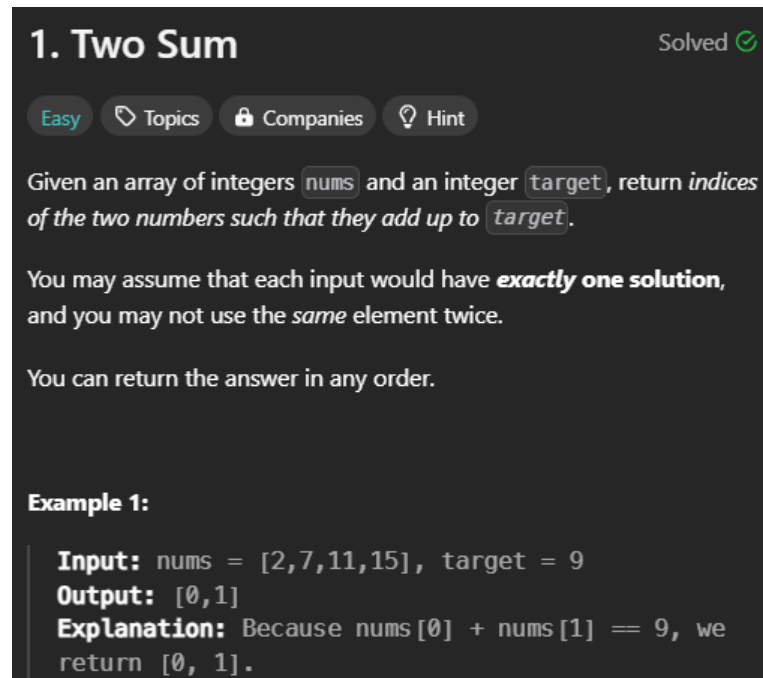
Berikut merupakan *screenshot* bukti bahwa solusi yang telah dibuat dapat berjalan dan diterima oleh sistem :



Gambar 1.3 screenshot accepted 11

12. Judul masalah : *Two Sum*.

Berikut merupakan *screenshot* dari masalah yang saya coba selesaikan :



gambar 1.1 screenshot masalah 12

Inti dari permasalahan ini yaitu : mencari target penjumlahan menggunakan index, jika target ditemukan akan mencetak index penjumlahan.

Source Code

Berikut merupakan *screenshot* dari kode program yang digunakan sebagai solusi dari permasalahan ini :

```
Python v Auto
1 class Solution(object):
2     def twoSum(self, nums, target):
3         kumpulan = {}
4         for i, num in enumerate(nums):
5             if target - num in kumpulan:
6                 return [kumpulan[target - num], i]
7             kumpulan[num] = i
8             print("L200220269")
9
```

gambar 1.2 screenshot kode masalah 12

Penjelasan

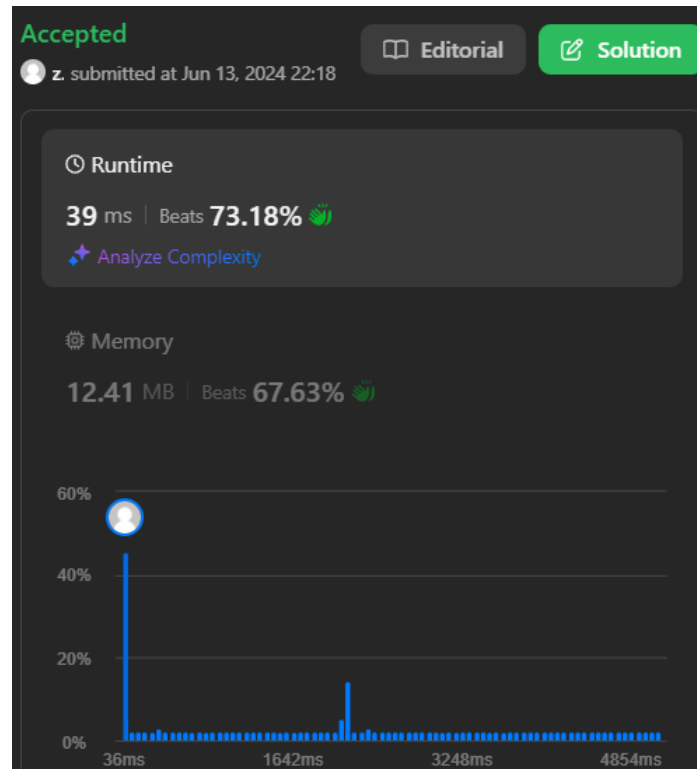
Untuk menyelesaikan permasalahan ini, saya menggunakan logika solusi sebagai berikut:

- Membuat variabel 'kumpulan' dalam bentuk *dictionary* untuk menyimpan angka dan index.

- Menggunakan 'enumerate' untuk mendapatkan index 'i' dan nilai 'num'.
- Jika 'target - num' ada di 'kumpulan', kembalikan index pasangan yang sesuai.
- Simpan 'num' dan index 'i' dalam 'kumpulan'.

Bukti *Accepted*

Berikut merupakan *screenshot* bukti bahwa solusi yang telah dibuat dapat berjalan dan diterima oleh sistem :



Gambar 1.3 screenshot accepted 12

13. Judul masalah : *Search in a Binary Search Tree.*

Berikut merupakan *screenshot* dari masalah yang saya coba selesaikan :

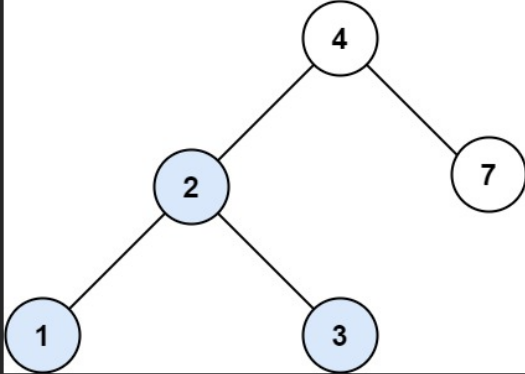
700. Search in a Binary Search Tree Solved

Easy Topics Companies

You are given the `root` of a binary search tree (BST) and an integer `val`.

Find the node in the BST that the node's value equals `val` and return the subtree rooted with that node. If such a node does not exist, return `null`.

Example 1:



Input: `root = [4,2,7,1,3]`, `val = 2`
Output: `[2,1,3]`

gambar 1.1 screenshot masalah 13

Inti dari permasalahan ini yaitu : mencari node yang sesuai dengan 'val' dan mengembalikan *subtree* dengan node 'val'.

Source Code

Berikut merupakan *screenshot* dari kode program yang digunakan sebagai solusi dari permasalahan ini :

```
Python Auto
1 # Definition for a binary tree node.
2 # class TreeNode(object):
3 #     def __init__(self, val=0, left=None, right=None):
4 #         self.val = val
5 #         self.left = left
6 #         self.right = right
7 class Solution(object):
8     def searchBST(self, root, val):
9         node = root
10        while node:
11            if node.val == val:
12                return node
13            elif val > node.val:
14                node = node.right
15            elif val < node.val:
16                node = node.left
17        return None
18        print("L200220269")
```

gambar 1.2 screenshot kode masalah 13

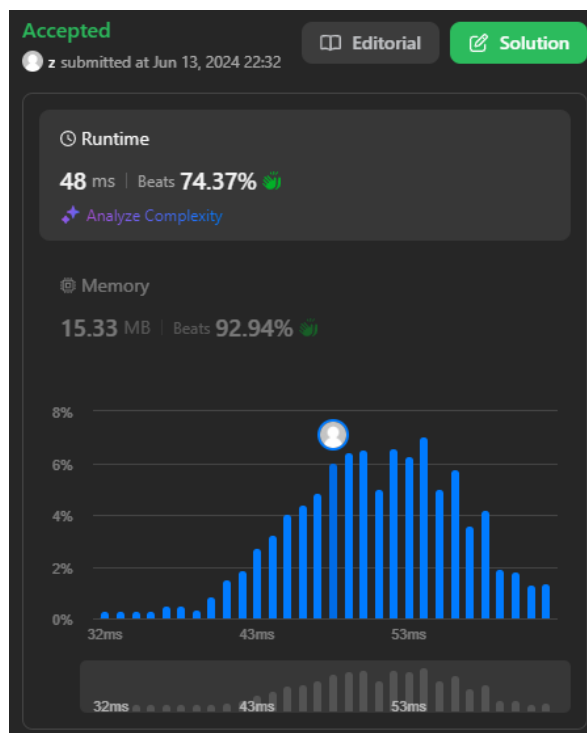
Penjelasan

Untuk menyelesaikan permasalahan ini, saya menggunakan logika solusi sebagai berikut:

- Membuat variabel 'node' dengan nilai 'root' untuk memulai pencarian dari akar pohon.
- Selama 'node' tidak 'None', periksa nilai 'node.val'.
- Jika 'node.val' sama dengan 'val', kembalikan 'node'.
- Jika 'val' lebih besar dari 'node.val', pindah ke subpohon kanan dengan mengatur 'node' menjadi 'node.right'.
- Jika 'val' lebih kecil dari 'node.val', pindah ke subpohon kiri dengan mengatur 'node' menjadi 'node.left'.
- Jika 'node' menjadi 'None', kembalikan 'None'.

Bukti Accepted

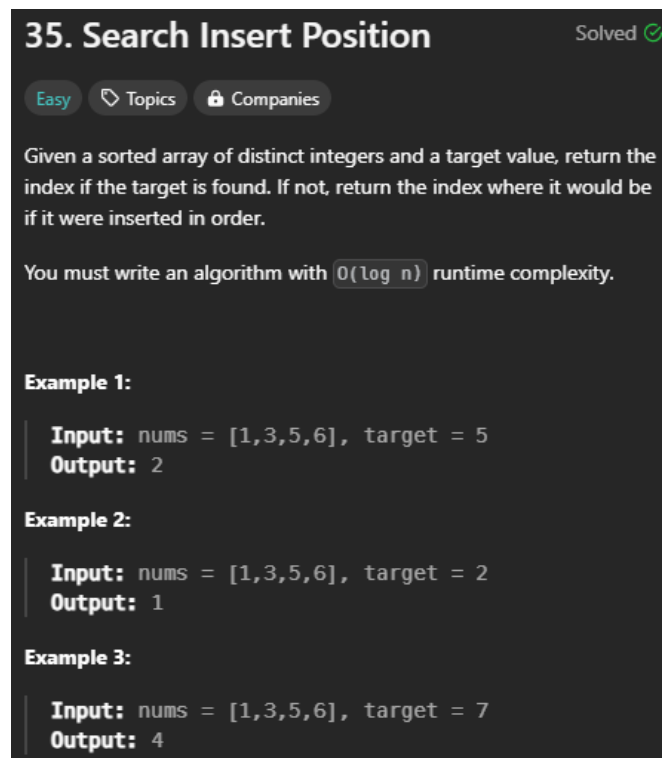
Berikut merupakan *screenshot* bukti bahwa solusi yang telah dibuat dapat berjalan dan diterima oleh sistem :



Gambar 1.3 screenshot accepted 13

14. Judul masalah : *Search Insert Position.*

Berikut merupakan *screenshot* dari masalah yang saya coba selesaikan :



gambar 1.1 screenshot masalah 14

Inti dari permasalahan ini yaitu : mencari target pada 'nums', jika ditemukan akan mengembalikan nilai index target.

Source Code

Berikut merupakan *screenshot* dari kode program yang digunakan sebagai solusi dari permasalahan ini :

```
Python ▾ 🔒 Auto
1  class Solution(object):
2      def searchInsert(self, nums, target):
3          kiri = 0
4          kanan = len(nums) - 1
5          while kiri <= kanan:
6              mid = (kiri + kanan) // 2
7              if nums[mid] == target:
8                  return mid
9              elif target < nums[mid]:
10                 kanan = mid - 1
11             else:
12                 kiri = mid + 1
13         return kiri
14         print("L200220269")
```

gambar 1.2 screenshot kode masalah 14

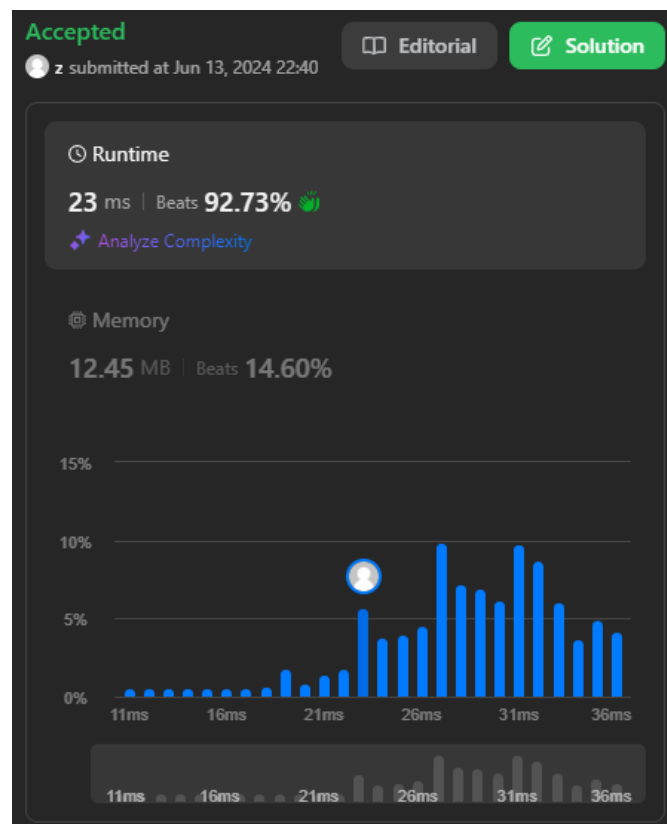
Penjelasan

Untuk menyelesaikan permasalahan ini, saya menggunakan logika solusi sebagai berikut:

- Membuat variabel 'kiri' dan 'kanan' untuk membuat batas atas dan batas bawah.
- Selama 'kiri' tidak melebihi 'kanan', hitung 'mid' sebagai index tengah dari bagian yang sedang dicari.
- Jika nilai 'nums[mid]' sama dengan target, kembalikan 'mid' karena target sudah ditemukan.
- Jika target lebih kecil dari 'nums[mid]', perbarui 'kanan' menjadi 'mid'-1 untuk mencari bagian kiri.
- Jika target lebih besar dari 'nums[mid]', perbarui 'kiri' menjadi 'mid'+1 untuk mencari bagian kanan.
- Setelah loop selesai, 'kiri' akan menjadi index dimana target harus disisipkan agar daftar tetap terurut.

Bukti Accepted

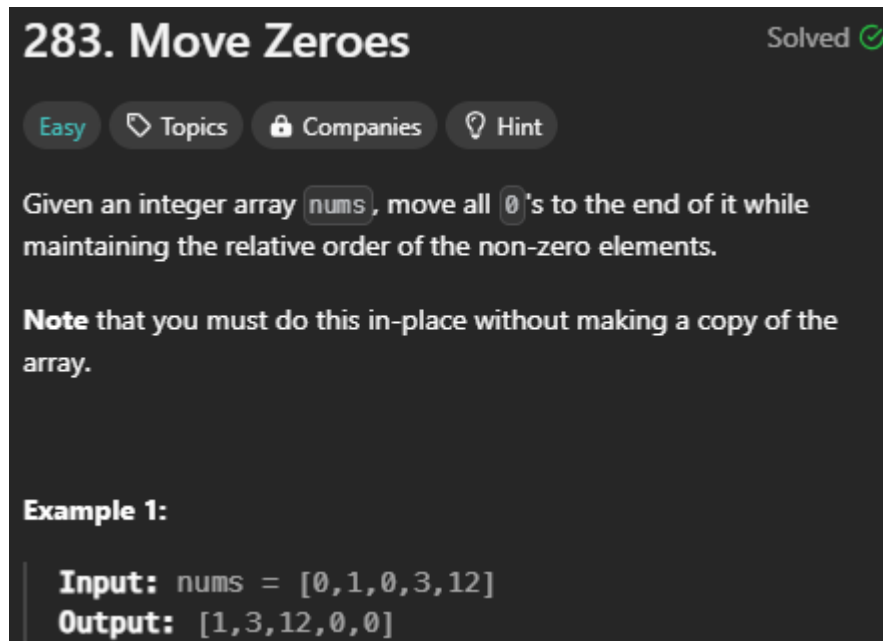
Berikut merupakan *screenshot* bukti bahwa solusi yang telah dibuat dapat berjalan dan diterima oleh sistem :



Gambar 1.3 screenshot accepted 14

15. Judul masalah : *Move Zeroes*.

Berikut merupakan *screenshot* dari masalah yang saya coba selesaikan :

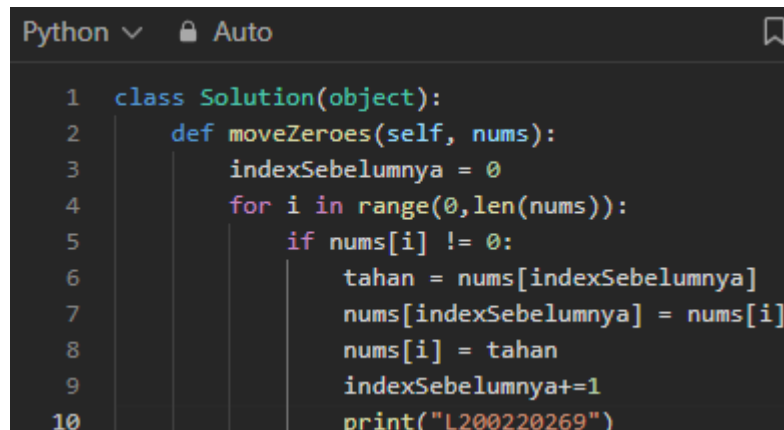


gambar 1.1 screenshot masalah 15

Inti dari permasalahan ini yaitu : memindahkan angka 0 ke bagian paling kanan.

Source Code

Berikut merupakan *screenshot* dari kode program yang digunakan sebagai solusi dari permasalahan ini :



gambar 1.2 screenshot kode masalah 15

Penjelasan

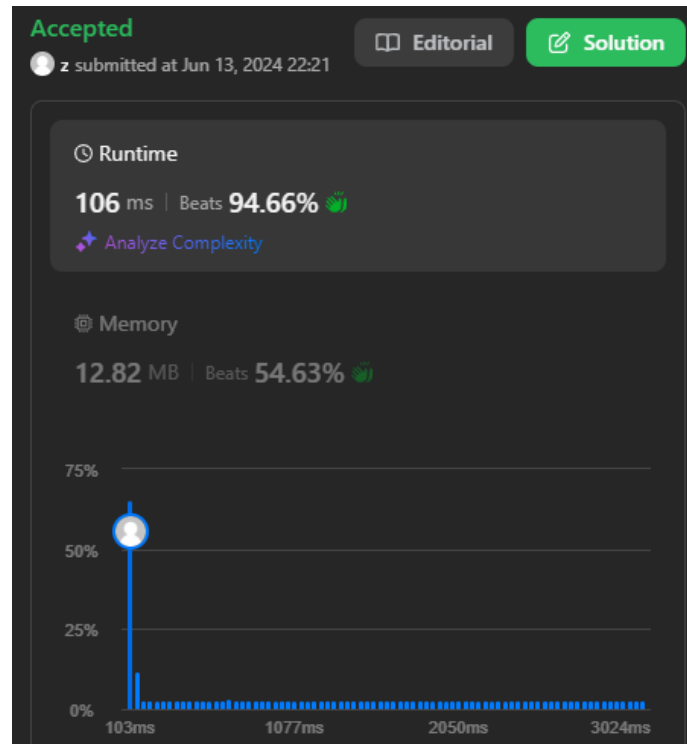
Untuk menyelesaikan permasalahan ini, saya menggunakan logika solusi sebagai berikut:

- Membuat variabel 'indexSebelumnya' dengan 0 untuk melacak posisi dimana elemen nol berikutnya akan ditempatkan.

- Melakukan loop index 'i' dari 0 hingga panjang daftar 'nums'.
- Jika 'nums[i]' bukan 0, tukar pada 'nums[indexSebelumnya]' dengan 'nums[i]'.

Bukti *Accepted*

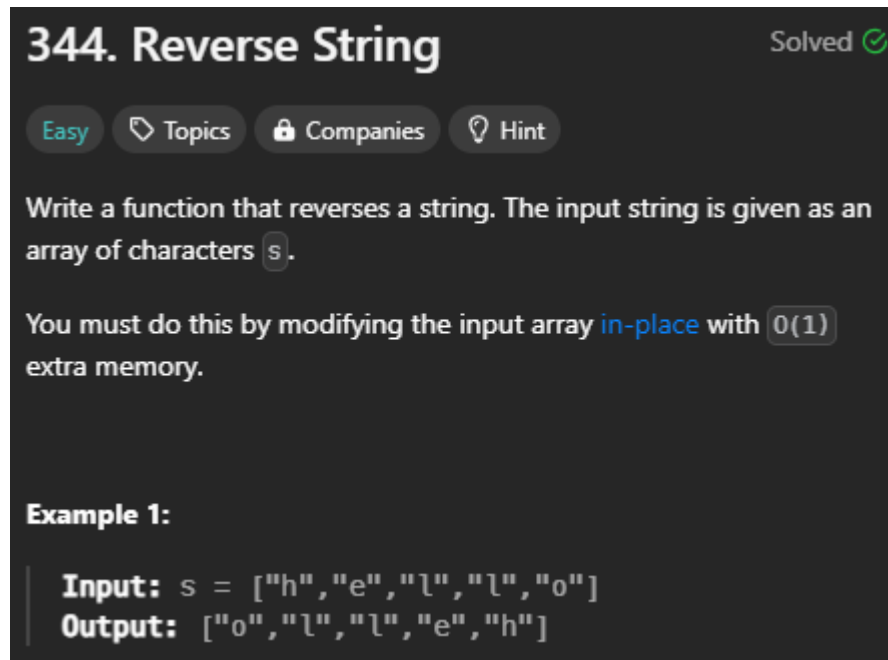
Berikut merupakan *screenshot* bukti bahwa solusi yang telah dibuat dapat berjalan dan diterima oleh sistem :



Gambar 1.3 screenshot accepted 15

16. Judul masalah : *Reverse String*.

Berikut merupakan *screenshot* dari masalah yang saya coba selesaikan :



gambar 1.1 screenshot masalah 16

Inti dari permasalahan ini yaitu : membalik *string*.

Source Code

Berikut merupakan *screenshot* dari kode program yang digunakan sebagai solusi dari permasalahan ini :

```
Python ▾ 🔒 Auto
1 class Solution(object):
2     def reverseString(self, s):
3         kiri = 0
4         kanan = len(s)-1
5         while kiri < kanan:
6             tahan = s[kiri]
7             s[kiri] = s[kanan]
8             s[kanan] = tahan
9             kiri+=1
10            kanan-=1
11            print("L200220269")
```

gambar 1.2 screenshot kode masalah 16

Penjelasan

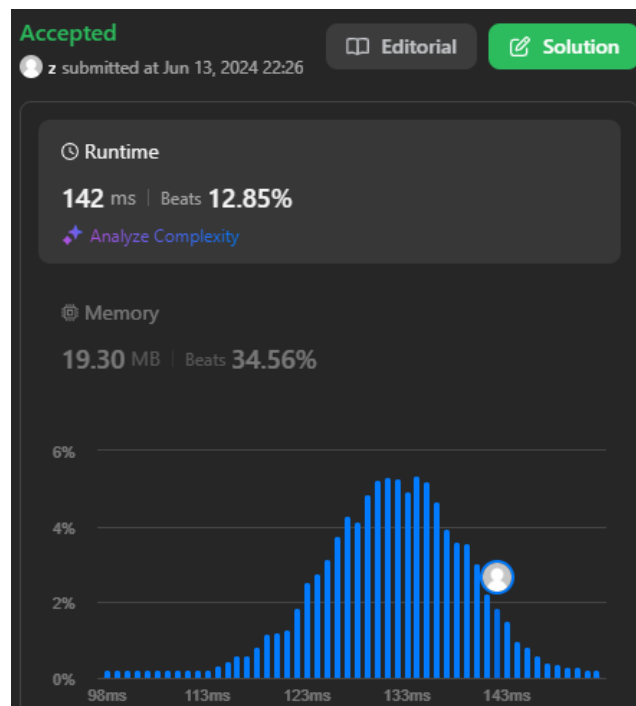
Untuk menyelesaikan permasalahan ini, saya menggunakan logika solusi sebagai berikut:

- Membuat variabel 'kiri' dan 'kanan' untuk membuat batas atas dan bawah.

- Selama ‘kiri’ lebih kecil dari ‘kanan’, lakukan penukaran di ‘s[kiri]’ dan ‘s[kanan]’.
- Setelah menukar, tambahkan ‘kiri’ dan kurangkan ‘kanan’ untuk bergerak menuju tengah daftar.

Bukti Accepted

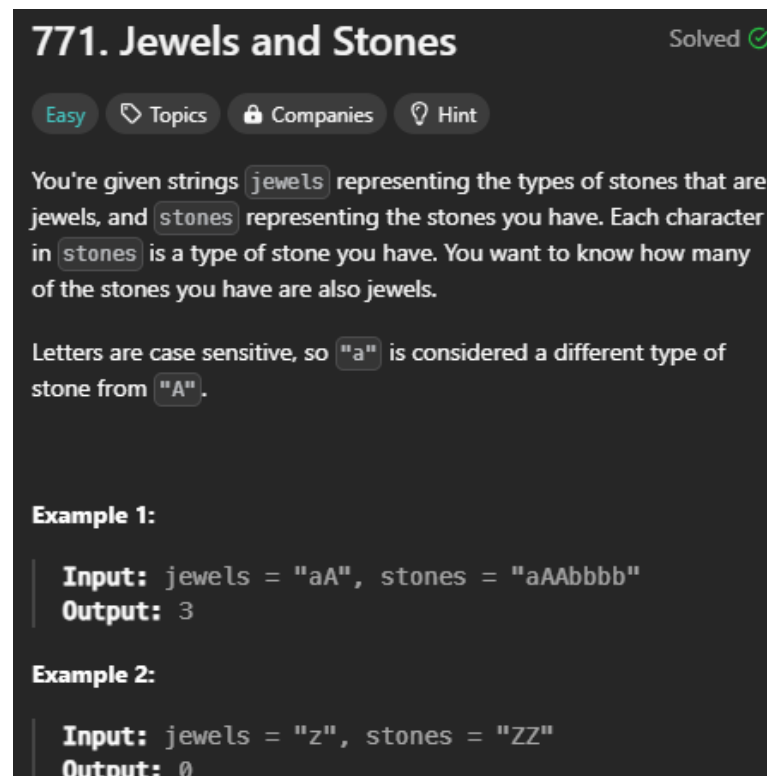
Berikut merupakan *screenshot* bukti bahwa solusi yang telah dibuat dapat berjalan dan diterima oleh sistem :



Gambar 1.3 screenshot accepted 16

17. Judul masalah : *Jewels and Stones*.

Berikut merupakan *screenshot* dari masalah yang saya coba selesaikan :



gambar 1.1 screenshot masalah 17

Inti dari permasalahan ini yaitu : mencari huruf yang ada pada *jewels* di *stones*.

Source Code

Berikut merupakan *screenshot* dari kode program yang digunakan sebagai solusi dari permasalahan ini :

```
Python ▾ 🔒 Auto
1 class Solution(object):
2     def numJewelsInStones(self, jewels, stones):
3         checkJewel = set(jewels)
4         total = 0
5         for i in stones:
6             if i in checkJewel:
7                 total += 1
8         return total
9         print("L200220269")
```

gambar 1.2 screenshot kode masalah 17

Penjelasan

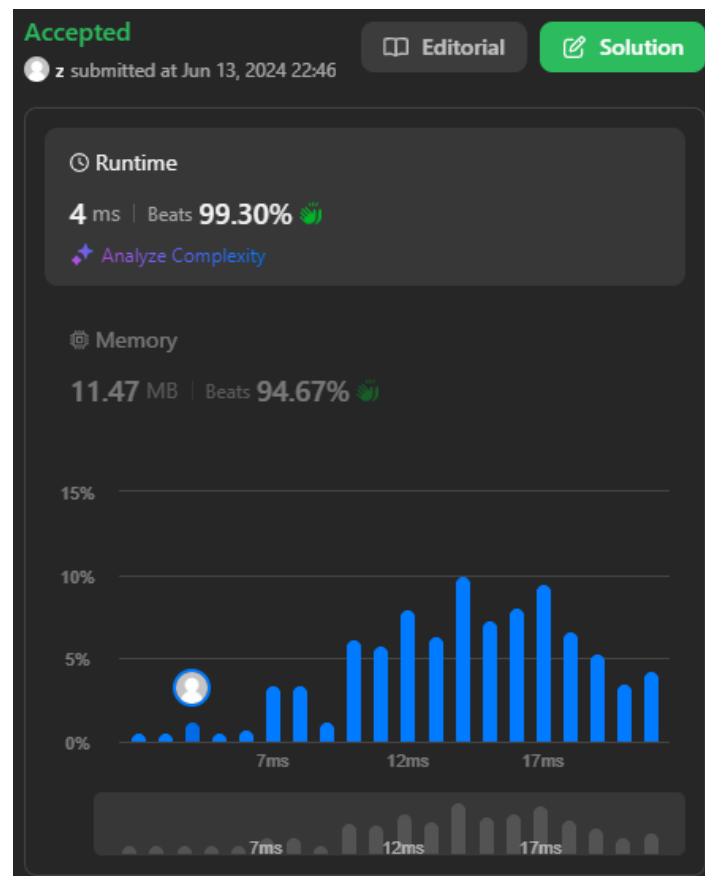
Untuk menyelesaikan permasalahan ini, saya menggunakan logika solusi sebagai berikut:

- Membuat variabel 'checkJewels' yang berisi 'set(jewels)'.

- Membuat variabel 'total' untuk menghitung 'jewels' dalam 'stones'.
- Lakukan perulangan setiap karakter 'i' dalam *string* 'stones'.
- Jika 'i' ada dalam 'checkJewels', tambahkan 'total' sebesar 1.
- Kembalikan 'total'.

Bukti Accepted

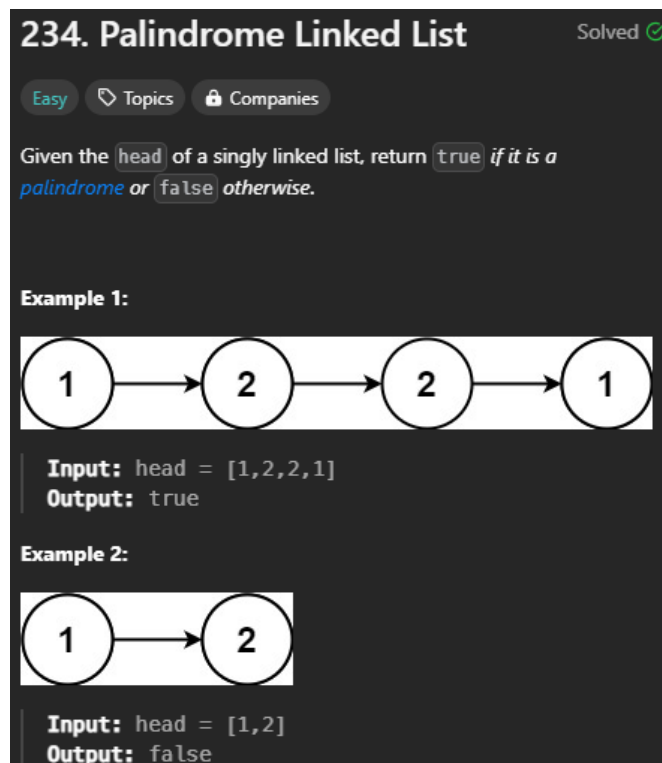
Berikut merupakan *screenshot* bukti bahwa solusi yang telah dibuat dapat berjalan dan diterima oleh sistem :



Gambar 1.3 screenshot accepted 17

18. Judul masalah : *Palindrome Linked List*.

Berikut merupakan *screenshot* dari masalah yang saya coba selesaikan :



gambar 1.1 screenshot masalah 18

Inti dari permasalahan ini yaitu : mengembalikan boolean *true* jika itu *palindrome*, jika tidak kembalikan boolean *false*.

Source Code

Berikut merupakan *screenshot* dari kode program yang digunakan sebagai solusi dari permasalahan ini :

```
Python v Auto
1 # Definition for singly-linked list.
2 # class ListNode(object):
3 #     def __init__(self, val=0, next=None):
4 #         self.val = val
5 #         self.next = next
6 class Solution(object):
7     def isPalindrome(self, head):
8         """
9         :type head: ListNode
10        :rtype: bool
11        """
12        list = []
13        while head:
14            list.append(head.val)
15            head = head.next
16        return list == list[::-1]
17        print("L200220269")
```

gambar 1.2 screenshot kode masalah 18

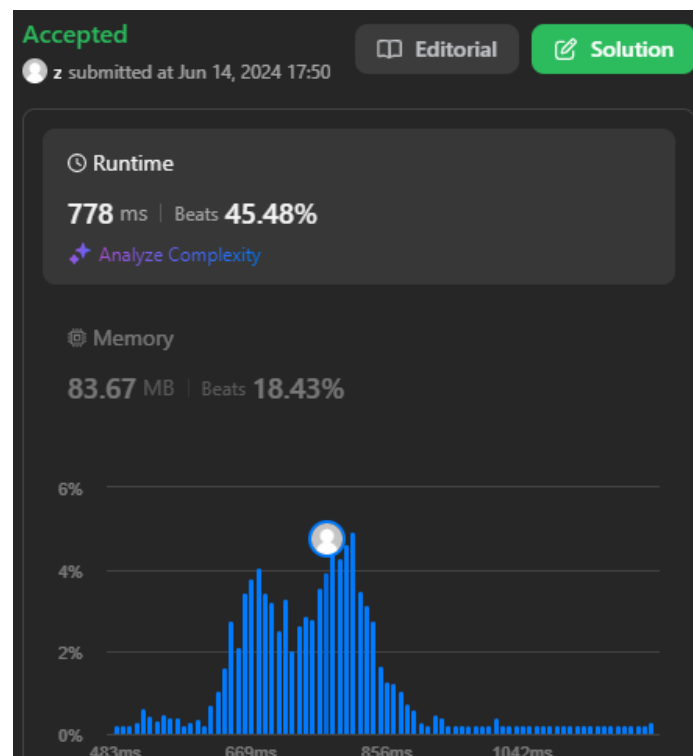
Penjelasan

Untuk menyelesaikan permasalahan ini, saya menggunakan logika solusi sebagai berikut:

- Membuat variabel 'list' untuk menyimpan nilai dari node linked list.
- Selama 'head' tidak 'None', tambahkan nilai node saat ini ('head.val') ke dalam list.
- Pindahkan 'head' ke node berikutnya.
- Bandingkan list asli dengan versi terbaliknya dan jika keduanya sama, berarti linked list adalah *palindrome* maka kembalikan *true*.
- Jika tidak kembalikan *false*.

Bukti Accepted

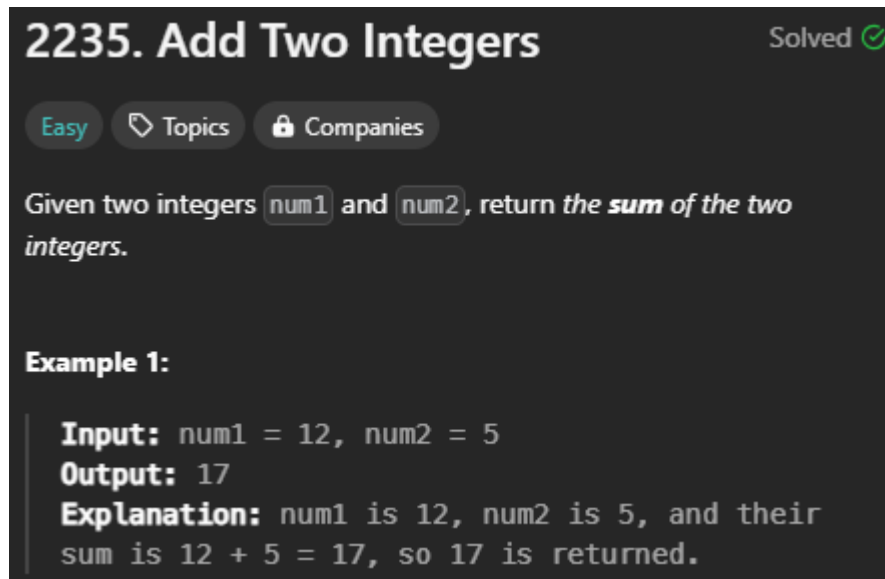
Berikut merupakan *screenshot* bukti bahwa solusi yang telah dibuat dapat berjalan dan diterima oleh sistem :



Gambar 1.3 screenshot accepted 18

19. Judul masalah : Add Two Integers.

Berikut merupakan *screenshot* dari masalah yang saya coba selesaikan :



gambar 1.1 screenshot masalah 19

Inti dari permasalahan ini yaitu : menambahkan dua *integers*.

Source Code

Berikut merupakan *screenshot* dari kode program yang digunakan sebagai solusi dari permasalahan ini :

```
Python v Auto
1 class Solution(object):
2     def sum(self, num1, num2):
3         """
4         :type num1: int
5         :type num2: int
6         :rtype: int
7         """
8         sum = num1 + num2
9         return sum
10    print("L200220269")
```

gambar 1.2 screenshot kode masalah 19

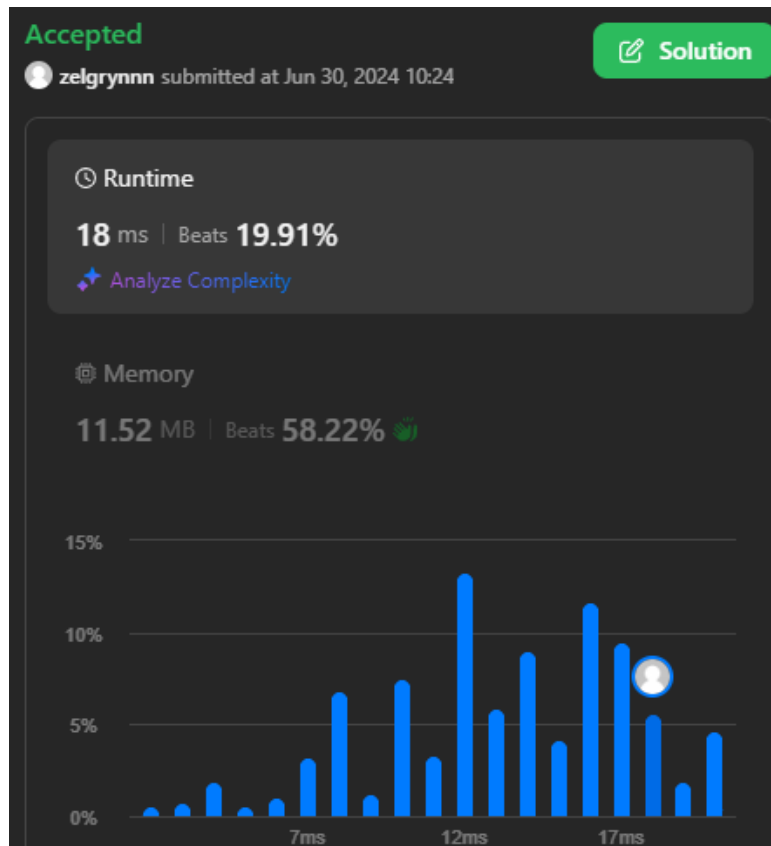
Penjelasan

Untuk menyelesaikan permasalahan ini, saya menggunakan logika solusi sebagai berikut:

- Membuat variabel 'sum' untuk menyimpan angka dari 'num1' dan 'num2'.
- Mengembalikan variabel 'sum'.

Bukti Accepted

Berikut merupakan *screenshot* bukti bahwa solusi yang telah dibuat dapat berjalan dan diterima oleh sistem :



Gambar 1.3 screenshot accepted 19

20. Judul masalah : *Sum Multiples*.

Berikut merupakan *screenshot* dari masalah yang saya coba selesaikan :

2652. Sum Multiples

Easy Topics Companies Hint

Given a positive integer n , find the sum of all integers in the range $[1, n]$ inclusive that are divisible by 3, 5, or 7.

Return an integer denoting the sum of all numbers in the given range satisfying the constraint.

Example 1:

Input: $n = 7$

Output: 21

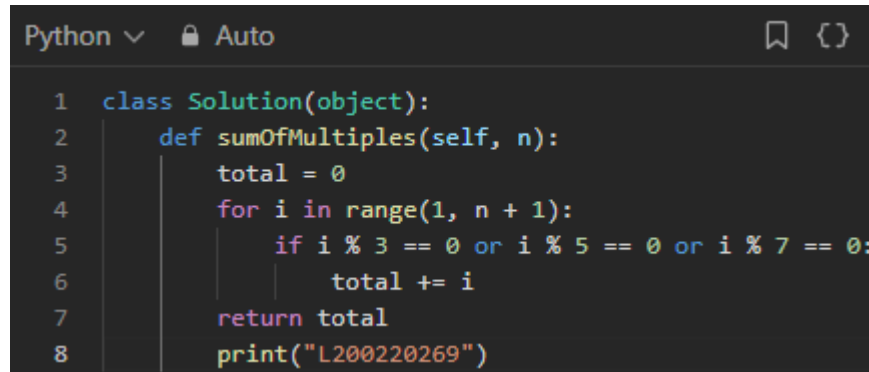
Explanation: Numbers in the range $[1, 7]$ that are divisible by 3, 5, or 7 are 3, 5, 6, 7. The sum of these numbers is 21.

gambar 1.1 screenshot masalah 20

Inti dari permasalahan ini yaitu : menghitung jumlah angka yang habis dibagi 3,5,7.

Source Code

Berikut merupakan *screenshot* dari kode program yang digunakan sebagai solusi dari permasalahan ini :



```
Python v Auto
1 class Solution(object):
2     def sumOfMultiples(self, n):
3         total = 0
4         for i in range(1, n + 1):
5             if i % 3 == 0 or i % 5 == 0 or i % 7 == 0:
6                 total += i
7         return total
8         print("200220269")
```

gambar 1.2 screenshot kode masalah 20

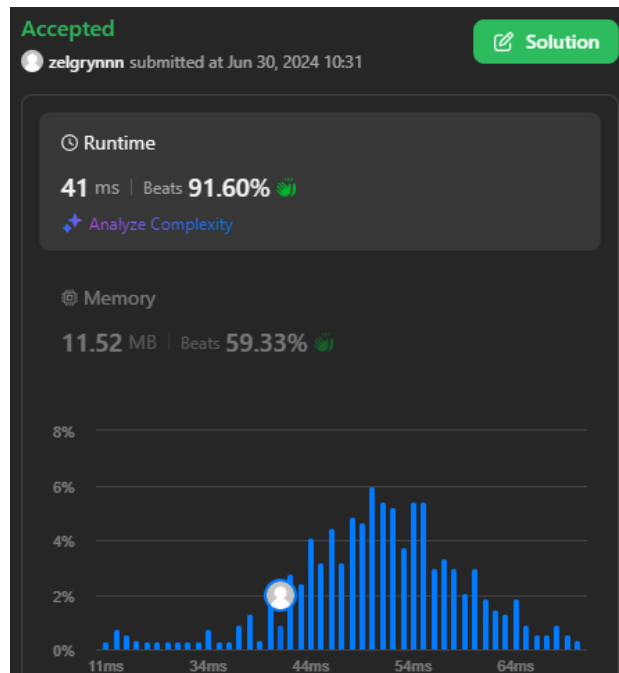
Penjelasan

Untuk menyelesaikan permasalahan ini, saya menggunakan logika solusi sebagai berikut:

- Membuat variabel 'total' untuk menyimpan jumlah bilangan yang habis dibagi 3,5,7.
- Melakukan perulangan dari *range* 1 sampai 'n+1'.
- Jika 'i' habis dibagi 3,4, atau 7 tambahkan 'i' ke 'total' dan kembalikan 'total'.

Bukti Accepted

Berikut merupakan *screenshot* bukti bahwa solusi yang telah dibuat dapat berjalan dan diterima oleh sistem :



Gambar 1.3 screenshot accepted 20

21. Judul masalah : *Divisible and Non-divisible Sums Difference.*

Berikut merupakan *screenshot* dari masalah yang saya coba selesaikan :

2894. Divisible and Non-divisible Sums Difference

Solved

Easy Topics Companies Hint

You are given positive integers n and m .

Define two integers, $num1$ and $num2$, as follows:

- $num1$: The sum of all integers in the range $[1, n]$ that are **not** divisible by m .
- $num2$: The sum of all integers in the range $[1, n]$ that are **divisible** by m .

Return the integer $num1 - num2$.

Example 1:

Input: $n = 10, m = 3$
Output: 19
Explanation: In the given example:
 - Integers in the range $[1, 10]$ that are not divisible by 3 are $[1, 2, 4, 5, 7, 8, 10]$, $num1$ is the sum of those integers = 37.
 - Integers in the range $[1, 10]$ that are divisible by 3 are $[3, 6, 9]$, $num2$ is the sum of those integers = 18.
 We return $37 - 18 = 19$ as the answer.

gambar 1.1 screenshot masalah 20

Inti dari permasalahan ini yaitu : menghitung selisih dari 'num1' berisi angka dari *range* [1,n] yang tidak habis dibagi 'm' dan 'num2' berisi angka dari *range* [1,n] yang habis dibagi 'm'.

Source Code

Berikut merupakan *screenshot* dari kode program yang digunakan sebagai solusi dari permasalahan ini :

gambar 1.2 screenshot kode masalah 20

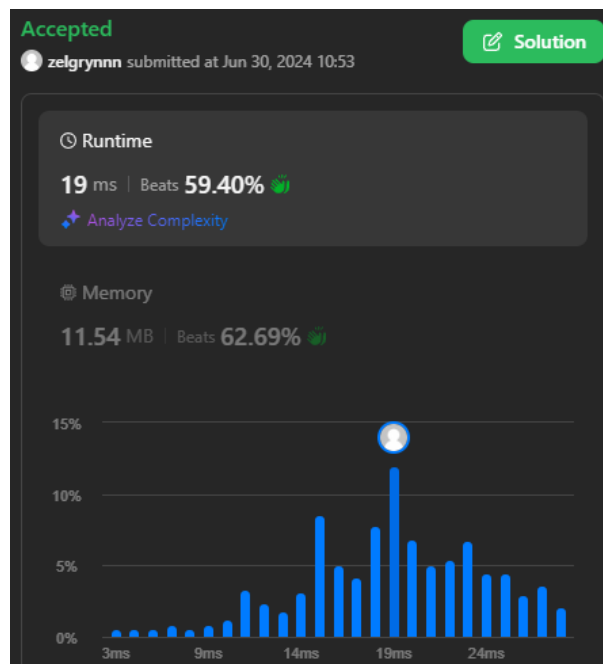
Penjelasan

Untuk menyelesaikan permasalahan ini, saya menggunakan logika solusi sebagai berikut:

- Membuat variabel 'num1' berisi angka dari *range* [1,n] yang tidak habis dibagi 'm'.
- Membuat variabel 'num2' berisi angka dari *range* [1,n] yang habis dibagi 'm'.
- Hitung selisih dan kembalikan nilai 'num1' dan 'num2'.

Bukti Accepted

Berikut merupakan *screenshot* bukti bahwa solusi yang telah dibuat dapat berjalan dan diterima oleh sistem :



Gambar 1.3 screenshot accepted 20