

LAPORAN PRAKTIKUM STRUKTUR DATA

MODUL KE-05

SEARCHING DALAM PYTHON



Disusun Oleh:

Nama : Restu Wibisono
NPM : 2340506061
Kelas : 03 (Tiga)

Program Studi S1 Teknologi Informasi

Fakultas Teknik, Universitas Tidar

Genap 2023/2024

I. Tujuan Praktikum

Adapun tujuan praktikum ini sebagai berikut :

1. Mahasiswa mampu menerapkan algoritma Searching pada bahasa pemrograman python

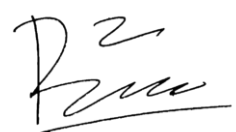
II. Dasar Teori

Searching dalam Python mengacu pada proses mencari nilai atau elemen tertentu dalam struktur data seperti array, list, atau struktur data lainnya. Tujuan dari pencarian adalah untuk menemukan lokasi atau keberadaan nilai yang dicari dalam struktur data tersebut. Proses pencarian ini penting dalam pemrograman karena seringkali perlu mencari data tertentu dalam kumpulan data yang besar untuk melaksanakan operasi tertentu. Python menyediakan berbagai algoritma pencarian yang dapat digunakan untuk mencari nilai dalam struktur data, seperti Linier Search, Binary Search, Jump Search, dan lainnya.

Kedua, mempelajari Binary Search, yang merupakan algoritma efisien untuk mencari elemen dalam array yang sudah diurutkan. Binary Search membandingkan elemen tengah array dengan elemen yang dicari, kemudian menentukan apakah pencarian dilanjutkan di setengah kiri atau kanan array. Dengan kompleksitas waktu terbaik $O(1)$ dan terburuk $O(\log n)$, Binary Search adalah pilihan yang baik untuk array yang besar.

Selanjutnya, ada Jump Search, sebuah variasi dari Binary Search yang berguna untuk array besar yang sudah diurutkan. Algoritma ini melompati elemen-elemen array dengan ukuran loncatan tertentu, kemudian melakukan pencarian linier di dalam blok yang kemungkinan mengandung elemen yang dicari. Jump Search memiliki kompleksitas waktu $O(\sqrt{n})$, menjadikannya efisien untuk array yang besar.

Tanda Tangan



III. Hasil dan Pembahasan

1. Linier Search

```
# Linier Search
def LinierSearch(arr, N, x):

    for i in range(0, N):
        if (arr[i] == x):
            return i
    return -1

# Driver code
if __name__ == '__main__':
    arr = [20, 10, 30, 50, 50]
    x = 10
    N = len(arr)

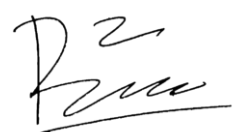
    # Dunction call
    result = LinierSearch(arr, N, x)
    if (result == -1):
        print("Element yang dicari tidak ada dalam array")
    else:
        print("Element yang dicari berada pada indeks ke-", result)

Element yang dicari berada pada indeks ke- 1
```

(Gambar 3.1)

- Memuat fungsi 'LinierSearch' yang mempunyai tiga parameter, 'arr' adalah dimana akan dilakukan pencarian, 'N' adalah panjang array, dan 'x' adalah elemen yang akan dicari.
- Loop 'for' akan melakukan iterasi melalui setiap elemen dalam array dari indeks 0 sampai N-
- Selanjutnya 'if (arr[i] == x):' berfungsi untuk memeriksa elemen pada indeks ke-i nilainya sama dengan yang dicari dalam 'x'.
- Kemudian pada 'return i' jika elemen ditemukan maka fungsi akan mengembalikan indeks dari elemen tersebut.
- lalu 'return -1' berfungsi jika pencarian telah mencapai array tetapi tidak menemukan nilai elemen yang dicari, fungsi ini akan mengembalikan dan menunjukkan bahwa elemen tidak ditemukan.

Tanda Tangan



- 'if __name__ == '__main__'' adalah skrip yang dijalankan secara langsung atau diimport sebagai modul, yang akan berfungsi untuk mengeksekusi blok kode yang ada dibawahnya.
- 'arr = [20, 10, 30, 50, 50]' Inisialisasi array 'arr' yang akan dijadikan sebagai objek pencarian.
- 'x = 10' Nilai yang akan dicari dalam array.
- 'N = len(arr)' Mendapatkan panjang array arr dan menyimpannya dalam variabel N.
- Kemudian memanggil fungsi 'LinierSearch' dengan parameter yang sesuai dan menyimpan hasilnya dalam variabel 'result'.
- Program akan memeriksa dengan fungsi 'if (result == -1)' untuk mengembalikan nilai -1 jika elemen tidak ditemukan di dalam array.
- Jika tidak di temukan akan mencetak pesan 'Element yang dicari tidak ada dalam array'.
- Jika elemen yang dicari ada maka program akan mencetak string 'Element yang dicari berada pada indeks keresult'.

2. Binary Search

```
# Algoritma Pencarian Biner Iteratif

def BinerSearch(arr, l, r, x):
    while l <= r:
        mid = l + (r - l) // 2;

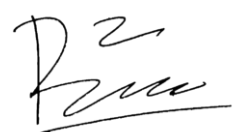
        # Check if x is present at mid
        if arr[mid] == x:
            return mid

        # If x is greater, ignore left half
        elif arr[mid] < x:
            l = mid + 1

        # If x is smaller, ignore right half
        else:
            r = mid - 1
```

(Gambar 3.2.1)

Tanda Tangan



```

    # If we reach here, then the element was not present
    return -1

# Driver code
if __name__ == "__main__":
    arr = [2, 5, 8, 12, 16, 23, 38, 56, 72, 91]
    x = 72

    # Function call
    result = BinerSearch(arr, 0, len(arr) - 1, x)
    if result != -1:
        print("Element yang di cari berada pada index ke-", result)
    else:
        print("Element yang di cari tidak ada dalam array")

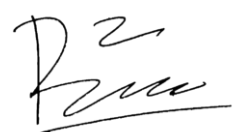
Element yang di cari berada pada index ke- 8

```

(Gambar 3.2.2)

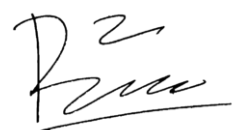
- Memuat 'def BinerSearch(arr, l, r, x):' yang menerima empat parameter: 'arr' adalah array yang sudah diurutkan, 'l' indeks awal dari subarray yang akan diperiksa, 'r' indeks akhir dari subarray yang akan diperiksa, dan 'x' elemen yang akan dicari.
- Memulai loop 'while' yang terus berjalan sampai indeks awal ('l') tidak melebihi indeks akhir ('r') pada 'while l <= r:'.
- 'mid = l + (r - l) // 2;' berfungsi untuk menghitung indeks tengah ('mid') dari subarray yang sedang diperiksa.
- Kemudian program akan memeriksa dengan 'if arr[mid] == x:' apakah elemen pada indeks tengah ('mid') sama dengan nilai yang dicari 'x'. Jika ya, maka elemen pada program tersebut ditemukan dan fungsi mengembalikan indeksnya.
- Jika elemen pada indeks tengah 'elif arr[mid] < x:' lebih kecil dari nilai yang dicari 'x', hasilnya hanya bagian kanan dari subarray yang mana menjadi pertimbangan untuk pencarian selanjutnya.

Tanda Tangan



- Menggeser indeks awal dengan $l = \text{mid} + 1$ ('l') ke kanan, yang mana setelah indeks tengah ('mid').
- Jika elemen pada indeks tengah ('mid') lebih besar dari nilai yang dicari 'x', program akan memakai 'else:' maka hanya bagian kiri sebelah dari subarray yang akan dipertimbangkan sebagai pencarian selanjutnya.
- Lalu $r = \text{mid} - 1$ berfungsi untuk menggeser indeks akhir ('r') ke kiri, yaitu sebelum indeks tengah ('mid').
- Pencarian menggunakan 'return -1' yang telah mencapai akhir dari loop 'while' tanpa menemukan elemen yang dicari, dengan fungsi mengembalikan nilai -1 untuk melihat bahwa elemen tidak ditemukan dalam array.
- 'if __name__ == '__main__'' adalah skrip yang dijalankan secara langsung atau diimport sebagai modul, yang akan berfungsi untuk mengeksekusi blok kode yang ada dibawahnya.
- Inisialisasi array 'arr' yang sudah diurutkan menjadi `arr = [2, 5, 8, 12, 16, 23, 38, 56, 72, 91]`.
- Nilai yang akan dicari dalam array adalah `x = 72`.
- `result = BinerSearch(arr, 0, len(arr) - 1, x)` Memanggil fungsi 'BinerSearch' dengan parameter yang sesuai dan menyimpan hasilnya dalam variabel 'result'.
- Program memeriksa apakah hasil pencarian pada `if result != -1:` mengembalikan nilai yang valid (bukan -1), yang akan berarti elemen ditemukan dalam array.
- Jika elemen yang dicari ada maka program akan mencetak string 'Element yang dicari berada pada indeks keresult'.
- Jika tidak ditemukan akan mencetak pesan 'Element yang dicari tidak ada dalam array'.

Tanda Tangan



3. Binary Rekursif

```
# Algoritma Pencarian Biner Rekursif

def BinerSearch(arr, l, r, x):
    if r >= l:
        mid = l + (r - l) // 2

        # If the element is present at the middle itself
        if arr[mid] == x:
            return mid

        # If the element is smaller than mid, then it
        # can only be present in the left subarray
        elif arr[mid] > x:
            return BinerSearch(arr, l, mid - 1, x)

        # Else the element can only be present
        # in the right subarray
        else:
            return BinerSearch(arr, mid + 1, r, x)

    else:
        # Element is not present in array
        return -1

# Driver code
if __name__ == "__main__":
    arr = [2, 5, 8, 12, 16, 23, 38, 56, 72, 91]
    x = 72

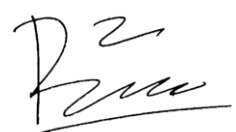
    # Function call
    result = BinerSearch(arr, 0, len(arr) - 1, x)
    if result != -1:
        print("Element yang di cari berada pada index ke-", result)
    else:
        print("Element yang di cari tidak ada dalam array")

Element yang di cari berada pada index ke- 8
```

(Gambar 3.3)

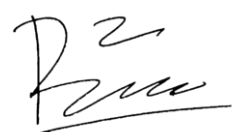
- Memuat 'def BinerSearch(arr, l, r, x):' yang menerima empat parameter: 'arr' adalah array yang sudah diurutkan, 'l' indeks awal dari subarray yang akan diperiksa, 'r' indeks akhir dari subarray yang akan diperiksa, dan 'x' elemen yang akan dicari.

Tanda Tangan



- Program akan melakukan pencarian dengan 'if r >= l:' yang dilakukan selama indeks awal ('l') tidak melebihi indeks akhir ('r'). Saat 'l' melebihi 'r', subarray sudah kosong dan pencarian akan berhenti.
- 'mid = l + (r - l) // 2;' berfungsi untuk menghitung indeks tengah ('mid') dari subarray yang sedang diperiksa.
- Kemudian program akan memeriksa dengan 'if arr[mid] == x:' apakah elemen pada indeks tengah ('mid') sama dengan nilai yang dicari 'x'. Jika ya, maka elemen pada program tersebut ditemukan dan fungsi mengembalikan indeksnya.
- Jika elemen pada indeks tengah 'elif arr[mid] < x:' lebih kecil dari nilai yang dicari 'x', hasilnya hanya bagian kanan dari subarray yang mana menjadi pertimbangan untuk pencarian selanjutnya.
- 'return BinerSearch(arr, l, mid - 1, x)' berfungsi secara rekursif yang dipanggil untuk mencari elemen di subarray kiri, yaitu dari indeks awal ('l') sampai indeks tengah dikurangi satu ('mid - 1').
- Jika elemen pada indeks tengah ('mid') lebih besar dari nilai yang dicari 'x', program akan memakai 'else:' maka hanya bagian kiri sebelah dari subarray yang akan dipertimbangkan sebagai pencarian selanjutnya.
- 'return BinerSearch(arr, mid + 1, r, x)' akan berfungsi secara rekursif yang dipanggil untuk mencari pada elemen di subarray kanan, yaitu dari indeks tengah yang ditambah satu ('mid + 1') hingga indeks akhir ('r').
- 'return -1' Jika pencarian telah mencapai akhir dari rekursi tanpa menemukan elemen yang dicari, fungsi mengembalikan nilai -1 untuk menunjukkan bahwa elemen tidak ditemukan dalam array.
- 'if __name__ == '__main__':' adalah skrip yang dijalankan secara langsung atau diimport sebagai modul, yang akan berfungsi untuk mengeksekusi blok kode yang ada dibawahnya.

Tanda Tangan



- Inisialisasi array 'arr' yang sudah diurutkan menjadi 'arr = [2, 5, 8, 12, 16, 23, 38, 56, 72, 91]'.
- 'x = 72' Nilai yang akan dicari dalam array.
- 'result = BinerSearch(arr, 0, len(arr) - 1, x)' Memanggil fungsi 'BinerSearch' dengan parameter yang sesuai dan menyimpan hasilnya dalam variabel 'result'.
- Program memeriksa apakah hasil pencarian pada 'if result != -1:' mengembalikan nilai yang valid (bukan -1), yang akan berarti elemen ditemukan dalam array.
- Jika elemen yang dicari ada maka program akan mencetak string 'Element yang dicari berada pada indeks keresult'.
- Jika tidak ditemukan akan mencetak pesan 'Element yang dicari tidak ada dalam array'.

4. Jump Search

```
# Jump search
import math

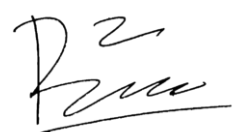
def JumpSearch(arr, x, n):
    # Finding block size to be jumped
    step = math.sqrt(n)

    # Finding the block where element is
    # present (if it is present)
    prev = 0
    while arr[int(min(step, n) - 1)] < x:
        prev = step
        step += math.sqrt(n)
        if prev >= n:
            return -1

    # Doing a linear search for x in
    # block beginning with prev.
    while arr[int(prev)] < x:
        prev += 1
```

(Gambar 3.41)

Tanda Tangan



```

        # If we reached next block or end
        # of array, element is not present.
        if prev == min(step, n):
            return -1

        # If element is found
        if arr[int(prev)] == x:
            return prev

        return -1

# Driver code
if __name__ == "__main__":
    arr = [0, 1, 1, 2, 3, 5, 8, 13, 21,
           34, 55, 89, 144, 233, 377, 610]
    x = 55
    n = len(arr)

    # Find the index of 'x' using Jump Search
    index = JumpSearch(arr, x, n)

    # Print the index where 'x' is located
    print("Number", x, "is at index", int(index))

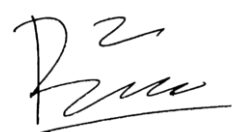
```

Number 55 is at index 10

(Gambar 3.4.2)

- Mengimpor modul 'math' untuk menggunakan fungsi matematika.
- Memuat fungsi 'JumpSearch' yang mempunyai tiga parameter, 'arr' adalah dimana akan dilakukan pencarian, 'n' adalah panjang array, dan 'x' adalah elemen yang akan dicari.
- Untuk menghitung ukuran blok menggunakan 'step = math.sqrt(n)' yang akan dilewati dengan menggunakan fungsi akar kuadrat dari panjang array 'n'.
- Inisialisasi variabel 'prev' yang akan menyimpan indeks terakhir dari blok yang sudah dilewati.
- Program akan melakukan pencarian linier menggunakan 'while arr[int(min(step, n) 1)] < x:' untuk menemukan blok yang mungkin mengandung elemen yang dicari. Iterasi akan dilakukan selama nilai

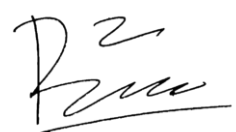
Tanda Tangan



elemen di indeks `int(min(step, n) - 1)` kurang dari nilai yang dicari `x`.

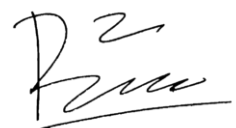
- Mengupdate nilai `prev` dengan ukuran blok `step`.
- Selanjutnya menambahkan ukuran blok ke `step += math.sqrt(n)` untuk memindahkan ke blok berikutnya.
- Program akan memeriksa apakah blok saat ini sudah melebihi panjang array dengan `if prev >= n`. Jika ya, elemen yang dicari tidak di temukan dalam array.
- Melakukan pencarian linier di dalam blok saat ini dengan `while arr[int(prev)] < x` untuk menemukan elemen yang dicari. Iterasi di lakukan selama nilai di elemen pada indeks `int(prev)` kurang dari nilai yang dicari `x`.
- Memeriksa dengan `if prev == min(step, n)` apakah iterasi pencarian pada linier sudah mencapai akhir blok. Jika ya, elemen yang dicari tidak di temukan dalam blok saat ini.
- Memeriksa apakah elemen yang ditemukan pada indeks `int(prev)` sama dengan nilai yang dicari `x`. Jika ya, elemen tersebut di temukan lalu fungsi akan mengembalikan indeksnya.
- `return -1` Jika pencarian telah mencapai akhir dari rekursi tanpa menemukan elemen yang dicari, fungsi mengembalikan nilai -1 untuk menunjukkan bahwa elemen tidak ditemukan dalam array.
- `if __name__ == '__main__'` adalah skrip yang dijalankan secara langsung atau diimport sebagai modul, yang akan berfungsi untuk mengeksekusi blok kode yang ada dibawahnya.
- Inisialisasi array `arr` yang sudah diurutkan `arr = [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610]`
- `x = 55` Nilai yang akan dicari dalam array.
- Mendapatkan panjang array `arr` dan menyimpannya dalam variabel `n`.

Tanda Tangan



- `'index = JumpSearch(arr, x, n)'` akan memanggil fungsi `'JumpSearch'` dengan parameter sesuai serta akan menyimpan hasilnya dalam variabel `'index'`.
- Terakhir adalah program untuk mencetak hasil pencarian dengan `'print("Number", x, "is at index", int(index))'`, yaitu indeks dimana nilai `'x'` ditemukan dalam array.

Tanda Tangan

A handwritten signature in black ink, consisting of stylized cursive letters, located below the text 'Tanda Tangan'.

IV. Studi Kasus

1. Studi Kasus 1

```
# Studi Kasus 1
def findMinMax(numbers):
    minNum = maxNum = numbers[0]
    for num in numbers[1:]:
        if num < minNum:
            minNum = num
        if num > maxNum:
            maxNum = num
    return minNum, maxNum

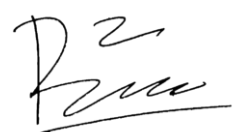
# Contoh penggunaan:
scores = [90, 85, 88, 92, 78, 88, 95, 89, 84, 96,
          87, 93, 82, 90, 91, 88, 85, 94, 92, 90,
          86, 87, 88, 92, 89, 90, 91, 92, 93, 94,
          85, 86, 87, 88, 89, 90, 91, 92, 93, 94]
minScore, maxScore = findMinMax(scores)
print(f"Nilai terendah: {minScore}")
print(f"Nilai tertinggi: {maxScore}")

Nilai terendah: 78
Nilai tertinggi: 96
```

(Gambar 4.1)

- Pertama mendefinisikan fungsi 'findMinMax' yang mempunyai menerima satu parameter yaitu 'numbers'.
- Variabel 'minNum' dan 'maxNum' dengan nilai pertama list 'numbers'. Bahwa nilai pertama adalah nilai minimum dan maksimum sementara.
- Iterasi 'for num in numbers[1:]' akan dimulai dari indeks kedua sampai akhir list, menginisialisasi 'minNum' dan 'maxNum' dengan nilai pertama.
- Memeriksa apakah nilai saat ini 'if num < minNum:' lebih kecil dari 'minNum'. Jika ya, nilai tersebut akan menjadi nilai minimum baru.
- Memeriksa apakah nilai saat ini 'if num > maxNum:' lebih besar dari 'maxNum'. Jika ya, nilai tersebut akan menjadi nilai maksimum baru.

Tanda Tangan



- Untuk mengembalikan nilai minimum dan maksimum menggunakan 'return minNum, maxNum' yang telah ditemukan.
- Inisialisasi list 'scores' yang berisi nilai-nilai skor mahasiswa 'scores = [90, 85, 88, 92, 78, 88, 95, 89, 84, 96, 87, 93, 82, 90, 91, 88, 85, 94, 92, 90, 86, 87, 88, 92, 89, 90, 91, 92, 93, 94, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94]'
- Memanggil fungsi 'findMinMax' dengan parameter 'scores' untuk mencari nilai terendah dan tertinggi di dalam list tersebut, lalu menyimpan hasilnya di dalam variabel 'minScore' dan 'maxScore'.
- 'print(f"Nilai terendah: {minScore}")' Mencetak nilai terendah yang ditemukan.
- 'print(f"Nilai tertinggi: {maxScore}")' Mencetak nilai tertinggi yang ditemukan.

2. Studi Kasus 2

```
# Studi Kasus 2
def findMissingRepeating(arr):
    n = len(arr)
    temp = [0] * n
    repeating = missing = -1

    for i in arr:
        if temp[i-1] == 0:
            temp[i-1] = 1
        else:
            repeating = i

    for i in range(n):
        if temp[i] == 0:
            missing = i + 1

    return missing, repeating

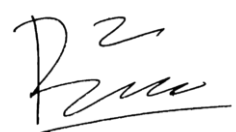
# Contoh penggunaan:
arr = [3, 1, 3]
missing, repeating = findMissingRepeating(arr)
print(f"Hilang = {missing}, Terulang = {repeating}")

arr = [4, 3, 6, 2, 1, 1]
missing, repeating = findMissingRepeating(arr)
print(f"Hilang = {missing}, Terulang = {repeating}")

Hilang = 2, Terulang = 3
Hilang = 5, Terulang = 1
```

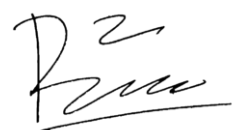
(Gambar 4.2)

Tanda Tangan



- Tentukan fungsi `findMissingRepeating(arr)`: adalah definisi dari fungsi `findMissingRepeating` yang akan menerima satu parameter, `arr`, yang merupakan sebuah daftar angka.
- Hitung panjang dari daftar `arr` dan simpan dalam variabel `n`.
- Membuat sebuah daftar `temp` dengan panjang sama dengan daftar `arr` dan inisialisasikan dengan nilai-nol. Dengan ini akan digunakan untuk tanda keberadaan setiap elemen dari `arr`.
- Inisialisasikan variabel `repeating` dan `missing` dengan nilai -1. Variabel `repeating` yang akan menyimpan nilai yang terulang, dan variabel `missing` akan menyimpan nilai yang hilang.
- Iterasi melalui setiap elemen dalam daftar `arr`.
- Periksa elemen `i` dari `arr` sudah pernah ditemukan sebelumnya yang berfungsi memeriksa nilai dari `temp[i-1]`, yang menandakan keberadaan elemen `i` dalam daftar.
- Tandai keberadaan elemen `i` dengan mengubah nilai dari `temp[i-1]` menjadi 1.
- Saat nilai dari `temp[i-1]` bukan 0 (berarti elemen `i` sudah pernah ditemukan sebelumnya), maka elemen `i` yaitu elemen yang terulang.
- Perbarui nilai dari variabel `repeating` dengan elemen yang terulang.
- Iterasi melalui setiap indeks dari 0 hingga `n-1`.
- Periksa elemen pada indeks `i` tidak ditemukan dalam daftar `arr`. Jika nilai dari `temp[i]` masih 0, maka pada elemen dengan nilai `i+1` adalah yang hilang.
- Perbarui nilai dari variabel `missing` dengan nilai yang hilang.
- Mengembalikan nilai yang hilang serta nilai yang terulang.
- Inisialisasi daftar `arr` dengan beberapa nilai numerik.
- Panggil fungsi `findMissingRepeating` dengan parameter `arr` yang berfungsi untuk mencari nilai yang hilang dan nilai yang terulang

Tanda Tangan



dalam daftar tersebut, serta simpan hasilnya dalam variabel `missing` dan `repeating`.

- Mencetak nilai yang hilang dan nilai yang terulang yang telah ditemukan.
- Mengulangi langkah yang sama untuk daftar kedua `arr`.

3. Studi Kasus 3

```
# Studi Kasus 3
def pairInSortedRotated(arr, x):
    n = len(arr)

    for i in range(n):
        if arr[i] > arr[i + 1]:
            break

    l = (i + 1) % n
    r = i

    while l != r:
        if arr[l] + arr[r] == x:
            return True

        if arr[l] + arr[r] < x:
            l = (l + 1) % n
        else:
            r = (n + r - 1) % n

    return False

# Contoh penggunaan:
arr = [11, 15, 6, 8, 9, 10]
x = 16
print(pairInSortedRotated(arr, x))

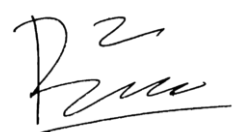
arr = [11, 15, 26, 38, 9, 10]
x = 35
print(pairInSortedRotated(arr, x))

arr = [11, 15, 26, 38, 9, 10]
x = 45
print(pairInSortedRotated(arr, x))

True
True
False
```

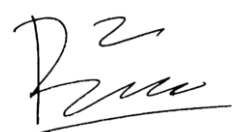
(Gambar 3.3)

Tanda Tangan



- Fungsi 'pairInSortedRotated(arr, x):' Ini adalah definisi dari fungsi 'pairInSortedRotated' yang akan menerima dua parameter yaitu 'arr', yang merupakan array sudah diurutkan dan diputar, dan 'x', yang merupakan nilai yang ingin dicari pasangannya.
- Menghitung panjang array 'arr' dan menyimpannya dalam variabel 'n'.
- Iterasi melalui setiap elemen array 'arr'.
- Memeriksa rotasi di dalam array nilai elemen saat ini lebih besar dari elemen berikutnya, maka akan menemukan titik rotasi dan iterasi dihentikan.
- Menghitung indeks awal ('l') pada bagian yang sudah diputar dengan menggunakan operasi modulus yang memastikan bahwa nilai 'l' tetap berada dalam rentang yang valid (0 hingga 'n-1').
- Menghitung indeks akhir ('r') dari bagian yang sudah diputar.
- Iterasi dilakukan selama indeks 'l' dan 'r' tidak sama.
- Memeriksa jumlah dua nilai yang berada di indeks 'l' dan 'r' sama dengan yang dicari 'x'. Jika ya, pasangan nilai tersebut ditemukan dan fungsi mengembalikan 'True'.
- Memeriksa jumlah dua nilai kurang dari nilai 'x'. Jika ya, harus mencari pasangan dengan nilai yang lebih besar, sehingga indeks 'l' diperbarui.
- Jika jumlah dua nilai tersebut lebih besar dari nilai 'x', harus mencari pasangan dengan nilai yang lebih kecil, sehingga indeks 'r' diperbarui.
- Jika setelah iterasi selesai tidak ditemukan pasangan nilai yang jumlahnya sama dengan nilai 'x', fungsi mengembalikan 'False'.
- Contoh penggunaan: Mendefinisikan tiga array yang berbeda dan nilai 'x' yang ingin dicari pasangannya.
- Memanggil fungsi 'pairInSortedRotated' dengan parameter 'arr' dan 'x', dan mencetak hasilnya.

Tanda Tangan



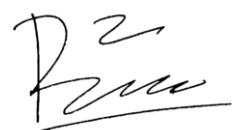
V. Kesimpulan

Pencarian struktur data praktis dengan Python memberikan pemahaman mendalam tentang berbagai algoritma pencarian yang digunakan dalam pemrograman. Dari praktikum ini diketahui bahwa pencarian adalah proses mencari nilai-nilai tertentu dalam suatu kumpulan data. Dalam implementasi Python, berbagai algoritma pencarian seperti pencarian linier dan pencarian biner diterapkan untuk mencari elemen dalam daftar.

Selain itu, praktikum ini juga memberikan pemahaman mengenai kompleksitas waktu pada setiap algoritma pencarian. Dengan memahami kompleksitas waktu, Anda dapat memilih algoritma yang paling efisien sesuai dengan kebutuhan dan ukuran data yang diproses.

Melalui praktikum ini, Anda dapat melihat pentingnya pemahaman struktur data dan algoritma pencarian dalam pengembangan perangkat lunak. Dengan menggunakan algoritma pencarian yang tepat, Anda dapat meningkatkan kinerja dan efisiensi program, serta mengoptimalkan penggunaan sumber daya komputasi.

Tanda Tangan

A handwritten signature in black ink, consisting of stylized cursive letters, is written over a white background within a rectangular box.