PERTEMUAN 9

TEKNIK MERGE SORT

A. TUJUAN PEMBELAJARAN

Setelah mempelajari materi pertemuan ini diharapkan Mahasiswa mengerti dan paham teknik merge sort, logika dan cara kerja merge sort di python dan mempu membuat

B. URAIAN MATERI

1. Merge Sort

Merge Sort adalah algoritma *Divide and Conquer.* Ini membagi array input dalam dua bagian, menyebut dirinya untuk dua bagian dan kemudian menggabungkan dua bagian yang diurutkan. Fungsi merge() digunakan untuk menggabungkan dua bagian. Penggabungan (arr, l, m, r) adalah proses kunci yang mengasumsikan bahwa arr[l.. m] dan arr[m+1..r] diurutkan dan menggabungkan dua sub-array yang diurutkan menjadi satu. Algoritma penyortiran gabungan digunakan untuk mengurutkan data yang ada dalam urutan naik atau turun. Mari lihat bagaimana dapat menggunakan algoritma dan menerapkannya di Python.

a. Merge sort adalah teknik pemilahan tujuan umum murni berdasarkan pendekatan divide and conquer. Dalam teknik divide and conquer, elemen dibagi menjadi bagian atau daftar yang lebih kecil. Kemudian fungsi yang tepat diterapkan pada setiap setengah dari daftar input utama. Selanjutnya, bagian digabungkan bersama untuk mendapatkan hasilnya.

b. Merge Sortis adalah elemen yang tidak disortir dibagi menjadi dua bagian / bagian dan fungsi memanggil dirinya untuk bagian yang berpisah sedemikian rupa sehingga bagian-bagiannya terus membelah diri menjadi dua bagian sampai seluruh array diurutkan.recursive technique Ini secara rekursif menyebut dirinya untuk bagian atau sub-daftar sampai mendapat semua elemen yang terpisah dan bahwa tidak ada pembagian lebih lanjut yang mungkin yaitu setiap sub-daftar berisi 1 (tunggal) elemen.

c. Kemudian, elemen diurutkan menggunakan teknik dasar perbandingan dan swap. Akhirnya, ia menggabungkan semua elemen bersama-sama untuk mendapatkan daftar item data yang diurutkan akhir.

Teknik pemisahkan dan taklukkan, seperti yang mungkin dikatakan orang Romawi kuno, adalah pertempuran yang efektif strategi. Ternyata konsep ini sangat penting saat menulis algoritma. Itu Algoritma Merge Sort merupakan salah satu contoh algoritma divide and conquer. Bagilah dan algoritme taklukkan biasanya ditulis secara rekursif, tetapi tidak harus selalu demikian.

Premis dasarnya adalah kita membagi masalah menjadi dua bagian. Masing-masing dari dua bagian lebih mudah untuk dipecahkan daripada mencoba untuk menangani seluruh masalah sekaligus karena keduanya potongan masing-masing lebih kecil. Algoritme pengurutan gabungan membawa strategi pembagian dan penaklukan ini ke posisi yang ekstrem. Saya membagi listnya, lalu membaginya lagi dan lagi, sampai kita mendapatkan list ukuran 1. Sublist dengan panjang 1 sudah diurutkan. Dua sublist yang diurutkan dapat digabungkan menjadi satu list diurutkan dalam waktu O (n). List dapat dibagi menjadi list ukuran 1 dengan memisahkan berulang kali dalam waktu O (log n). Masing-masing list terpisah kemudian digabungkan bersama dalam waktu O (n). Ini menghasilkan kompleksitas O (n log n) untuk jenis gabungan.

Fungsi penggabungan menangani penggabungan dua sublist yang berdekatan. Sublist pertama berlangsung dari awal hingga pertengahan 1. Sublist kedua dimulai dari pertengahan hingga perhentian-1. Elemen-elemen dari dua sublist yang diurutkan disalin, dalam waktu O (n), ke list baru. Kemudian list yang diurutkan adalah disalin kembali ke urutan aslinya, lagi dalam waktu O (n). Dalam fungsi penggabungan, file first while loop menangani penggabungan dua sublist hingga satu atau sublist lainnya kosong. Sementara loop kedua dan ketiga menangani penyelesaian sublist mana saja memiliki elemen sisa. Hanya satu sublist akan memiliki elemen yang tersisa jadi hanya satu kondisi pada loop kedua dan ketiga akan pernah benar. Perhatikan bahwa loop sementara ketiga dalam kode diberi komentar. Menyalin elemen dari j ke stop di perulangan while ketiga tidak diperlukan karena mereka hanya akan disalin kembali ke tempat yang sama ketika konten lst disalin kembali ke sequrutan. Pengoptimalan ini mempercepat sedikit penggabungan. Satu

optimasi lainnya adalah untuk mengalokasikan satu list lagi untuk menyalin nilai dan kemudian bergantian di antaranya menggabungkan aslinya dan salinan yang dialokasikan sebelumnya. Dengan cara ini biaya overhead membuat dan menambahkan list dihindari. Mengkodekan salah satu dari dua pengoptimalan ini tidak meningkatkan kompleksitas komputasi algoritme, tetapi dapat meningkatkan kinerja keseluruhannya sedikit. Salah satu kritik dari algoritma merge sort adalah itu elemen dari dua sublist tidak dapat digabungkan tanpa menyalin ke list baru dan lalu kembali lagi. Metode pengurutan lainnya, seperti Quicksort, memiliki O yang sama (n log n) kompleksitas sebagai jenis gabungan dan tidak memerlukan list tambahan.

Algoritma pengurutan data mergesort dilakukan dengan menggunakan cara divideandconquer yaitu dengan memecah kemudian menyelesaikan setiap bagian kemudian menggabungkannya kembali. Pertama data dipecah menjadi 2 bagian dimana bagian pertama merupakan setengah (jika data genap) atau setengah minus satu (jika data ganjil) dari seluruh data, kemudian dilakukan pemecahan kembali untuk masing-masing blok sampai hanya terdiri dari satu data tiap blok. Setelah itu digabungkan kembali dengan membandingkan pada blok yang sama apakah data pertama lebih besar daripada data ke-tengah+1, jika ya maka data ke-tengah+1 dipindah sebagai data pertama, kemudian data ke-pertama sampai ke-tengah digeser menjadi data ke-dua sampai ke-tengah+1, demikian seterusnya sampai menjadi satu blok utuh seperti awalnya. Sehingga metode mergesort merupakan metode yang membutuhkan fungsi rekursi untuk penyelesaiannya. Dengan hal ini deskripsi dari algoritma dirumuskan dalam 3 langkah berpola divide-and-conquer. Berikut menjelaskan langkah kerja dari Mergesort.

a. Divide Memilah elemen – elemen dari rangkaian data menjadi dua bagian.

b. Conquer Conquer setiap bagian dengan memanggil prosedur mergesortsecararekursif Kombinasi

Mengkombinasikan dua bagian tersebut secara rekursif untuk mendapatkanrangkaian data berurutan. Proses rekursi berhenti jika mencapai elemen dasar. Hal ini terjadi bilamana bagian yang akan diurutkan menyisakan tepat satu elemen. Sisa pengurutan satu elemen tersebut menandakan bahwa bagian tersebut telah terurut sesuai rangkaian. Contoh penerapan atas sebuah

larik/array sebagai data sumber yang akan diurutkan {3, 9, 4, 1, 5, 2} adalah sebagai berikut:

a. pertama kali larik tersebut dibagi menjadi dua bagian, {3, 9, 4} dan {1, 5, 2}
b. Kedua larik kemudian diurutkan secara terpisah sehingga menjadi {3, 4, 9} dan {1, 2, 5}
c. Sebuah larik baru dibentuk yang sebagai penggabungan dari kedua larik tersebut {1}, sementara nilai-nilai dalam masing larik {3, 4, 9} dan {2, 5} (nilai 1 dalam elemen larik ke dua telah dipindahkan ke larik baru)
d. langkah berikutnya adalah penggabungan dari masing-masing larik ke dalam larik baru yang dibuat sebelumnya
e. {1, 2} ↔{3, 4, 9} dan {5}
f. {1, 2, 3} ↔ {4, 9} dan {5}
g. {1, 2, 3, 4}↔{9} dan {5}
h. {1, 2, 3, 4, 5}↔{9} dan {null}
i. {1, 2, 3, 4, 5, 9}↔{null}dan {null}

Dalam python, menggabungkan jenis didefinisikan sebagai salah satu algoritma penyortiran yang tujuan umum, menggunakan pembadatan berbasis perbandingan dengan membagi dan menaklukkan algoritma di mana idenya adalah untuk memecah daftar menjadi sub-daftar sampai setiap sub-daftar memiliki maksimal satu elemen dan menggabungkan semua sub-daftar dalam urutan terbalik untuk mendapatkan sub-daftar yang diurutkan dan akhirnya satu daftar yang diurutkan disebut jenis gabungan. Ini adalah algoritma penyortiran yang efisien, tujuan umum, dan terbaik dengan kompleksitas waktu keseluruhan, rata-rata, dan terburuk adalah O (nlogn).

Langkah-langkah berikut diikuti dengan cara rekursif untuk melakukan Merge Sort dan memanfaatkan hasil yang sesuai:

a. Temukan elemen tengah yang diperlukan untuk membagi array asli menjadi dua bagian.
b. Bagi daftar asli menjadi dua bagian dengan cara rekursif, sampai setiap sub-daftar berisi satu elemen. Yaitu memanggil fungsi merge_sort() untuk setiap setengah rekursif.

c. Periksa nilai data, jika ditemukan dalam urutan yang tidak disortir, tukar elemen dan gabungkan sub-daftar untuk mendapatkan daftar yang diurutkan asli.

2. Logika Merge Sort di Python

Dalam jenis gabungan, akan diberi daftar angka 'n' yang tidak diurutkan yang dibagi menjadi sub-daftar sampai setiap sub-daftar hanya memiliki satu elemen. Jadi akan membagi daftar menjadi n sub-daftar memiliki satu elemen, yang diurutkan dengan sendirinya. Sekarang akan menggabungkan semua sub-daftar untuk mendapatkan sub-daftar yang diurutkan dan akhirnya sub-daftar yang diurutkan. Logika membagi dan Menaklukkan dalam merge sort

a. Membagi masalah menjadi beberapa subproblems.

b. Memecahkan sub-masalah dengan lebih membagi sub-masalah menjadi masalah atom di mana mereka memiliki solusi yang tepat.

c. Kemudian menggabungkan semua sub solusi untuk mendapatkan solusi akhir untuk masalah yang diberikan.

Di sini masalahnya adalah mengurutkan daftar yang diberikan sebagai penyortiran gabungan diikuti sebagai metode terbagi dan menaklukkan; dimana akan membagi daftar yang diberikan ke tengah daftar. Mertimbangkan, mengingat daftar memiliki lima elemen, yang sekarang akan membagi daftar ke pertengahan dengan menghitung pertengahan (start + end / 2), di sini adalah 2,5, dan akan mempertimbangkan integer bagian 2, sehingga satu sub-daftar memiliki 2 elemen dan sub-daftar lain memiliki 3 elemen, tetapi 2 sub-daftar tidak cukup untuk memecahkan masalah. Dimana akan membagi sub-masalah lebih lanjut sampai memiliki satu elemen di setiap sub-daftar sebagai masalah atom, dan akhirnya, kami akan menggabungkan semua sub-daftar untuk membentuk sub-daftar yang diurutkan dan akhirnya daftar yang diurutkan.Pertimbangkan daftar dengan elemen 12 5 9 2 10; sekarang, akan memiliki langkah demi langkah membagi dan menggabungkan sebagai berikut:

Langkah 1: 12 5 9 2 10 (pertengahan = 5/2 = 2)

Langkah 2: 12 5 9 | 2 10

Langkah 3: 12 5 9 | 2 10

Langkah 4: 12 5 9 | 2 10

Semua langkah diatas, lakukan cara membagi masalah menjadi subproblems sampai setiap subproblem hanya memiliki satu elemen, seperti pada langkah 4.Sekarang akan menggabungkan semua subproblems dengan solusi untuk mendapatkan solusi akhir.

Langkah 5: 5 12 9 | 2 10

Langkah 6: 5 9 12 | 2 10

Langkah 7: 2 5 9 10 12

Dari semua langkah di atas, orang dapat mengetahui bahwa pada setiap langkah, kita membagi daftar ukuran N ke N / 2 sub-daftar sampai kita tidak dapat membagi lebih lanjut. Dalam implementasi urutkan gabungan, mengikuti langkah-langkah di bawah ini:

a. Mengambil dua variabel, "mulai" dan "akhir", di mana "mulai" akan menyimpan indeks awal, yaitu 0 di dalamnya, dan akhir akan menyimpan panjang daftar, yaitu 5 dalam contoh di atas.
b. Dengan menggunakan variabel awal dan akhir, akan menemukan bagian tengah daftar dengan menggunakan rumus sebagai (start + end) / 2 dan menyimpan hasilnya di pertengahan variabel, dan kita akan membagi daftar menjadi 2 sub-daftar sebagai awal hingga pertengahan sebagai satu sub-daftar dan pertengahan + 1 untuk berakhir sebagai sub-daftar lain.
c. Membagi sub-daftar menjadi sub-daftar lebih lanjut sebagai masalah atom dengan mengikuti proses di langkah 2.
d. Akhirnya, kami akan menggabungkan semua sub-daftar dengan solusi untuk mendapatkan daftar akhir yang akan diurutkan.

Jenis penggabungan dalam python. Sejauh ini, definisi jenis penggabungan, apa logika di balik implementasi jenis gabungan dan penjelasan logika, berbagai jenis implementasi jenis penggabungan dalam python dengan contoh dan output yang sesuai. Di sini akan memiliki pemahaman yang jelas dan baik tentang jenis penggabungan dan dengan mudah memecahkan masalah jenis penggabungan.

3. Cara Kerja Merge Sort di Python

Pada penggambaran cara kerja ini dengan bantuan contoh daftar elemen: 11, 31, 7, 41, 101, 56, 77, 2

a. Urutkan Gabungkan dalam Python

Seperti disebutkan di atas, pada awalnya membagi daftar asli elemen data dalam dua bagian. Karena array asli di atas berisi 8 elemen, membagi array menjadi sub-array dari 4 elemen. Array terus rekursif membagi dirinya menjadi sub-daftar, sampai satu elemen diperoleh per sub-daftar yaitu tidak ada lagi divisi lebih lanjut yang mungkin.

b. Urutkan Gabungkan dalam Pemisahan Python Elemen Data. Seperti yang dinyatakan dengan jelas, daftar tersebut secara rekursif dibagi menjadi dua bagian / bagian sampai semua elemen dipisahkan sebagai individu. Setelah pemisahan elemen, akan melihat elemen individu sebagai berikut:

c. Urutkan Gabungkan dalam Hasil Python Setelah Pemisahan Elemen Rekursif

d. Setelah elemen dipisahkan, kita perlu menggabungkan elemen data dengan cara yang sama seperti kita telah membagi elemen.

e. Pertimbangkan elemen 11 dan 31. Karena berada dalam posisi yang diurutkan, kami menggabungkannya dan menggabungkannya bersama dalam array. Elemen 7 dan 41 juga muncul di tempat yang diurutkan, jadi kami menggabungkannya juga.

f. Sekarang, jika melihat elemen 101 dan 56, perlu menukar posisi mereka dan menggabungkannya bersama. Dengan cara yang sama, elemen 77 dan 2 ditukar sehubungan dengan posisi mereka dan digabungkan bersama.

g. Penggabungan dan Pemilahan Iterasi 1

Membawanya ke depan, dalam iterasi kedua, kami membandingkan sub-array dari dua elemen dengan sub-array lainnya dan jika elemen ditemukan diurutkan, kami menggabungkan sub-array sama sekali.

Sub-array [11,31] dibandingkan dengan [7,41] dan
subarray [56.101] dibandingkan dengan [2,77].

Karena item data tidak dalam urutan yang diurutkan, posisi mereka ditukar.

h. Penggabungan dan Pengurutan Iterasi 2

Dalam iterasi ketiga, sub-array dari 4 elemen dibandingkan dengan sub-array lainnya yaitu [7, 11, 31, 41] dibandingkan dengan [2, 56, 77, 101]. Seperti yang terlihat jelas, elemen tidak berada dalam posisi yang diurutkan, sehingga elemen ditukar dan digabungkan untuk mendapatkan array yang diurutkan akhir.

i. Penggabungan dan Pengurutan Iterasi 3

4. Implementasi Merge Sort di Python

Implementasi Mergesort terlihat pada source code dibawah ini.

```
1 def merge_sort(inp_arr):
2     size = len(inp_arr)
3     if size > 1:
4         middle = size // 2
5         left_arr = inp_arr[:middle]
6         right_arr = inp_arr[middle:]
7         merge_sort(left_arr)
8         merge_sort(right_arr)
9         p = 0
10        q = 0
11        r = 0
12        left_size = len(left_arr)
13        right_size = len(right_arr)
14        while p < left_size and q < right_size:
15            if left_arr[p] < right_arr[q]:
16              inp_arr[r] = left_arr[p]
17              p += 1
18            else:
19                inp_arr[r] = right_arr[q]
20                q += 1
21            r += 1
22        while p < left_size:
23            inp_arr[r] = left_arr[p]
24            p += 1
25            r += 1
26        while q < right_size:
27            inp_arr[r]=right_arr[q]
28            q += 1
29            r += 1
30 inp_arr = [12, 33, 17, 40, 100, 50, 7, 24]
31 print("Input Array:\n")
32 print(inp_arr)
33 merge_sort(inp_arr)
34 print("Sorted Array:\n")
35 print(inp_arr)
36
```

Jika di run hasilnya sebagai berikut.

[12, 33, 17, 40, 100, 50, 7, 24]

Sorted Array:

[7, 12, 17, 24, 33, 40, 50, 100]

Contoh 2 implementasi pada python

```python
# Python program for implementation of MergeSort
# Merges two subarrays of arr[].
# First subarray is arr[l..m]
# Second subarray is arr[m+1..r]

def merge(arr, l, m, r):
    n1 = m - l + 1
    n2 = r - m
    # create temp arrays
    L = [0] * (n1)
    R = [0] * (n2)

    # Copy data to temp arrays L[] and R[]
    for i in range(0, n1):
        L[i] = arr[l + i]

    for j in range(0, n2):
        R[j] = arr[m + 1 + j]

    # Merge the temp arrays back into arr[l..r]
    i = 0     # Initial index of first subarray
    j = 0     # Initial index of second subarray
    k = l     # Initial index of merged subarray

    while i < n1 and j < n2:
        if L[i] <= R[j]:
            arr[k] = L[i]
            i += 1
        else:
            arr[k] = R[j]
            j += 1
        k += 1

    # Copy the remaining elements of L[], if there
    # are any
    while i < n1:
        arr[k] = L[i]
        i += 1
        k += 1

```

```
    # Copy the remaining elements of R[], if there
    # are any
    while j < n2:
        arr[k] = R[j]
        j += 1
        k += 1

# l is for left index and r is right index of the
# sub-array of arr to be sorted

def mergeSort(arr, l, r):
    if l < r:

        # Same as (l+r)//2, but avoids overflow for
        # large l and h
        m = l+(r-l)//2

        # Sort first and second halves
        mergeSort(arr, l, m)
        mergeSort(arr, m+1, r)
        merge(arr, l, m, r)

# Driver code to test above
arr = [121, 10, 18, 51, 6, 73]
n = len(arr)
print("Given array is")
for i in range(n):
    print("%d" % arr[i]),

mergeSort(arr, 0, n-1)
print("\n\nSorted array is")
for i in range(n):
    print("%d" % arr[i]),
```

Hasilnya berikut ini.

Given array is

121 10 18 51 6 73

Sorted array is

6 10 18 51 73 121

Contoh 2 implementasi Mergesort dengan bahasa Python sebagai berikut.

```
def mergeSort(alist):
    print("Splitting ",alist)
    if len(alist)>1:
        mid = len(alist)//2
        lefthalf = alist[:mid]
        righthalf = alist[mid:]
        mergeSort(lefthalf)
        mergeSort(righthalf)

        i=0
        j=0
        k=0
        while i<len(lefthalf) and j<len(righthalf):
            if lefthalf[i]<righthalf[j]:
                alist[k]=lefthalf[i]
                i=i+1
            else:
                alist[k]=righthalf[j]
                j=j+1
            k=k+1
        while i<len(lefthalf):
            alist[k]=lefthalf[i]
            i=i+1
            k=k+1
        while j<len(righthalf):
            alist[k]=righthalf[j]
            j=j+1
            k=k+1
    print("Merging ",alist)
alist = [54,26,93,17,77,31,44,55,20]
mergeSort(alist)
print(alist)
```

```
('Splitting ', [93])
('Merging ', [93])
('Splitting ', [17])
('Merging ', [17])
('Merging ', [17, 93])
('Merging ', [17, 26, 54, 93])
('Splitting ', [77, 31, 44, 55, 20])
('Splitting ', [77, 31])
('Splitting ', [77])
('Merging ', [77])
('Splitting ', [31])
('Merging ', [31])
('Merging ', [31, 77])
('Splitting ', [44, 55, 20])
('Splitting ', [44])
('Merging ', [44])
('Splitting ', [55, 20])
('Splitting ', [55])
('Merging ', [55])
('Splitting ', [20])
('Merging ', [20])
('Merging ', [20, 55])
('Merging ', [20, 44, 55])
('Merging ', [20, 31, 44, 55, 77])
('Merging ', [17, 20, 26, 31, 44, 54, 55, 77, 93])
[17, 20, 26, 31, 44, 54, 55, 77, 93]
```

Jadi, dalam artikel ini, kami telah memahami cara kerja jenis Merge di Python. Lihatlah algoritma penyortiran lainnya di Python.

C. LATIHAN SOAL

1. Jelaskan yang anda ketahui teknik merge Sort!
2. Buatlah analogi deret bilangan dengan Merge sort dalam 6 angka yang akan dilakukan pengurutan!
3. Jelaskan cara kerja Merge sort!
4. Jelaskan langkah dengan cara rekursif untuk melakukan Merge Sort!
5. Buatlah contoh implementasi teknik merge sort dengan bahasa python!

D. REFERENSI

Emi Sita Eriana, A. Z. (2021). *Praktikum Algoritma dan Pemrograman.* Tangerang Selatan: Unpam Press.

Mohamad Aslam Katahman, M. F. (2021). Pembangunan Aplikasi Realiti Terimbuh Untuk Pengenalan Struktur Data. Information Technology And Computer Science.

Nasrullah, A. H. (2021). Implementasi Algoritma Decision Tree Untuk Klasifikasi Produk Laris. Jurnal Ilmiah Ilmu Komputer I.

Peng Qi, Y. Z. (2020). Stanza: A Python Natural Language Processing Toolkit For Many Human Languages.

Pradana Setialana, T. B. (2017). Pencarian Hubungan Kekerabatan Pada Struktur Data Genealogy Dalam Graph Databas.

Ranny Meilisa, D. P. (2020). Model Pembelajaran Flipped Classroom Pada Mata Kuliah Algoritma Dan Struktur Data. Jurnal Ilmiah Pendidikan Dan Pembelajaran (Jipp).

Revanza, M. G. (2020). Struktur Data Dan Bahasa Pemrograman.

Risah Subariah, E. S. (T.Thn.). Praktikum Analisis & Perancangan Sistem (Uml).

Sianipar, R. H. (2013). Pemrograman & Struktur Data C: Belajar Dari Contoh Untuk Programmer Pemula Maupun Berpengalaman. Penerbit Informatika, 2013.

Thanaki, J. (2017). Python Natural Language Processing. Mambai.

Zein, A. (2018). Pendeteksian Kantuk Secara Real Time Menggunakan Pustaka Opencv Dan Dlib Python. Sainstech : Jurnal Penelitian Dan Pengkajian Sains Dan Teknologi.