

ALGORITMA DAN STRUKTUR DATA

Penyusun:

Afrizal Zein

Emi Sita Eriana



Jl. Surya Kencana No. 1 Pamulang
Gd. A, Ruang 212 Universitas Pamulang
Tangerang Selatan – Banten

ALGORITMA DAN STRUKTUR DATA

Penulis:

Afrizal Zein

Emi Sita Eriana

ISBN: 978-623-6352-65-6

Editor:

Emi Sita Eriana

Penyunting:

Emi Sita Eriana

Desain Sampul:

Aden

Tata Letak:

Ramdani Putra

Penerbit:

Unpam Press

Redaksi:

Jl. Surya Kencana No. 1

R. 212, Gd. A Universitas Pamulang Pamulang | Tangerang Selatan | Banten

Tlp/Fax: 021. 741 2566 – 7470 9855 Ext: 1073

Email: unpampress@unpam.ac.id

Cetakan pertama, 14 Januari 2022

Hak cipta dilindungi undang-undang.

Dilarang memperbanyak karya tulis ini dalam bentuk dan dengan cara apapun tanpa izin penerbit.

DATA PUBLIKASI UNPAM PRESS
| Lembaga Penerbit dan Publikasi Universitas Pamulang

Gedung A. R. 212 Kampus 1 Universitas Pamulang
Jalan Surya Kencana Nomor 1 Pamulang Barat, Tangerang Selatan, Banten
Website: www.unpam.ac.id | Email: unpampress@unpam.ac.id

Algoritma dan Struktur Data / Afrizal Zein dan Emi Sita Eriana -1STed

ISBN. 978-623-6352-65-6

1. Algoritma dan Struktur Data I. Afrizal Zein II. Emi Sita Eriana

M196-14012022-01

Ketua Unpam Press: Pranoto

Koordinator Editorial: Aden

Koordinator Hak Cipta: Susanto

Koordinator Produksi: Dameis Surya Anggara

Koordinator Publikasi: Kusworo, Heri Haerudin

Koordinator Dokumentasi: Ramdani Putra. Nara Dwi Angesti

Desain Cover: Putut Said Permana

Cetakan pertama, 14 Januari 2022

Hak cipta dilindungi undang-undang.

Dilarang memperbanyak karya tulis ini dalam bentuk dan dengan cara apapun tanpa ijin penerbit.

MATA KULIAH

ALGORITMA DAN STRUKTUR DATA

IDENTITAS MATA KULIAH

Program Studi	: Sistem Informasi S-1
Mata Kuliah / Kode	: Struktur Data/ KB1101
Sks	: 3 Sks
Prasyarat	:
Deskripsi Mata Kuliah	: Mata kuliah ini membahas mengenai Algoritma dan Struktur Data. Mata kuliah ini merupakan mata kuliah wajib Program Studi S-1 sistem Informasi yang membahas tentang Pengantar Algoritma dan struktur data, tipe data, array, linked List, teknik searching , teknik Sorting, stack, queue, binary tree, graph dan algoritma djiktra.
Capaian Pembelajaran	: Setelah pembelajaran ini, mahasiswa mampu mampu membuat program dengan menggunakan bahasa python dengan benar
Ketua Program Studi Sistem Informasi S-1	Ketua Tim Penyusun
Dede Supriadi S.Kom, M.Kom NIDN 0442760018	Drs. Afrizal Zein, M.Kom. NIDN 0313076504

KATA PENGANTAR

Dengan menucap puji kesyukuran kehadirat Allah SWT, dimana memberikan kelimpahan rahmat dan hidayahNya, sampai saat syukur kami panjatkan atas kelancaran dan kemudahan untuk menyusun bahan ajar Algoritma dan Struktur Data. Harapan kami dengan modul ajar ini dapat memotivasi dan menjadi acuan belajar mahasiswa Fakultas Teknik Program Studi Sistem Informasi.

Bahan ajar ini disusun berdasar pengalaman mengajar dari tahun 2014 beserta beberapa referensi dari buku, jurnal, serta karya ilmiah yang mendukung materi pada setiap bab dalam 18 pertemuan. Modul ini berisi teori, uraian soal, penjelasan mendetail sehingga memberikan kemudahan untuk mahasiswa pelajari mandiri. Selain itu contoh pengaplikasian dalam praktek Algoritma dan Struktur Data kedalam bahasa Python yang memiliki bahasa sederhana, struktur yang mampu menggambarkan alur pemograman yang baik. Besar Harapan kami dengan contoh yang diberikan secara simpel pengguna akan dapat diterima, dipahami dan dipengerti oleh pembaca, khususnya mahasiswa Prodi SI maupun khalayak Umum.

Permintaan maaf kami sampaikan apabila terdapat kesalahan dalam tulisan yang kurang berkenan atas penyampaian materi atau kalimat dalam modul ini, harapan kami kritik dan saran yang membangun dari para pembaca untuk kemajuan modul ini dan memperbaiki kekurangan kedepannya.

Tangerang Selatan, Desember 2021

Penyusun

DAFTAR ISI

ALGORITMA DAN STRUKTUR DATA	ii
DATA PUBLIKASI UNPAM PRESS	iii
IDENTITAS MATA KULIAH	iv
KATA PENGANTAR	iv
DAFTAR ISI.....	vi
DAFTAR GAMBAR	xiii
DAFTAR TABEL	xv
PERTEMUAN 1	1
PENGANTAR ALGORITMA DAN STRUKTUR DATA.....	1
A. TUJUAN PEMBELAJARAN.....	1
B. URAIAN MATERI	1
1. Sejarah Algoritma	1
2. Definisi, Implementasi dan Struktur Algortima.....	3
3. Sejarah Pemograman.....	5
4. Definisi Pemrograman	7
5. Jenis Pemrograman Komputer.....	8
6. Konsep Struktur Data.....	13
C. LATIHAN SOAL.....	16
D. REFERENSI.....	16
PERTEMUAN 2	18
PENGENALAN PEMROGRAMAN PYTHON	18
A. TUJUAN PEMBELAJARAN.....	18
B. URAIAN MATERI	18
1. Pemrograman Python	18
2. Baris Perintah.....	20
C. LATIHAN SOAL.....	34
D. REFERENSI.....	34
PERTEMUAN 3	35
TIPE DATA	35
A. TUJUAN PEMBELAJARAN.....	35
B. URAIAN MATERI	35
1. Pengertian Tipe Data	35

2. Tipe data abstrak (ADT)	36
3. Implementasi ADT	37
4. Jenis Struktur Data	43
5. Memilih Struktur Data	45
C. LATIHAN SOAL.....	47
D. REFERENSI.....	47
PERTEMUAN 4	49
IF-ELSE	49
A. TUJUAN PEMBELAJARAN.....	49
B. URAIAN MATERI	49
1. Pernyataan IF.....	49
2. Pernyataan Else	52
3. Pernyataan Elif.....	55
4. Pernyataan If Bersarang	57
C. LATIHAN SOAL.....	63
D. REFERENSI.....	63
PERTEMUAN 5	65
ARRAY DAN LINKED LIST.....	65
A. TUJUAN PEMBELAJARAN.....	65
B. URAIAN MATERI	65
1. Array	65
2. Pengakses Elemen Array.....	66
3. Struktur Array	68
4. Implementasi Array	71
5. Tipe Data Abstrak Array	73
6. Linked List.....	75
7. Node dalam Linked List.....	77
8. Implementasi Linked List di Python	83
C. LATIHAN SOAL.....	84
D. REFERENSI.....	84
PERTEMUAN 6	86
FUNGSI REKURSIF	86
A. TUJUAN PEMBELAJARAN.....	86
B. URAIAN MATERI	86
1. Definisi Fungsi.....	86

2. Memanggil Function/Fungsi.....	88
3. Melewati Referensi Vs Nilai	89
4. Rekursif	95
5. Anonymous/Lambda Function.....	99
6. Fungsi dalam Python	100
C. LATIHAN SOAL.....	101
D. REFERENSI.....	101
PERTEMUAN 7	103
TEKNIK SEARCH.....	103
A. TUJUAN PEMBELAJARAN.....	103
B. URAIAN MATERI	103
1. Konsep Dasar Search.....	103
2. Teknik Pencarian.....	105
3. Cara Kerja Binary Search.....	107
4. Pencarian Binery di Python.....	112
C. LATIHAN SOAL.....	114
D. REFERENSI.....	114
PERTEMUAN 8	116
BUBBLE SORT.....	116
A. TUJUAN PEMBELAJARAN.....	116
B. URAIAN MATERI	116
1. Sorting	116
2. Bubble Sort.....	117
3. Representasi Visual	119
4. Keuntungan dan Kekurangan bubble sort.....	122
5. Analisis Kompleksitas Bubble Sort	122
6. Insertion Sort (Metode Penyisipan)	124
7. Visualisasi Insertion Sort.....	124
8. Implementasi Bubble Sort dan Insert Sort di Python	126
C. LATIHAN SOAL.....	129
D. REFERENSI.....	129
PERTEMUAN 9	131
TEKNIK MERGE SORT.....	131
A. TUJUAN PEMBELAJARAN.....	131
B. URAIAN MATERI	131

1. Merge Sort	131
2. Logika Merge Sort di Python	135
3. Cara Kerja Merge Sort di Python	137
4. Implementasi Merge Sort di Python	138
C. LATIHAN SOAL.....	142
D. REFERENSI.....	143
PERTEMUAN 10	144
TEKNIK SELECTION SORT	144
A. TUJUAN PEMBELAJARAN.....	144
B. URAIAN MATERI	144
1. Selection Sort.....	144
2. Representasi Visual	148
3. Kompleksitas Waktu Selection Sort.....	153
4. Implementasi Selection Sort di Python	154
C. LATIHAN SOAL.....	157
D. REFERENSI.....	158
PERTEMUAN 11	159
TEKNIK QUICK SORT	159
A. TUJUAN PEMBELAJARAN.....	159
B. URAIAN MATERI	159
1. Quick Sort	159
2. Logika Quick Sort	161
3. Teknik Divide and Conquer.....	166
4. Quicksort dengan Python.....	167
C. LATIHAN SOAL.....	173
D. REFERENSI.....	173
PERTEMUAN 12	175
STACK	175
A. TUJUAN PEMBELAJARAN.....	175
B. URAIAN MATERI	175
1. Stack.....	175
2. Implementasi Menggunakan List	182
3. Implementasi Collections.Deque.....	183
4. Dequelist	184
5. Implementasi Stack dalam Antrian	185

6. Implementasi Menggunakan Singly Linked List.....	187
7. Tumpukan Python dan Threading.....	187
8. Implementasi Python Stacks	188
C. LATIHAN SOAL.....	191
D. REFERENSI.....	191
PERTEMUAN 13	193
QUEUE.....	193
A. TUJUAN PEMBELAJARAN.....	193
B. URAIAN MATERI	193
1. Queue.....	193
2. Insert Button Antrian	194
3. Menambahkan Elemen.....	197
4. Implementasi Menggunakan Antrian.....	197
5. Cara Menggunakan Antrian di Python	199
6. Cara Kerja Queue	200
7. Menambahkan Item Antrian.....	201
8. Menghapus Item Antrian	202
9. Penerapan Antrian di Python.....	203
10. Implementasi Menggunakan List	204
11. Implementasi Menggunakan Collections.Deque.....	205
C. LATIHAN SOAL.....	208
D. REFERENSI.....	208
PERTEMUAN 14	210
ANTRIAN DENGAN ARRAY.....	210
A. TUJUAN PEMBELAJARAN.....	210
B. URAIAN MATERI	210
1. Antrian	210
2. Antrian dengan Array	213
3. Operasi Antrian Menggunakan Array	215
4. Implementasi Array Antrian.....	222
5. Implementasi Queue dengan Pointer.....	222
C. LATIHAN SOAL.....	223
D. REFERENSI.....	223
PERTEMUAN 15	225
BINARY TREE DAN SEARCH TREE	225

A. TUJUAN PEMBELAJARAN.....	225
B. URAIAN MATERI	225
1. Binary Tree	225
2. Binary Tree Search di Python.....	229
3. Pohon Pencarian Biner.....	231
4. Cara Menyisipkan Elemen Search Tree	232
5. Mencari Elemen di Search Tree.....	234
C. LATIHAN SOAL.....	236
D. REFERENSI.....	236
PERTEMUAN 16	238
TREE TRAVERSAL.....	238
A. TUJUAN PEMBELAJARAN.....	238
B. URAIAN MATERI	238
1. Definisi Tree Traversal.....	238
2. Algoritma Level Order Tree Traversal.....	240
3. Traversal Tree Inorder	242
4. Pre-order Traversal	245
5. Post-order Traversal	245
6. Kompleksitas Waktu.....	249
C. LATIHAN SOAL.....	250
D. REFERENSI.....	250
PERTEMUAN 17	252
GRAPH.....	252
A. TUJUAN PEMBELAJARAN.....	252
B. URAIAN MATERI	252
1. Sejarah Graph.....	252
2. Terminologi Graph.....	254
3. Manfaat Graph.....	258
4. Graph dan Analisis Jaringan.....	259
5. Graph dalam Python	262
6. Visualisasi Graph.....	265
C. LATIHAN SOAL.....	270
D. REFERENSI.....	271
PERTEMUAN 18	272
ALGORITMA DIJKSTRA	272

A. TUJUAN PEMBELAJARAN.....	272
B. URAIAN MATERI	272
1. Algoritma Dijkstra	272
2. Algoritma Dijkstra dalam Python.....	273
3. Greedy Approach	276
4. Implementasi Algoritma Dijkstra di Python	280
C. LATIHAN SOAL.....	284
D. REFERENSI.....	284
DAFTAR PUSTAKA.....	285
RENCANA PEMBELAJARAN SEMESTER	273
(RPS).....	273

DAFTAR GAMBAR

Gambar 2. 1 Gambar Alur pembuatan python	21
Gambar 2. 2 Installasi Python	23
Gambar 2. 2 Installasi Python	23
Gambar 3. 1 Implementasi Abstract Data Type.....	38
Gambar 3. 2 List ADT	38
Gambar 3. 3 List Fungsi ADT.....	39
Gambar 3. 4 Stack ADT	40
Gambar 3. 5 Queue ADT	42
Gambar 4. 1 Diagram Alur IF	51
Gambar 4. 2 Diagram Alur If-Else	53
Gambar 4. 3 Diagram Alur Elif	55
Gambar 4. 4 Diagram Alur If Bersarang	58
Gambar 5. 1 Array 2 Demensi	74
Gambar 10. 1 Representasi Visual Selection Sort	149
Gambar 10. 2 Parameter Variable Global	152
Gambar 11. 1 Ilustrasi Quick Sort	160
Gambar 12. 1 Ilustrasi Tumpukan Surat.....	176
Gambar 12. 2 Operasi Stack.....	177
Gambar 13. 1 Antrian Orang.....	193
Gambar 13. 2 Operasi Metode Kelas Antrian.....	195
Gambar 13. 3 Ilustrasi Queue dalam Struktur Data.....	196
Gambar 13. 4 FIFO dalam Antrian	197
Gambar 14. 1 Ilustrasi Queue	210
Gambar 14. 2 Ilustrasi Cara Pertama Queue	211

Gambar 14. 3 Ilustrasi penghapusan cara kedua.....	212
Gambar 14. 4 Ilustrasi Cara Ketiga Queue	213
Gambar 14. 5 Queue Linier Array	215
Gambar 17. 1 Sejarah Graph.....	253
Gambar 17. 2 Ilustrasi Graph.....	259
Gambar 17. 3 Grafh V.....	260
Gambar 17. 4 Ilustrasi Komponen Network.....	261
Gambar 17. 5 Graph Segitiga	265
Gambar 17. 6 Visualisasi Graph Titik	267
Gambar 17. 7 Graph Kota Scotlandia	268

DAFTAR TABEL

Tabel 5. 1 Typecode Array.....	72
Tabel 17. 1 Daftar Adjacency Kota Scotlandia	269
Tabel 17. 2 Matriks Adjacency Kota Scotlandia	270

PERTEMUAN 1

PENGANTAR ALGORITMA DAN STRUKTUR DATA

A. TUJUAN PEMBELAJARAN

Setelah mempelajari materi ini diharapkan mahasiswa mampu memahami sejarah, konsep Algoritma dan Struktur Data dan mengimplementasikan kedalam bahasa pemrograman

B. URAIAN MATERI

1. Sejarah Algoritma

Algoritma iaah jantung dari ilmu komputer. Komputasi banyak memiliki cabang disebutkan didalam terminologi algoritma sebagai contoh algoritma untuk merutekan pesan didalam jaringan komputer, algoritma Brezenhamdigunakan menelusuri infografis, algoritma KnuthMorrisPratt dapat menemukan pola pada komputer jaringan. dalam teks pencarian informasi dan sebagainya.

Dilihat sudut pandang etimologis, kata "algoritma" itu memiliki sejarah aneh. Kata algoritma tidak muncul di kamus Webster tahun 1957. Jika menggunakan angka Arab, maka termasuk sebagai ahli algoritma. Ahli bahasa mencoba mencari kata algoritma, tetapi hasilnya tidak memuaskan. Akhirnya, sejarawan matematika Kata algoritma berasal dari nama seorang penulis ternama yaitu Abu Ja'far Muhammad ibn Musa al-Khuwarizmi (al-Khuwarizmi diucapkan sebagai algoritma oleh orang Barat). Al-Khuwarizmi menulis buku berjudul Kitab al jabar wal-muqabala. Dari judul kata "aljabar". Perubahan dari kata algoritma ke algoritma terjadi karena kata algoritma sering salahkan dengan aritmatika, sehingga akhiran -sm menjadi -thm. Karena perhitungan angka arab telah biasa, kata algoritma secara bertahap menjadi metode perhitungan umum, sehingga hilanglah arti aslinya.

Pada bahasa Indonesia diserap menjadi "algoritma". Tahun 1950, muncul pertama dalam "Algoritme Euclid" (Algoritme Euclid). Matematikawan Yunani Euclid (lahir 350 M)menulis dalam bukunya "Elements" serangkaian langkah yang digunakan sebagai penemu dan pembagi persekutuan paling besar dari

dua bilangan bulat m dan n. (Tentu saja, Euclid tidak menyebut metodenya sebagai algoritma, sampai zaman modern orang menyebut metodenya "algoritma Euclidean"). Pembagian sekutuan paling besar dari 2 bilangan bulat non-negatif ialah bilangan bulat positif paling besar dimana membagi dua bilangan tersebut, misalnya, m=60 dan n=12 Semua faktor pembagi 60 adalah

1,2,3,4,5,6,10,12,15,20,30,60

dan semua faktor pembagi 16 adalah

1,2,4,8,16

maka $\text{gcd}(60,16) = 4$. Langkah mencari $\text{gcd}(60,16)$ dengan algoritma Euclidean sebagai berikut:

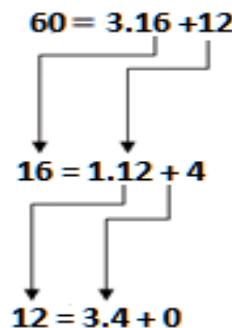
$$60 \text{ dibagi } 16 \text{ hasilnya} = 3 \text{ sisa} = 12 \quad (\text{atau: } 60 = 3 \cdot 16 + 12)$$

$$16 \text{ dibagi } 12 \text{ hasilnya} = 1, \text{sisa} = 4 \quad (\text{atau: } 16 = 1 \cdot 12 + 4)$$

$$12 \text{ dibagi } 4 \text{ hasilnya} = 3, \text{sisa} = 0 \quad (\text{atau: } 12 = 3 \cdot 4 + 0)$$

oleh karena hasil akhirnya 0, maka pembagian sebelumnya adalah 4, maka $\text{gcd}(60,16)$ jadi $\text{gcd}(60,16)$. Jadi, $\text{gcd}(60,16) = \text{gcd}(16,12) = \text{gcd}(4,0) = 4$.

Proses mencari gcd dari 60 dan 16 juga dapat digambarkan berikut:



Algoritma Euclidean memiliki banyak versi, diantaranya sebagai berikut. Terdapat 2 bilangan bulat tak negatif yaitu x dan y ($x \geq y$). Algoritma Euclidean akan cari nilai bagi sama yang paling besar, gcd, dari dua bilangan tersebut, yakni bilangan bulat positif paling besar yang habis membagi x dan y.

- Jika $y = 0$ maka x ialah jawaban, berhenti tapi apabila $n \neq 0$, selanjutnya langkah ke-2
- Bagi x dengan y dan dimisalkan z adalah sisa

- c. Kemudian mengganti nilai x dengan nilai y dan dengan nilai z , lalu kembali ke langkah pertama.

Melalui algoritma Euclidean, dapat menghitung gcd yang berasal dari 2 bilangan bulat sembarang secara sistematis.

2. Definisi, Implementasi dan Struktur Algoritma

Algoritma banyak digunakan dalam bidang ilmu komputer. Dalam membuat program komputer, algoritma ialah suatu konsep yang mendasar digunakan membuat tugas pemrograman yang efisien, efektif, dan terstruktur. Pada pertemuan awal ini membahas tentang definisi dari algoritma. Mendefinisikan algoritma ialah proses atau sekumpulan aturan pada proses menghitung untuk memecahkan suatu permasalahan. Dengan makna lain algoritma ialah sekumpulan metode atau urutan program yang sistematis, logis dan terstruktur guna memecahkan masalah secara efisien. Dapat dikatakan bahwa algoritma ialah proses rinci yang dapat menyelesaikan suatu masalah melalui rangkaian langkah berurutan, teratur dan memberikan solusi yang sesuai.



Gambar 1. 1 Implementasi Algoritma dalam Teknologi

Penerapan algoritma biasanya digunakan para pengajar, sarjana atau awam. Kadang-kadang seseorang tidak mengetahui bahwa dia telah menganalisis dan mengeksekusi teknik algoritmik. Bagi para mahasiswa atau siswa SMK di bidang teknik informatika komputer atau jurusan sejenis, algoritma merupakan mata kuliah wajib kejuruan dalam mata kuliahnya, bahkan ada pepatah yang mengatakan bahwa algoritma merupakan inti dari

informatika. Ilmu komputasi termasuk dalam bidang algoritma dan pemrograman. Struktur Algoritma disusun dari bagian Head dari file header, yang berisi bagian frame dari file header dan tahap akhir dari deskripsi. uraian masalah yang menentukan tahap binding ke variabel atau objek binding dari tipe data Tahap rinci pemecahan masalah, yang berisi langkah-langkah sistem logis untuk memecahkan masalah untuk menghasilkan solusi atau output dari masalah yang dihadapi perumpamaan ringan algoritma dalam sehari-hari dimana seorang ingin membuat kopi

- a. Header: Bagian ini mengenai statemen membuat kopi
- b. Deklarasi: Bagian menjelaskan bahan kopi seperti gula, bubuk kopi, air, kompor, panci, sendok dan gelas.
- c. Deskripsi permasalah: Variabel di bagian deklarasi diproses. Letakkan air dalam panci yang dimasak diatas kompor, tunggu mendidih kemudian tuang bubuk kopi dan gula dalam gelas lalu tiangkan air mendidih dan aduk sampai gula menyatu dengan air

Suatu ciri atau Karakteristik algoritma sebagai berikut.

- a. Algoritme punya awal dan akhir, dan stop apabila selesai
- b. Algoritma punya keadaan input
- c. Algoritma tidak memiliki makna double atau ambigu
- d. Algoritme memiliki kondisi akhir atau pemecahan masalah/keluaran
- e. Masalah yang harus diselesaikan oleh algoritma

Sedangkan sifat dalam algoritma meliputi,:

- a. Input: Algoritma punya masukkan sebelum algoritma dieksekusi.
- b. Output: hasil luaran algoritma adalah dari metode yang diimplementasikan sebagai jalan keluar masalah
- c. Kepastian: Langkah pada algoritma didefinisikan secara nyata untuk menjelaskan /memutuskan secara benar.
- d. Solusi hingga: Algoritme harus memberikan beberapa kondisi akhir atau keluaran hingga menurut setiap kondisi awal atau masukan yang diberikan.
- e. Efektivitas: Mencapai keberhasilan dalam menghasilkan hasil yang diharapkan dan memberikan solusi yang diharapkan.

f. Kemman: Prinsip validitas atau validitas tidak universal.

3. Sejarah Pemograman

Melalui Pemahaman mengena sejarah kemajuan dunia pemrograman, dimungkinkan untuk mengungkap sumber pembentukan teknologi untuk peran pemrograman makin dibutuhkan saat ini. Pemrograman dibentuk sebagai hasil temuan yang dibuat dengan teknologi dari metode teknologi tertentu. Salah satu cabang ilmu pengetahuan, pemrograman dari sejak tahun 1822. Ada penemuan baru dalam teknologi mesin yang mampu mengolah data ditemukan oleh Charles Babbage. Mesin pengolah data pertama disebut Mesin Perbedaan oleh Babbage. Sayangnya, perbedaan mesin Babbage saat itu hanya bisa menghasilkan satu jenis hasil atau output. Selama bertahun-tahun, Difference Engine dikembangkan dan semakin lancar dalam menghasilkan keluaran data yang diharapkan. Sekitar 10 tahun setelah penemuan pertama, pada tahun 1849, Charles Babbage mampu mengembangkan versi ke-2 dari mesin perbedaan data, yakni mesin pengolah.

Penemuan ini diturunkan dari generasi ke generasi, putra Charles Babbage Henry Prevost melanjutkan proses penemuan ayahnya. Saat itu, Prévost melakukan percobaan dengan membuat salinan mesin kalkulasi algoritma yang dimiliki oleh Difference Engine, salinan perhitungan algoritma tersebut kemudian dikirim oleh Prévost ke berbagai institusi IT di seluruh dunia. Sejalan dengan difusi mesin algoritma Diferensial Engine milik keluarga Prévost, perkembangan mulai terjadi di dunia komputer, salah satunya pada tahun 854 akhirnya tercipta logika tentang hubungan antar komponen aritmatika, logika yang ditemukan dan dikembangkan oleh George Boole. Boole berhasil menemukan aturan sistem logika yang sekarang kita kenal sebagai logika Boolean. Aturan logika Boolean menyatakan bahwa ada hubungan antara komponen mayor, minor, sama dengan dan tidak sama dengan komponen aritmatika. sistem logis ini masih digunakan sampai sekarang untuk menyusun sistem pemrograman.

Sebagai pelopor kemajuan teknologi, jalur pengembangan bahasa pemrograman masih sangat panjang. Setelah munculnya teori Boolean, proses perkembangan logika aritmatika terus berkembang di masyarakat dunia, pada tahun 1935 kalkulator biner pertama yang disebut Z1 oleh Konrad Zuse

muncul. Zuse adalah seorang ilmuwan Jerman dan pada saat itu ia berhasil membuat kalkulator biner pertamanya dengan pengetahuan logika yang ia kembangkan. Setelah membuat Z1, Zuse dipanggil untuk dinas militer. Dedikasinya dilanjutkan dengan menciptakan seri berikutnya dari teknologi komputer biner, Z2, Z3 dan Z4 pada tahun 1939. Baru setelah ia mengembangkan Z4 Zuse menyadari bahwa bahasa pemrograman mesin yang ada sangat kuat. Selain itu, mesin yang ada bahasa masih merupakan bahasa tingkat rendah berupa jawaban "ya" dan "tidak" hanya untuk sekumpulan kode 0 dan 1. Pada tahun 1945, tingkat tinggi pertama di dunia.

Muncul bahasa pemrograman tingkat yang disebut Plankalkul alias Plan Calculus, penerapan bahasa Plankalkul terlihat dalam penemuan mesin catur terkomputerisasi pertama di dunia. Tetapi pada tahun yang sama terjadi kecelakaan serius yang merusak sistem digital. Setelah melakukan riset akhirnya saya menemukan bug (debugging) pada sistem. Bug inilah yang sampai sekarang dikenal sebagai gangguan yang dapat membuat sistem bekerja dengan tidak semestinya dimana menyebabkan sistem tidak berfungsi dan bahkan menyebabkan beberapa masalah sebagai akibatnya. Namun, kemunculan bahasa pemrograman Plankalkul tampaknya menjadi "suntikan motivasi" bagi para aktivis komputer dan ilmuwan untuk menciptakan bahasa pemrograman yang lebih baik.

Dilanjutkan pada tahun 1949, Kode Pendek ditemukan sebagai bahasa pemrograman tingkat tinggi lainnya. Kode Pendek ini juga digunakan untuk mengembangkan teknologi komputer elektronik John Mauchly. Sayangnya, Short Code memiliki sistem pemrosesan kode yang lebih lama karena program harus ditransfer ke mesin bahasa saat menjalankannya. Pada awal 1950-an, Alick Glennie mengembangkan bahasa pemrograman yang disebutnya Autocode. Bahasa Autocode digunakan sebagai media kompilasi, dengan tugas utama melakukan konversi otomatis ke bahasa mesin. Bahasa Autocode ini pertama kali diterapkan pada tahun 1952 untuk komputer Mark 1 milik University of Manchester. Selanjutnya, perkembangan bahasa pemrograman menjadi semakin pesat. Ilmuwan komputer Brian Kernighan dan Dennis Ritchie menemukan bahasa pemrograman C, yang masih banyak digunakan sampai sekarang. Bahasa C pada awalnya dibuat untuk menggunakan engine DEC PDP11, namun berkat hadirnya bahasa C, muncul berbagai macam bahasa pemrograman baru seperti Java, C++, C# dan sebagainya. Dengan

perkembangan bahasa pemrograman dan pemrograman, fungsinya menjadi semakin beragam. Dua konsep terbesar dari pemrograman komputer modern, yaitu layanan Internet dan platform seluler. Pakar IT akan terus mengembangkan sistem pemrograman untuk terus memberikan kemudahan akses kepada semua orang di masyarakat. Tujuannya agar setiap orang bisa lebih mudah melakukan segala pekerjaannya dengan teknologi. Hingga saat ini, hampir semua aspek kehidupan masyarakat telah dilanggar oleh teknologi komputer dan pemrograman.

4. Definisi Pemrograman

Kemajuan jaman menjadikan kebutuhan pemograman komputer dibutuhkan terutama bagi yang bekerja dan masyarakat abad ke-20. Peranan penting teknologi memiliki kepentingan utama masyarakat, dikarenakan komputer menjadi kebutuhan dan berperan dalam hidup banyak masyarakat diberbagai bidang. Seiring kebutuhan teknologi dibutuhkan manusia yang mengetahui dan tahu dalam membuat pemogramab sebagai suatu penyusun dalam program yang di buat dan gunakan, pemograman komputer sebagaimakna proses menyusun, melakukan pemeliharaan, proses modify agar mesin mampu menerima perintah manusia, yakni melakukan pengolahan data dari input hingga menghasilkan output yang dibutuhkan

Proses lengkap diawali dari pengkodean, pengujian code, sampai koneksi debug yang ada pada pemrograman, serta pemeliharaan code program melalui regenerasi perangkat lunak. Tujuan pemograman untuk membuat suatu program yang dapat dipahami komputer, sehingga komputer dapat menjalankan pekerjaan sesuai yang diingkan user. Untuk melakukan pemograman yang harus dimiliki adalah ketrampilan bahasa pemograman, algoritma, matematika dan logic. Selain itu, dari survei setiap tahun, bahasa pemrograman paling populer yang dipakai pengembang. Bahasa pemrograman yang paling populer digunakan adalah Java, Visual Basic dan Python, mempelajari bahasa pemrograman populer yang digunakan juga akan mempercepat diterima bekerja di perusahaan, karena akan butuh programer dengan keterampilan bahasa tersebut.

Profesi programmer merupakan profesi yang memiliki prospek paling besar di era modern ini. Berbeda dengan dulu, ketika digital atau IT belum

masif seperti sekarang ini, saat itu profesi developer dipandang sebelah mata. Saat ini, karena teknologi merajalela dan bahkan "tertanam" ke dalam kehidupan manusia, sumber daya pemrograman yang berkualitas telah menjadi suatu item yang paling dibutuhkan di diberbagai tempat. Oleh karena itu, tidak heran apabila makin banyak belajar coding setiap hari. Berbekal pengetahuan, keterampilan, dan semangat pemrograman. Faktanya, semakin mempelajari pemrograman, semakin dekat dengan karir yang sukses sebagai programmer profesional. Tapi apa cara terbaik untuk belajar pemrograman. Pada awalnya, tentu saja, perlu menguasai pemahaman tentang pemrograman dan sistem kerja secara benar dan lengkap. Dengan begitu programmer pemula dapat belajar ilmu pemrograman secara baik.

5. Jenis Pemrograman Komputer

Setelah memahami pengertian pemrograman dan sejarah perkembangan pemograman, sebaiknya faham jenis pemrograman apa yang digunakan untuk menyusun sebuah sistem komputer. Dalam praktiknya terdapat berbagai jenis program komputer. Jenis pemrograman dikategorikan sesuai dengan fungsinya masing-masing. Dengan Pemhaman terhadap jenis pemrograman komputer yang ada akan memberi penggambaran keseluruhan pekerjaan sebuah komputer dan juga bisa lebih siap dengan baik.

a. Notasi Algoritma

Rinaldi Munir dalam buku Algoritma dan Pemrograman (1997) notasi algoritma ialah desain berisi langkah terurut untuk memecahkan masalah dimana tidak masuk dalam golongan bahas apemograman apapun. Sehingga notasi algoritma merupakan dasar dibuatnya suatu program komputer dan dapat diterjemahkan ke dalam berbagai bahasa pemrograman. Notasi algoritma mampu diartikan ke berbagai bahasa pemrograman. Bagiannya Secara umum notasi algoritma terdiri dari tiga jenis, yaitu kalimat deskriptif, pseudocode, dan flowchart. Berikut adalah ketiga penjelasan dari tiga notasi pemrograman tersebut Kalimat Deskriptif Notasi algoritma kalimat deskriptif adalah notasi algoritma yang menggunakan bahasa Inggris atau bahasa Indonesia dalam menjabarkan desain langkah pemrograman yang akan dibuat. Dalam notasi kalimat deskriptif akan dijumpai berbagai kata kerja dalam bahasa manusia seperti

mulai, tulis, baca, tampilkan, maka, ulangi, read, print, if, dan end. Notasi kalimat deskriptif ditulis dengan kerangka utama berupa judul, deklarasi, dan deskripsi. Deklarasi adalah bagian yang mendefinisikan variabel, fungsi, dan juga konstanta yang akan digunakan dalam pemrograman.

Adapun deskripsi berisi uraian langkah atau inti dari algoritma untuk suatu program komputer. Contoh notasi algoritma kalimat deskriptif: Notasi algoritma Seperti disebutkan di atas, algoritma ini merupakan langkah terurut atau perintah yang dijalankan guna menyelesaikan suatu permasalahan. Algoritma ditulis dengan notasi, penilaian algoritma adalah dasar yang harus diketahui apabila ingin belajar pemrograman, dikarenakan dalam algoritma terdapat struktur-struktur program. Karakteristik algoritma dengan notasi baik apabila mampu diartikan oleh macam-macam bahasa pemrograman. Terpenting mengenai rating ialah terbaca dengan mudah serta memahaminya. Namun untuk menghindari kerancuan, terhadap notasi harus tetap diperhatikan. Berikut 3 notasi yang biasa digunakan dalam penulisan algoritma, yakni:

- 1) Notasi natural
- 2) Flowchart / Flowchart
- 3) PseudoCode (seperti kode)

Penulisan algoritma dengan notasi natural berarti menulis instruksi yang dijalankan dalam menyelesaikan masalah berbentuk kalimat deskripsi. Dengan notasi, deskripsi setiap langkah dijelaskan dalam bahasa sederhana. Prosesnya dimulai dengan kata kerja seperti 'membaca', 'menghitung', 'membagi', 'mengganti', dan seterusnya, sedangkan pernyataan kondisional dilambangkan dengan 'jika ... maka' ... '. Notasi ini baik untuk algoritma sederhana, tetapi untuk masalah dengan algoritma komplek maka tidak efisien. Selain itu, mengubah notasi algoritma ke notasi quari pemrograman relatif memiliki kesulitan.

b. Flowchart

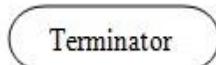
Pengkonversian notasi algoritma yang sulit, maka dibantu dengan diagram alir atau flowchart dimana memiliki 2 macam diagram alir,

- 1) Sistem diagramair/flowcart, yang menjelaskan gambaran
- 2) Data diproses dalam program
- 3) Macam-macam perangkat yang digunakan oleh file
- 4) operasi file
- 5) masuk atau keluar

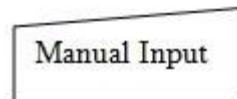
Bagan alir program (biasa disebut flowchart saja), adalah gambar yang menggambarkan urutan:

- 1) Membaca data
- 2) Mengolah data
- 3) Membuat keputusan data
- 4) Menyajikan hasil pengolahan data

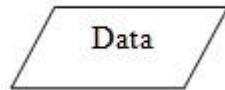
Flowcarht adalah diagram alur dalam program komputer pada bahasa basic, fortran dan cobol. Secara visua alur perintah pemograman menunjukkan struktur program yang akan dibuat. Notasi flowcart sesuai untuk menangani masalah sedehana,tidak sesuai untuk masalah yang kompleks karena butuh halaman saat perancangan bagannya yang banyak. Selain itu untuk menuangkan kebahas apemograman sukar, simbolpada flowcarth sebagai berikut.



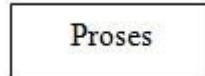
Lambang terminator untuk menyatakan bahwa algoritma dimulai dan diakhir



Input secara manual merupakan input kotak yang berfungsi membaca data dari user bisa berupa variabel atau nilai



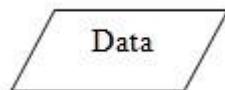
Kotak data dipakai sebagai pembaca dan penampil data, seperti mesin ATM dimana kita ingin mengetahui saldo, ATM inilah sebagai penampil data



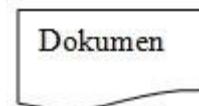
Kotak penugasan atau proses adalah melakukan hitung matematika logika dimana menghasilkan suatu nilai variabel



Kotak keputusan untuk percabangan yang di gunakan sebagai pengambil keputusan dengan kondisi yang ada yakni keputusan yang salah atau yang benar



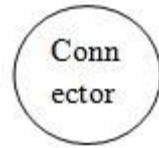
Selain sebagai input berfungsi sebagai output untuk menampilkan informasi



Kotak luaran dokumen untuk melakukan pencetakan dari hasil luaran

Selain diatas terdapat simbol atau lambang di bawah ini.

→ Lambang anak panah



Simbol penghubung jika alur putus tapi masih dalam laman sama



Simpul penghubung oof page, dimana sebagai penghubung kembali alur yang putir dihalaman yang beda

c. Pseudocode

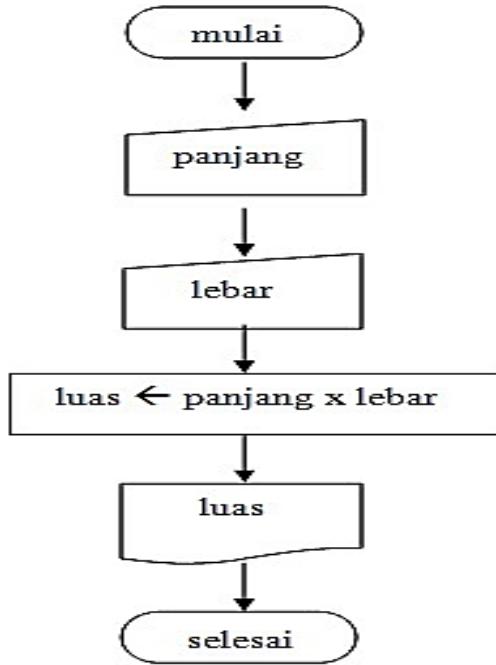
Pseudocode merupakan bentuk notasi pemograman bahasa tinggi yang ada dalam bahasa C ataupun pascal, beberapa notasi memiliki kesamaan contohnya if then else, while do, repeat until, read, write dan lainnya. Pada pseudocode berbeda dengan bahasa pemograman yang menggunakan titik koma, format, kata khusus. Pseudocode lebih bisa diterima yang penting perintah masih dipahami oleh pembaca, kelebihannya dengan adanya pseudocode dapat dengan mudah menuangkan kebahas apemograman, cocok digunakan pada algoritma kompleks karena adanya korespondensi antara bahasa pemograman dan pseudocode. Contoh studi kasus : buat algoritma hitung luas persegi panjang dengan notasi algoritma!

Jawab :

Notasi Alami

- 1) pemperoleh nilai dari panjang
- 2) pemperoleh nilai dari lebar
- 3) Kalikan nilai panjang dengan lebar dan beri hasilnya pada luas
- 4) Tampilkan nilai hasil diluas

Flowcard sebagai berikut:



3. Pseudo-code

p, l, ls : numeric

begin

read (p, l)

ls : p x l

print (ls)

end.

6. Konsep Struktur Data

Data menjadi hal yang terpenting dan tidak diabaikan dalam penggunaan komputer. Data dapat diperoleh dari berbagai sumber, seperti hasil pengukuran laboratorium, hasil survei, hasil kuisioner, dll. Secara sederhana, dapat mengumpulkannya dalam struktur data yang berisi informasi tentang hubungan antar elemen yang dikandungnya. Struktur data sebagai sarana untuk

menyimpan, menyusun, dan mengatur data pada media penyimpanan komputer agar data tersebut digunakan secara efisien. Struktur data berfungsi untuk menyimpan data dalam bentuk yang efisien, memudahkan pembacaan data, membantu kinerja algoritma. Jadi, dari struktur data itu sendiri, berguna untuk mengatur data yang disimpan lebih mudah dibaca, dimodifikasi, dan diproses ulang nanti. Namun, kita harus ingat bahwa struktur data berevolusi dan setiap perkembangan memiliki trade-off Berikut adalah beberapa contoh struktur data:

- a. Tabel
- b. linked list: tunggal dan ganda
- c. stack
- d. Queue
- e. Tree
- f. Balanced Tree
- g. Graph

Jadi mengapa perlu mempelajari struktur data? akan lebih mudah untuk melakukan pengurutan huruf yang benar, tanpa struktur data jika dibayangkan jutaan data pengguna Facebook apabila tidak disimpan teratur dan jika data dari GPS tidak menyimpan huruf yang benar.

Dalam struktur data merupakan tempat disimpannya data yang mampu direpresentasi dalam komputer saat data dibutuhkan, data yang disimpan dalam bentuk tulisan, video , gambar, suara, sinyal maupun simbol. Dalam struktur data terdapat data yang disimpan dan dikategorikan sesuai dengan tipe data dimana dapat dikategorikan menjadi tipe data yang simple atau sederhana misalnya tipe data tunggal sebagai contoh integer, real, boolean dan char. Sedangkan tipe data sederhana yang majemuk sebagai contoh adalah string. Secara sederhana dalam struktur data memiliki bentuk array dan record, sedangkan pada struktur data majemuk terbagi menjadi linier terdiri dari bentuk operasi Stack, Queue, dan Linier Linked List dan Nonlinier terdiri dari Binary Tree, Binary Search Tree, Graph, dan lainnya. Penggunaan struktur data yang sesuai akan mampu menghasilkan sebuah program dengan basis algoritma yang jelas dan sesuai, sehingga lebih efisien dan simpel sehingga

lebih dipahami. Kemiripan dunia nyata dalam struktur data dalam bahasa pemrograman digunakan secara digital, yang memiliki fungsi sebagai berikut.

a. Memasukkan informasi

Input data yang diperoleh,yaitu Informasi apa yang dapat diinput dan data baru yang dapat ditambah diawal, tengah atau akhir. Apakah perlu perbaruan atau dimusnahkan.

b. Memproses informasi

Melakukan proses secara manipulasi dalam struktur data dimana dapat dilakukan secara bersama atau hasil dari pemrosesan lainnya,penyimpanan data, akomodasi data baru atau dihapus.

c. Maintaining informasi

Mempertahankan data disusun dalam struktur. Hubungan yang perlukan dan banyak memori yang mengalokasikan.

d. Mengambil informasi

Retrieving untuk menemukan dan mengembalikan data yang disimpan dalam struktur.

Memilih Struktur Data terbaik pada struktur data yang berbeda lebih cocok untuk tugas yang berbeda. Memilih struktur data yang salah dapat mengakibatkan kode lambat atau tidak responsif. Oleh karena itu, penting untuk mempertimbangkan beberapa faktor dalam mengambil keputusan:

a. Apa tujuan dari data?

b. Apakah ada struktur data dengan fungsi bawaan yang cocok untuk tujuan ini?

c. keinginan menemukan, mengurutkan, atau menelusuri data sehingga beberapa struktur data lebih cocok daripada yang lain?

d. Jika ingin atau perlu mengontrol bagaimana memori dicadangkan untuk menyimpan data.

Meskipun alokasi memori bukanlah sesuatu yang perlu dipertimbangkan dalam bahasa, Python atau Javascript, tapi perlu pada bahasa lain seperti C. Dua struktur data mungkin dapat melakukan tugas sama, tetapi satu mungkin lebih cepat atau pertimbangan runtimenya

C. LATIHAN SOAL

1. Jelaskan menurut anda mengenai Algoritma dalam ilmu komputer?
2. Jelaskan apa yang anda ketahui mengenai Struktur Data dan berikan contohnya?
3. Jelaskan apa yang anda ketahui mengenai Pemograman dan berikan contoh bahasa Pemograman minimal 5?
4. Jelaskan apa yang anda ketahui kelebihan bahasa Python dengan Bahasa Pemograman lainnya
5. Butalah notasi dan flowcart dalam studi kasus suatu algoritma dan pemogramannya!

D. REFERENSI

- Amanulhaq, A. A. (2021). Implementasi Algoritma Image Hashing Dan Hamming Distance Untuk Deteksi Kemiripan Gambar. *repository ITS*.
- Basant Agarwal, B. B. (2018). Hand-On Data Structures and Algorithms With Python. London: Packt Publishing.
- Emi Sita Eriana, A. Z. (2021). Praktikum Algoritma dan Pemrograman. Tangerang Selatan: Unpam Press.
- Jodi, u. R. (2020). Algoritma Dan Struktur Data.
- 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.
- 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). Pendekripsi Kantuk Secara Real Time Menggunakan Pustaka OPENCV dan DLIB PYTHON. Sainstech : Jurnal Penelitian dan Pengkajian Sains dan Teknologi.

PERTEMUAN 2

PENGENALAN PEMROGRAMAN PYTHON

A. TUJUAN PEMBELAJARAN

Setelah mempelajari materi ini diharapkan mahasiswa mengenal pemrograman python, installasi perangkat lunak dan mampu membuat program dengan Python.

B. URAIAN MATERI

1. Pemrograman Python

Menemukan struktur data dan algoritma mengharuskan mengomunikasikan instruksi secara rinci ke komputer. Cara terbaik untuk melakukan komunikasi semacam itu adalah menggunakan bahasa komputer tingkat tinggi, seperti Python. Pemrograman Python bahasa ini awalnya dikembangkan oleh Guido van Rossum pada awal 1990-an, dan sejak itu menjadi bahasa yang digunakan secara menonjol dalam industri dan pendidikan. Versi utama kedua dari bahasa, Python 2, dirilis pada tahun 2000, dan yang ketiga versi utama, Python 3, dirilis pada tahun 2008. tercatat bahwa ada kompatibilitas yang signifikan antara Python 2 dan Python 3.

Bahasa pemograman Python termasuk bahasa pemograman yang mudah dibaca dan terstruktur, Namun pertama-tama, akan meninjau beberapa pola untuk pemrograman dan lihat bahasa pemrograman Python untuk memastikan dalam memahami dasar struktur dan sintaks bahasa. Untuk mulai menulis program menggunakan Python, perlu menginstal Python di perangkat komputer yang sesuai dengan kapasitas memori. Pada Bahan ajar ini akan menggunakan contoh menggunakan Python 3. Python 2 tidak kompatibel dengan Python 3 jadi pastikan apabila ingin melakukan menginstal Python 3 atau yang lebih baru di komputer. Saat menulis program dalam bahasa apa pun, Lingkungan Pengembangan Terpadu yang baik (IDE) adalah alat yang berharga, jadi Anda pasti ingin memasang IDE juga.

Python secara formal adalah bahasa yang ditafsirkan. Perintah dieksekusi melalui bagian dari perangkat lunak yang dikenal sebagai julu bahasa Python.

Interpreter menerima perintah, mengevaluasi perintah tersebut, dan melaporkan hasil dari perintah tersebut. Selagi interpreter dapat digunakan secara interaktif (terutama saat debugging), seorang programmer biasanya mendefinisikan serangkaian perintah terlebih dahulu dan menyimpan perintah tersebut dalam file teks biasa yang dikenal sebagai kode sumber atau skrip. Untuk Python, kode sumber secara konvensional disimpan dalam file bernama dengan akhiran .py (mis., demo.py).

Pada sebagian besar sistem operasi, juru bahasa Python dapat dimulai dengan mengetik python dari baris perintah. Secara default, penerjemah dimulai secara interaktif mode dengan ruang kerja yang bersih. Perintah dari skrip yang telah ditentukan sebelumnya disimpan di file (mis., demo.py) dieksekusi dengan memanggil penerjemah dengan nama file sebagai argumen (mis., python demo.py), atau menggunakan flag -i tambahan untuk jalankan skrip dan kemudian masuk ke mode interaktif (mis. python -i demo.py). Banyak lingkungan pengembangan terintegrasi (IDE) menyediakan perangkat lunak yang lebih kaya platform pengembangan untuk Python, termasuk satu bernama IDLE yang disertakan dengan distribusi Python standar. IDLE menyediakan editor teks tertanam dengan dukungan untuk menampilkan dan mengedit kode Python, dan debugger dasar, memungkinkan langkah-demi-langkah eksekusi program sambil memeriksa nilai-nilai variabel kunci. Alasan utama Pyton untuk dipelajari oleh calon programmer atau progemmer adalah:

- a. Python sangat mudah dipelajari. Sintaksnya mudah dan kodennya sangat mudah dibaca.
- b. Python memiliki banyak aplikasi. Ini digunakan untuk mengembangkan aplikasi web, ilmu data, IoT, pengembangan aplikasi yang cepat, dan sebagainya.
- c. Hal ini memungkinkan Anda untuk menulis program dalam baris kode yang lebih sedikit daripada sebagian besar bahasa pemrograman.
- d. Ini memiliki dukungan komunitas yang sangat besar & forum aktif untuk mendukung pengguna.
- e. Kehadiran Modul Pihak Ketiga membuat bahasa Python lebih kuat.

- f. Perpustakaan pada Python dukungan ekstensif (mis: NumPy untuk perhitungan numerik, Panda untuk analisis data dll) membantu pengguna untuk memecahkan masalah besar dengan mudah.
- g. Ini memiliki struktur data yang sangat user-friendly yang menyederhanakan desain kode dan logika.
- h. Popularitas Python berkembang pesat. Sekarang ini adalah salah satu bahasa pemrograman yang paling populer.

2. Baris Perintah

Program terdiri dari perintah yang memberi tahu komputer apa yang harus dilakukan. Ini perintah disebut pernyataan, yang dijalankan komputer. Bab ini mendeskripsikan pernyataan Python yang paling sederhana dan menunjukkan bagaimana pernyataan tersebut digunakan untuk melakukan aritmatika, yang merupakan salah satu tugas paling umum untuk komputer dan juga tempat yang bagus untuk mulai belajar membuat program. Itu juga dasar dari hampir segala sesuatu yang mengikuti.

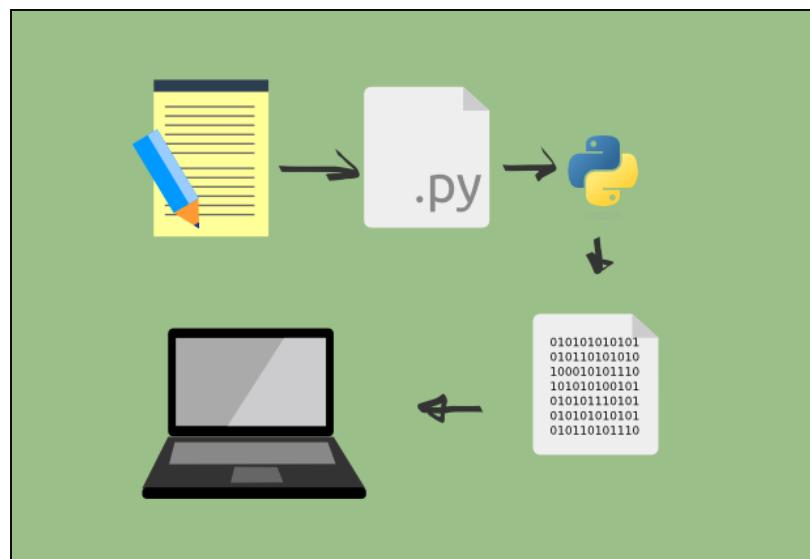
a. Aturan Penulisan

Python berorientasi objek dimana semua item data dalam Python adalah objek. Dengan Python, item data yang dapat dianggap serupa diberi nama berdasarkan tipe atau kelas. Syarat type dan class dalam Python adalah sinonim: mereka adalah dua nama untuk hal yang sama. Jadi ketika membaca tentang tipe dalam Python, baiknya untuk dapat memikirkan kelas atau sebaliknya.

Ada beberapa tipe data bawaan dalam Python termasuk int, float, str, list, dan dict yang merupakan kependekan dari kamus. Jenis data ini dan operasi terkaitnya disertakan dalam lampiran di akhir teks sehingga memiliki referensi cepat jika perlu merujuknya saat memprogram. Selain itu juga bisa mendapatkan bantuan untuk jenis apa pun dengan mengetik help(typename) di shell Python, di mana typename adalah tipe atau kelas di Python. Referensi bahasa yang sangat bagus dapat ditemukan di <http://python.org/doc>, the situs web dokumentasi resmi Python algoritma untuk menangani sejumlah besar data. Tanpa pemahaman yang tepat efisiensi, adalah mungkin untuk menghentikan komputer tercepat sekalipun

ketika bekerja dengan kumpulan data yang besar. Ini telah terjadi sebelumnya, dan akan segera mengerti betapa mudahnya hal itu bisa terjadi. Tapi pertama-tama, akan meninjau beberapa pola untuk pemrograman dan melihat bahasa pemrograman Python untuk memastikan untuk memahami dasarnya struktur dan sintaksis bahasa.

Saat menulis program dalam bahasa apa pun, Integrated Development Environment (IDE) yang baik adalah alat yang berharga sehingga ingin menginstal IDE. Ada beberapa konsep umum tentang Python yang harus ketahui saat membaca teks. Python adalah bahasa yang ditafsirkan. Itu berarti tidak harus pergi melalui langkah tambahan apa pun setelah menulis kode Python sebelum dapat menjalankannya. Kamu bisa cukup tekan tombol debug di Wing IDE dan itu untuk menyimpan program yang dibuat jika belum melakukannya setidaknya sekali, kemudian akan menjalankan program yang dibuat, tidak akan mendapatkan kesalahan apa pun sebelum menjalankan program seperti yang dilakukan dengan beberapa bahasa pemrograman.



Gambar 2. 1 Gambar Alur pembuatan python

Python yang diciptakan oleh Guido van Rossum, dan dirilis pada tahun 1991. Ini adalah open-source dan dapat dengan bebas menggunakan dan mendistribusikan Python, bahkan untuk penggunaan komersial. Ini sangat populer untuk pengembangan web dan dapat membangun hampir semua hal seperti aplikasi seluler, aplikasi web, alat, analisis data, pembelajaran mesin, dll. Ini dirancang untuk menjadi sederhana dan mudah seperti bahasa

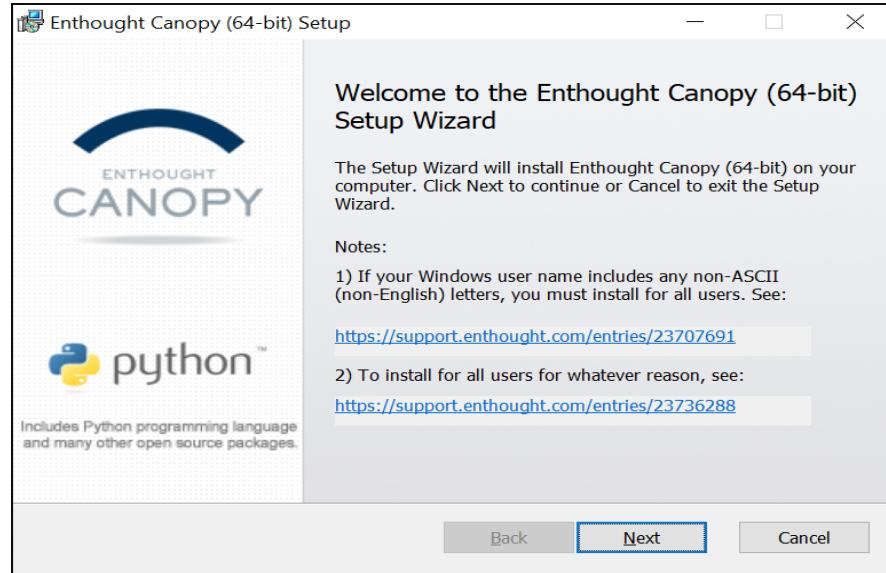
Inggris. Jauh lebih mudah untuk membaca dan menulis program Python dibandingkan dengan bahasa lain seperti C++, Java, C#. Ini sangat produktif dan efisien yang membuatnya menjadi bahasa pemrograman yang sangat populer. Program-program yang ditulis dalam Python secara khas jauh lebih pendek dibandingkan dengan program-program C atau C++, karena beberapa pertimbangan:

- a. Tipe data tingkat tinggi digunakan untuk menyatakan operasi kompleks dalam suatu statemen tunggal pengelompokan statemen telah selesai dengan indentasi sebagai pengganti dari pengurungan mulai dan akhiran.
- b. Tidak ada deklarasi-deklarasi argumentasi atau variabel yang diperlukan.
- c. Sintaks Python sangat bergantung pada penggunaan spasi. Pernyataan individu biasanya diakhiri dengan karakter baris baru, meskipun perintah dapat diperpanjang ke baris lain, baik dengan karakter garis miring terbalik penutup (\), atau jika pembukaan pembatas belum ditutup, seperti karakter { dalam mendefinisikan peta nilai spasi juga merupakan kunci dalam membatasi badan struktur kontrol dengan Python.

Secara khusus, blok kode diindentasi untuk menetapkannya sebagai badan kontrol struktur, dan struktur kontrol bersarang menggunakan peningkatan jumlah lakukan. Komentar adalah anotasi yang disediakan untuk pembaca manusia, namun diabaikan oleh Penerjemah Python. Sintaks utama untuk komentar dalam Python didasarkan pada penggunaan dari karakter #, yang menunjuk sisa baris sebagai komentar.

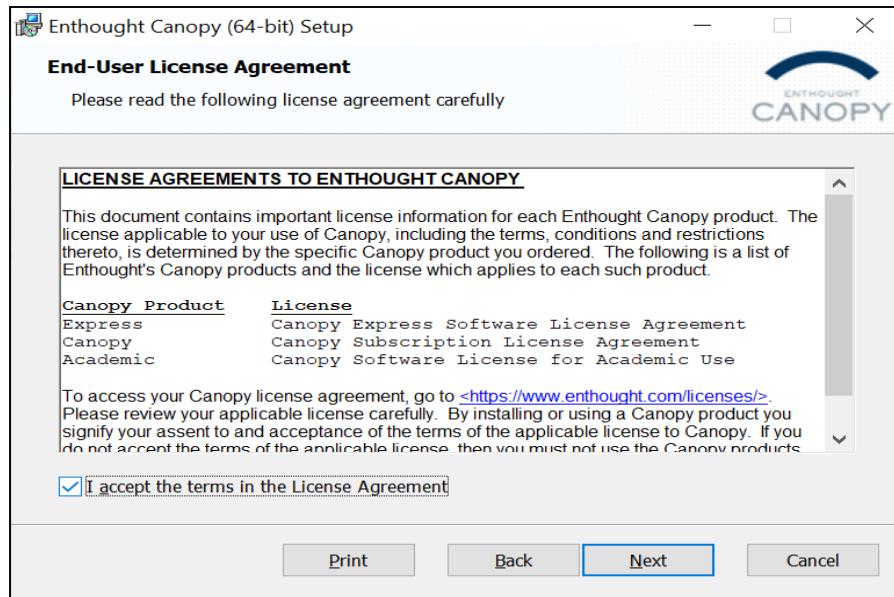
b. Instalasi Pyton

Siapkan software Pemrograman Python dapat di download diinternet, perangkat lunak yang digunakan adalah Enthought Canapy sebagai berikut.

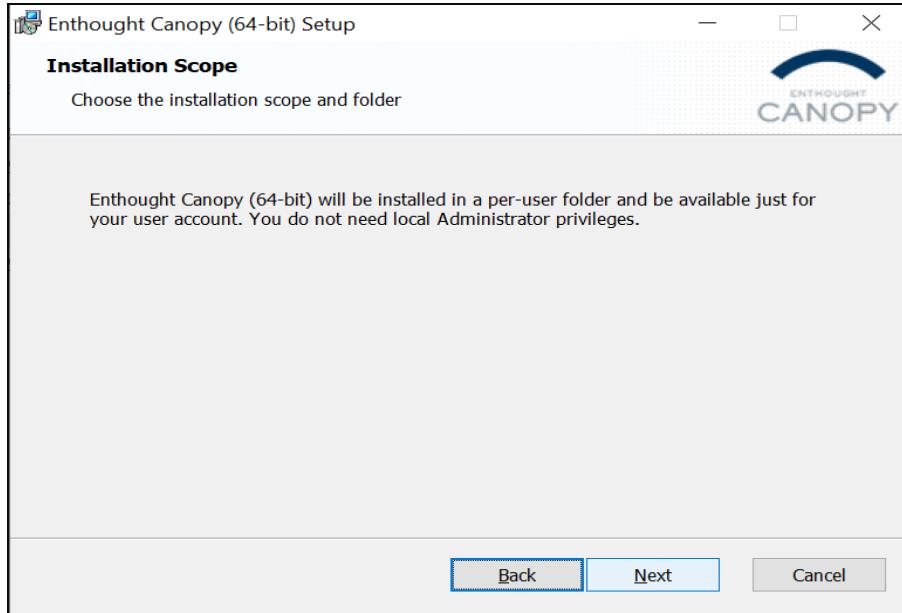


Gambar 2. 2 Installasi Python

Setelah tampil maka klik next sehingga tampil seperti gambar dibawah ini.

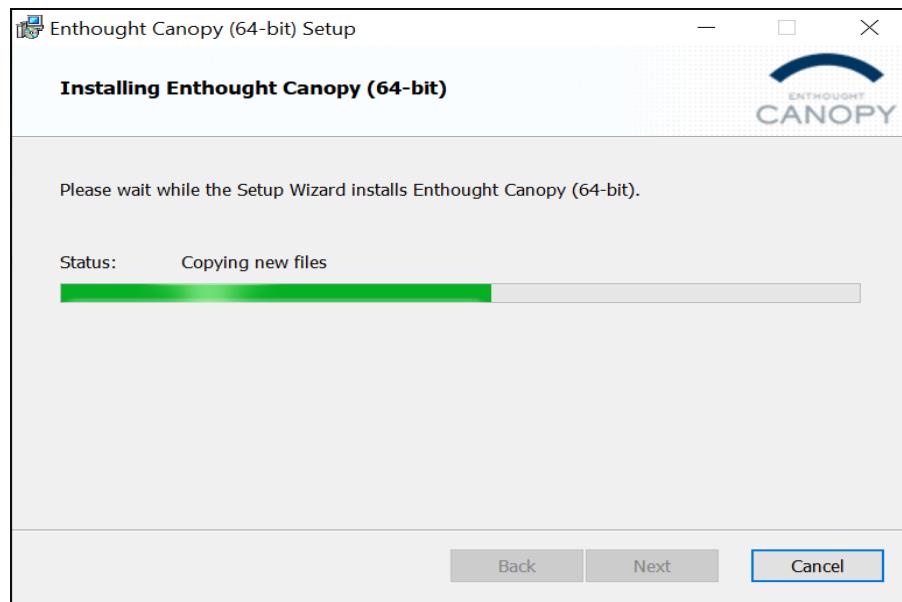


Ceklist dan persejutuan untuk melanjutkan instalasi, dan akan tampil seperti dibawah ini

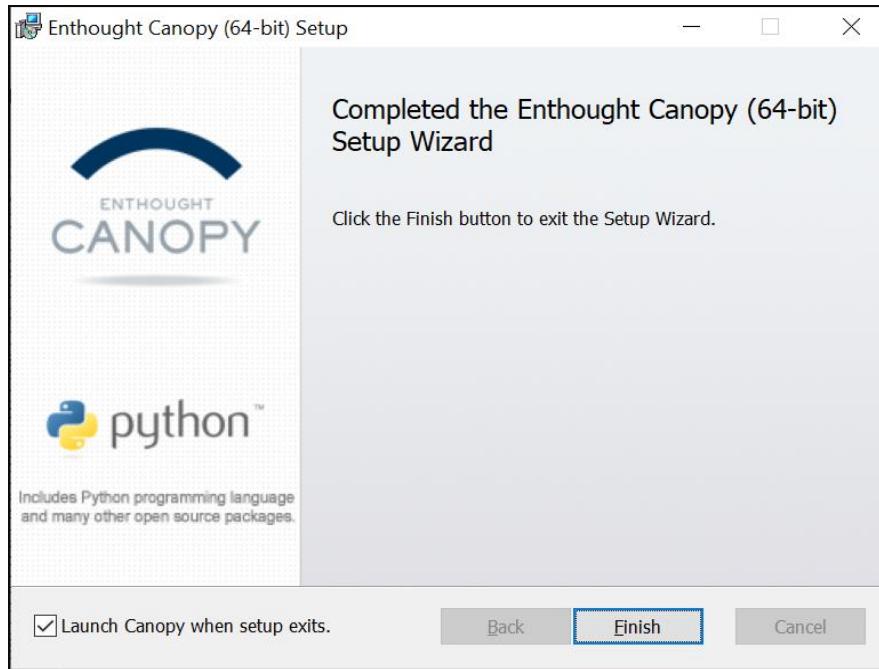


Klik next untuk melanjutkan install sehingga muncul

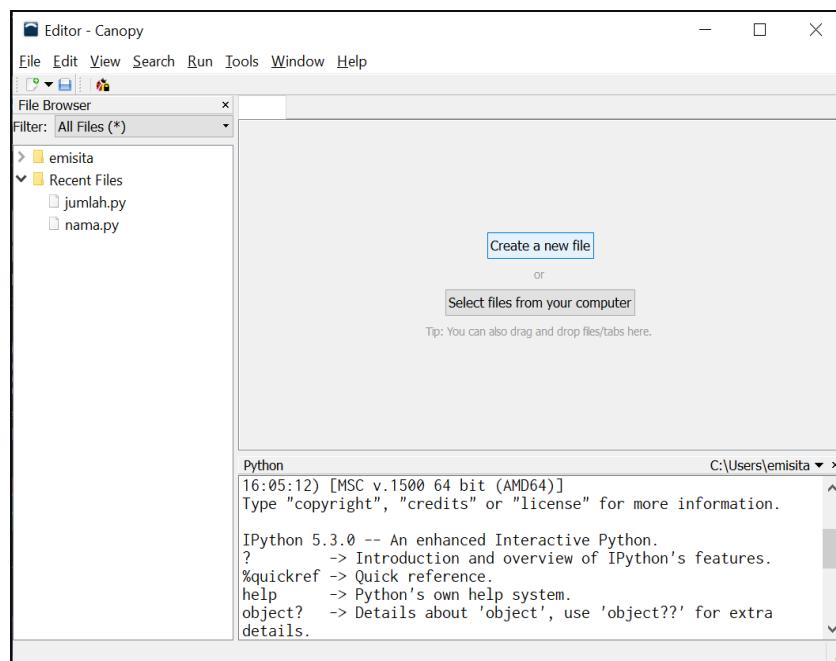
Pilih next lalu akan muncul status dalam copy file seperti dibawah ini.



Setelah 100% maka proses installasi selesai seperti dibawah



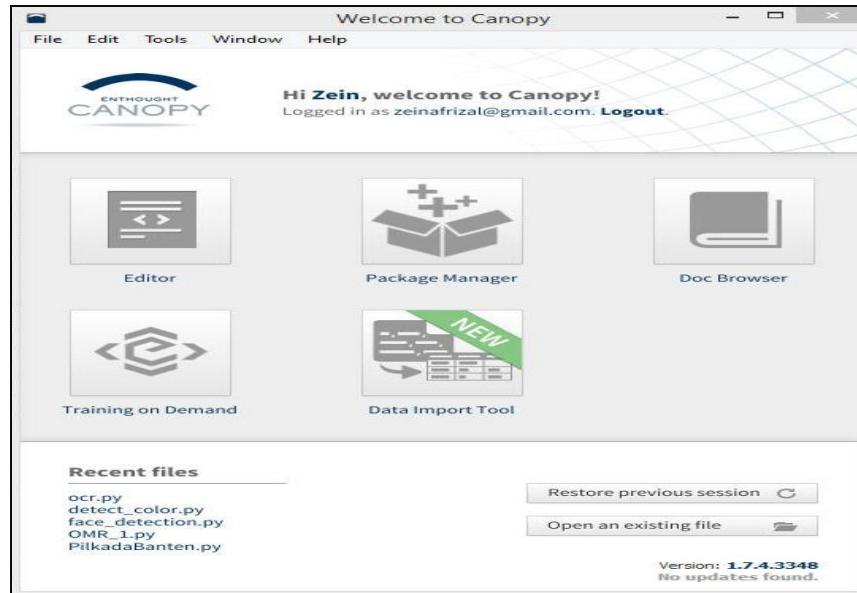
Pada tampilan selanjutnya adalah editor Canopy telah tampil seperti dibawah.



Dari tampilan diatas mengartikan bahwa pemrograman Python telah siap di gunakan pada software enthought canopy

c. Memulai Python

Bahasa ini sudah terinstal secara default di CANOPY berikut dengan editornya. Jadi hanya cukup klik icon Canopy di windows maka akan tampil seperti ini :



Untuk menulis program anda tingal klik icon editor maka akan tampil :

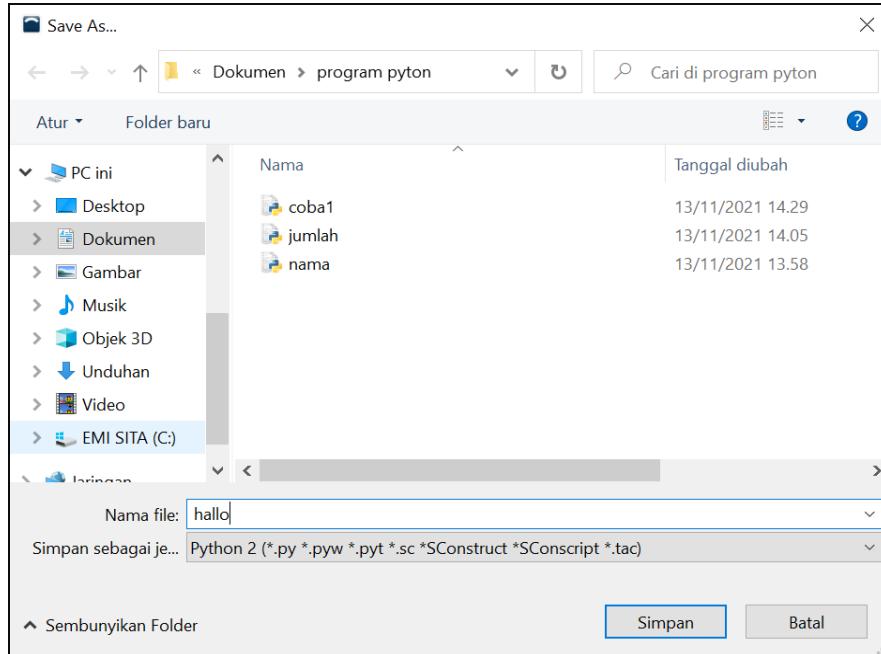
```

1 nama = "emi sita"
2 print("halo siapa namamu : " + nama)

```

The screenshot shows the 'Editor - Canopy' window. On the left is a 'File Browser' sidebar with a tree view showing 'emisita' and 'Recent Files' with 'jumlah.py' and 'nama.py'. The main area contains a code editor with the above Python script. Below the code editor is a 'Python' terminal window showing the IPython 5.3.0 prompt and help documentation. The bottom status bar indicates 'Cursor pos 2: 38' and 'Python 2'.

Setelah melakukan penulisan sesuai sintax dan source code dapat menyimpan dalam lokasi memory yang dibuat dengan penamaan file dibelakangan .py kemudian save



Python adalah bahasa yang sangat sederhana, dan memiliki sintaks yang sangat mudah. Ini mendorong pemrogram untuk memprogram tanpa kode boilerplate (disiapkan). Direktif paling sederhana di Python adalah perintah "print" ini hanya mencetak sebuah baris (dan juga menyertakan baris baru, tidak seperti di C). Ada dua versi Python utama, Python 2 dan Python 3. Python 2 dan 3 sangat berbeda. Tutorial ini menggunakan Python 3, karena lebih tepat secara semantik dan mendukung fitur yang lebih baru.

Misalnya, satu perbedaan antara Python 2 dan 3 adalah pernyataan cetaknya. Dalam Python 2, pernyataan "print" bukanlah sebuah fungsi, dan oleh karena itu ia dipanggil tanpa tanda kurung. Namun, di Python 3, ini adalah sebuah fungsi, dan harus dipanggil dengan tanda kurung. Untuk mencetak string dengan Python 3, cukup tulis: `print("Selamat dating di Python.")`

1) Variables and Types

Python sepenuhnya berorientasi objek, dan tidak "diketik secara statis", dan tidak perlu mendeklarasikan variabel sebelum menggunakannya, atau mendeklarasikan tipenya. Setiap variabel dalam Python adalah sebuah objek. Tutorial ini akan membahas beberapa tipe dasar variabel.

Angka

Python mendukung dua jenis bilangan - bilangan bulat (bilangan bulat) dan bilangan floating point (desimal). (Ini juga mendukung bilangan kompleks, yang tidak akan dijelaskan dalam tutorial ini).

Untuk mendefinisikan integer, gunakan sintaks berikut:

```
myfloat = 7.0
print(myfloat)
myfloat = float(7)
print(myfloat)
```

2) Lists

List sangat mirip dengan array. Mereka dapat berisi jenis variabel apa pun, dan dapat berisi variabel sebanyak yang Anda inginkan. List juga dapat diulang dengan cara yang sangat sederhana. Berikut ini contoh cara membuat list

```
mylist = []
mylist.append(1)
mylist.append(2)
mylist.append(3)
print(mylist[0]) # prints 1
print(mylist[1]) # prints 2
print(mylist[2]) # prints 3

# prints out 1,2,3
for x in mylist:
    print(x)
```

a) Operator Dasar

Bagian ini menjelaskan cara menggunakan operator dasar dengan Python.

b) Operator Arithmetik

Sama seperti bahasa pemrograman lainnya, operator penjumlahan, pengurangan, perkalian, dan pembagian dapat digunakan dengan angka.

```
number = 1 + 2 * 3 / 4.0
```

```
print(number)
```

c) Menggunakan Operator dengan String

Python mendukung string penggabungan menggunakan operator penambahan:

```
helloworld = "hello" + " " + "world"  
print(helloworld)
```

Python juga mendukung perkalian string untuk membentuk string dengan urutan berulang:

```
lotsofhellos = "hello" * 10  
print(lotsofhellos)
```

Menggunakan Operator List

List dapat digabungkan dengan operator tambahan:

```
even_numbers = [2,4,6,8]  
angka_ganjil = [1,3,5,7]  
all_numbers = odd_numbers + even_numbers  
cetak (all_numbers)
```

d) Format String

Python menggunakan pemformatan string gaya-C untuk membuat string baru yang diformat. Operator "%" digunakan untuk memformat sekumpulan variabel yang diapit oleh "tuple" (list ukuran tetap), bersama dengan format string, yang berisi teks normal bersama dengan "penentu argumen", simbol khusus seperti "% s" dan "% d".

Misalkan memiliki variabel yang disebut "nama" dengan nama pengguna di dalamnya, dan kemudian ingin mencetak (salah untuk pengguna tersebut.)

```
# This prints out "Hello, John!"  
  
name = "John"  
  
print("Hello, %s!" % name)
```

Objek apapun yang bukan string dapat diformat menggunakan operator % s juga. String yang dikembalikan dari metode "repr" dari objek tersebut diformat sebagai string. Sebagai contoh:

```
# This prints out: A list: [1, 2, 3]
mylist = [1,2,3]
print("A list: % s" % mylist)
```

e) Kondisi

Python menggunakan logika boolean untuk mengevaluasi kondisi. Nilai boolean True dan False dikembalikan saat ekspresi dibandingkan atau dievaluasi. Sebagai contoh:

```
x = 2
print(x == 2) # prints out True
print(x == 3) # prints out False
print(x < 3) # prints out True
```

Perhatikan bahwa penugasan variabel dilakukan menggunakan operator sama dengan tunggal "=", sedangkan perbandingan antara dua variabel dilakukan menggunakan operator sama dengan ganda "==" . Operator "tidak sama dengan" ditandai sebagai "!=".

f) Operator Boolean

Operator "dan" dan "atau" boolean mengizinkan pembuatan ekspresi boolean yang kompleks, misalnya:

```
name = "John"
age = 23
if name == "John" and age == 23:
    print("Your name is John, and you are also 23 years old.")

if name == "John" or name == "Rick":
    print("Your name is either John or Rick.")
```

g) Operator "dalam"

Operator "dalam" dapat digunakan untuk memeriksa apakah objek tertentu ada dalam wadah objek yang dapat diulang, seperti list:

```

name = "John"
if name in ["John", "Rick"]:
    print("Your name is either John or Rick.")

```

Python menggunakan lekukan untuk menentukan blok kode, bukan tanda kurung. Indentasi Python standar adalah 4 spasi, meskipun tab dan ukuran ruang lainnya akan berfungsi, selama itu konsisten. Perhatikan bahwa blok kode tidak memerlukan penghentian apa pun. Berikut adalah contoh penggunaan pernyataan "if" Python menggunakan blok kode:

```

statement = False
another_statement = True
if statement is True:
    # do something
    pass
elif another_statement is True: # else if
    # do something else
    pass
else:
    # do another thing
    pass

```

Pernyataan dievaluasi sebagai benar jika salah satu dari berikut ini benar:

- (1) Variabel boolean "Benar" diberikan, atau dihitung menggunakan ekspresi, seperti perbandingan aritmatika.
- (2) Sebuah objek yang tidak dianggap "kosong" dilewatkan. Berikut beberapa contoh objek yang dianggap kosong: 1. String kosong: "" 2. List kosong: [] 3. Angka nol: 0 4. Variabel boolean palsu: Salah.

h) Operator '='

Tidak seperti operator ganda sama dengan "==" , operator "adalah" tidak cocok dengan nilai variabel, tetapi contoh itu sendiri. Sebagai contoh:

```
x = [1,2,3]
y = [1,2,3]
print(x == y) # Prints out True
print(x is y) # Prints out False
```

i) Operator "bukan"

Menggunakan "tidak" sebelum ekspresi boolean akan membalikkannya:

```
print(not False) # Prints out True
print((not False) == (False)) # Prints out False
```

j) Loop

Untuk loop, lakukan iterasi pada urutan tertentu. Berikut ini contohnya:

```
primes = [2, 3, 5, 7]
```

```
for prime in primes:
```

```
    print(prime)
```

Untuk loop dapat mengulang urutan angka menggunakan fungsi "range" dan "xrange". Perbedaan antara range dan xrange adalah bahwa fungsi range mengembalikan list baru dengan angka dari kisaran yang ditentukan, sedangkan xrange mengembalikan iterator, yang lebih efisien. (Python 3 menggunakan fungsi range, yang bertindak seperti xrange). Perhatikan bahwa fungsi rentang berbasis nol.

Pemrograman dalam bahasa berorientasi objek biasanya berarti mengimplementasikan kelas yang menggambarkan objek yang menyimpan informasi yang dibutuhkan oleh program yang sedang menulis. Objek berisi data dan metode beroperasi pada data tersebut. Sebuah kelas adalah definisi data dan metode untuk jenis objek tertentu. Setiap kelas berisi satu metode khusus yang disebut konstruktor. Konstruktor tugasnya adalah membuat instance objek dengan menempatkan referensi ke data di dalam objek diri. Misalnya, pertimbangkan kelas yang disebut hewan sapi. Seekor sapi memiliki nama, tanggal lahir, dan suara yang dihasilkannya saat mengaum.

3) Pemanggilan Method

Python mendukung fungsi tradisional (lihat Bagian 1.5) yang dipanggil dengan pajak syn seperti diurutkan(data), dalam hal ini data adalah parameter yang dikirim ke fungsi. Kelas Python juga dapat mendefinisikan satu atau lebih metode (juga dikenal sebagai anggota fungsi), yang dipanggil pada instance tertentu dari kelas menggunakan titik (".") operator. Misalnya, kelas daftar Python memiliki metode bernama sort yang dapat dipanggil dengan sintaks seperti data.sort(). Metode khusus ini mengatur ulang isi daftar sehingga mereka diurutkan.

Ekspresi di sebelah kiri titik mengidentifikasi objek yang menjadi dasar metode dipanggil. Seringkali, ini akan menjadi pengidentifikasi (misalnya, data), tetapi kita dapat menggunakan operator titik untuk memanggil metode pada hasil langsung dari beberapa operasi lain. Untuk contoh, jika respons mengidentifikasi instance string, sintaks response.lower() .startswith(y) pertama-tama mengevaluasi metode panggilan, response.lower(), yang dengan sendirinya mengembalikan instance string baru, dan kemudian metode startwith(y) dipanggil pada string perantara itu.

Saat menggunakan metode kelas, penting untuk memahami perilakunya. Beberapa metode mengembalikan informasi tentang keadaan suatu objek, tetapi tidak berubah yang menyatakan. Ini dikenal sebagai pengakses. Metode lain, seperti metode sortir dari kelas daftar, lakukan perubahan status objek. Metode-metode ini dikenal sebagai mutator atau metode pembaruan

C. LATIHAN SOAL

1. Jelaskan menurut anda sejarah bahasa pemrograman Python!
2. Jelaskan alasan mengapa programmer untuk mempelajari bahasa Python!
3. Jelaskan 3 perbedaan Python dengan bahasa C!
4. Jelasakan alur penyimpanan file dalam bahasa Python!
5. Jelaskan cara memanggil method pada pemrograman 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). Pendekripsi Kantuk Secara Real Time Menggunakan Pustaka Opencv Dan Dlib Python. *Sainstech : Jurnal Penelitian Dan Pengkajian Sains Dan Teknologi*.

PERTEMUAN 3

TIPE DATA

A. TUJUAN PEMBELAJARAN

Setelah mempelajari materi ini diharapkan mahasiswa Mahasiswa mengetahui tentang Tipe data, ADT dan implementasi ADT pada Python.

B. URAIAN MATERI

1. Pengertian Tipe Data

Item data direpresentasikan dalam komputer sebagai urutan digit biner. Ini urutan bisa tampak sangat mirip tetapi memiliki arti yang berbeda sejak komputer dapat menyimpan dan memanipulasi jenis data yang berbeda. Misalnya, biner sequence 01001100110010110101110011011100 bisa berupa string karakter, nilai yang lebih kuat, atau nilai nyata. Untuk membedakan antara berbagai jenis data, istilah tipe data sering digunakan untuk merujuk pada kumpulan nilai dan istilah tipe data untuk merujuk ke tipe yang diberikan bersama dengan kumpulan operasi untuk memanipulasi nilai dari tipe yang diberikan.

Bahasa pemrograman biasanya menyediakan tipe data sebagai bagian dari bahasa diri. Tipe data ini, yang dikenal sebagai primitif, hadir dalam dua kategori: sederhana dan kompleks. Tipe data sederhana terdiri dari nilai yang paling banyak bentuk dasar dan tidak dapat diuraikan menjadi bagian-bagian yang lebih kecil. Bilangan bulat dan tipe nyata, misalnya, terdiri dari nilai numerik tunggal. Tipe data kompleks, disisi lain, dibangun dari beberapa komponen yang terdiri dari tipe sederhana atau tipe kompleks lainnya. Di Python, objek, string, list, dan kamus, yang bisa mengandung banyak nilai, semuanya adalah contoh tipe kompleks. Tipe primitif yang disediakan oleh bahasa mungkin tidak cukup untuk memecahkan masalah kompleks yang besar.

Dengan demikian, sebagian besar bahasa memungkinkan konstruksi tipe data tambahan, yang dikenal sebagai tipe yang ditentukan pengguna karena mereka didefinisikan oleh pemrogram, bukan bahasa. Beberapa dari tipe data ini sendiri bisa sangat kompleks.

2. Tipe data abstrak (ADT)

Tipe data abstrak adalah kelas/ tipe untuk objek yang perilakunya ditentukan oleh satu set nilai dan satu set operasi. Definisi ADT hanya menyebutkan operasi apa yang akan dilakukan, tetapi tidak menyebutkan bagaimana operasi ini akan dilakukan. Itu tidak menentukan bagaimana mengatur data dalam memori dan algoritma apa yang akan digunakan untuk mengimplementasikan operasi. Ini disebut "abstraksi" karena memberikan tampilan yang tidak bergantung pada implementasi. Proses memberikan hanya informasi dasar dan menyembunyikan detailnya disebut abstraksi.

Tipe data abstrak (ADT) adalah model matematika dari objek data yang meningkatkan tipe data dengan mengaitkan tipe data dengan fungsi yang beroperasi pada data terkait. Selain itu, penting untuk disadari bahwa operasi pada data objek yang bersangkutan termasuk dalam spesifikasi ADT. Misalnya, himpunan ADT didefinisikan sebagai kumpulan data yang diakses oleh operasi himpunan (seperti gabungan, perpotongan, dan perbedaan himpunan).

Penggunaan ADT harus menyediakan beberapa cara untuk merepresentasikan elemen dari tipe data (seperti matriks) dan cara untuk mengimplementasikan operasi matriks. Secara umum, akan digunakan beberapa algoritma untuk menggambarkan operasi ADT. Algoritma biasanya merupakan serangkaian instruksi yang digunakan untuk menentukan dengan tepat bagaimana komputer akan melakukan operasi. ADT adalah tipe data spesifik yang didefinisikan oleh programmer untuk memudahkan pemrograman dan beradaptasi dengan tipe data yang tidak secara spesifik disesuaikan dengan bahasa pemrograman yang digunakan. Oleh karena itu, untuk menyatakan ADT secara informal:

- a. Tipe data abstrak ADT pertama kali ditemukan oleh ilmuwan komputer dan digunakan untuk memisahkan struktur penyimpanan dari perilaku tipe data abstrak (seperti tumpukan dan antrian). Seperti yang kita harapkan, programmer tidak perlu tahu bagaimana mengubah stack ke implementasi ADT dan tidak mengubah program yang menggunakan secara keseluruhan., dengan catatan bahwa interface ADT tersebut dengan 'dunia luar' tetap dipertahankan.

- b. Pemakaian dan pembuatan ADT dapat dilakukan secara terpisah. yang perlu dibicarakan antara pembuat dan pengguna ADT adalah interface ADT yang bersangkutan.
- c. ADT adalah suatu alat dalam pengembangan sistem dimana memiliki sifat modular, yang dapat dimungkinkan suatu sistem dikembangkan beberapa orang anggota tim kerja dimana setiap anggota tim dapat melakukan bagian masing-masing secara tetap mempertahankan keterpaduan dengan anggota tim lainnya.

Dalam hal tersebut diperlukan untuk pembeda antara struktur data dengan ADT. Struktur data hanya melihatkan bagaimana data di organisir, sedang pada ADT memiliki cakupan yang lebihluas, yakni dengan memuat struktur data yang tertentu sekalian dengan operasi yang dibuat pada struktur data tersebut. Sehingga dapat didefinisikan secara umumnya ADT secara luas sebagai berikut:

3. Implementasi ADT

Bahasa pemrograman memiliki bentuk tipe data:

- a. Built-in Type

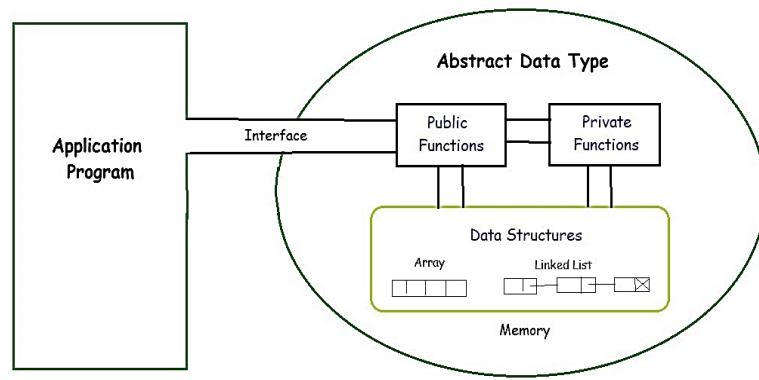
Built-in ini bermakna bahwa bahasa pemograman telah menyediakan tipe data dimana torientasi tidak pada permasalahan yang dihadapi.

- b. User Defined Type(UDT)

Pada UDT pemrogram telah melakuakan pembuatan tipe data dimana memiliki kedekatan dalam menyelesaikan masalah yang dihadapi, Mysalnya pada record Pascal, struct di bahasa C, class di bahasa Java.

- c. Abstract Data Type(ADT)

Konsep UDT dapat diperluas dengan menambah enkapsulasi diman memiliki isi sifat dan operasi yang dapat dibuat pada kelas tersebut

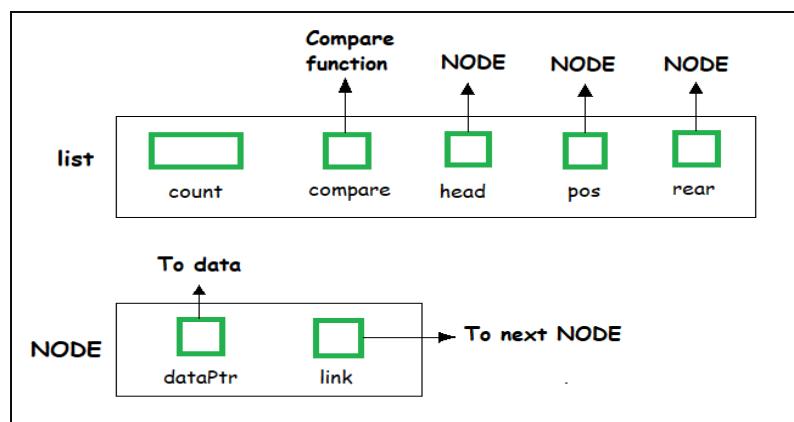


Gambar 3. 1 Implementasi Abstract Data Type

Pengguna tipe data tidak perlu mengetahui bagaimana tipe data tersebut diimplementasikan, misalnya kita telah menggunakan nilai-nilai Primitif seperti tipe data int, float, char hanya dengan pengetahuan bahwa tipe data ini dapat beroperasi dan dijalankan tanpa gagasan tentang bagaimana mereka diimplementasikan. Jadi pengguna hanya perlu tahu apa yang bisa dilakukan oleh tipe data, tetapi tidak bagaimana implementasinya. Pikirkan ADT sebagai kotak hitam yang menyembunyikan struktur dalam dan desain tipe data. Sekarang mengulang kembali pembahasan diatas mendefinisikan tiga ADT yaitu List ADT, Stack ADT, Queue ADT.

a. List ADT

Secara umum data tersimpan pada urutan kunci pada list dimana mempunyai bentuk head terdiri berupa nilai, pointer dan alamat fungsi dibuat pembanding sebagai pembanding data pada list.

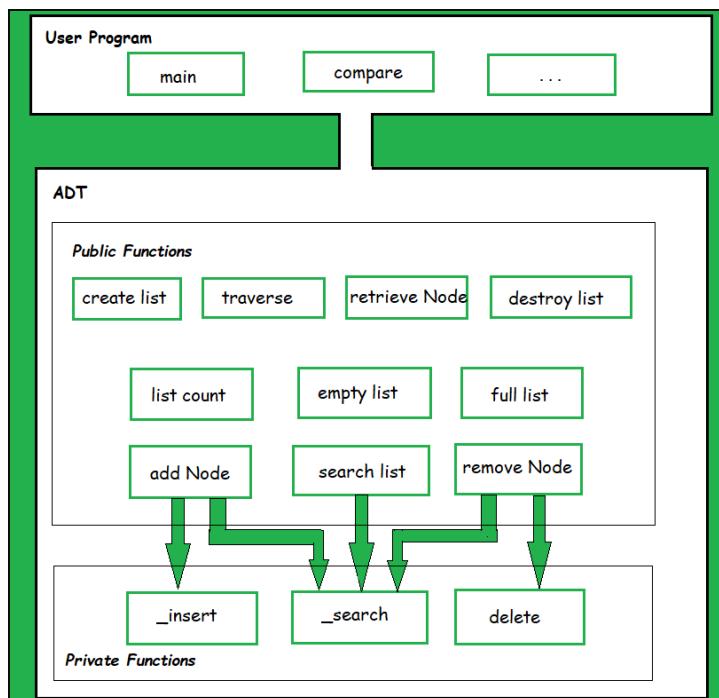


Gambar 3. 2 List ADT

Node data berisi pointer ke struktur data dan pointer self-referensial yang menunjuk ke node berikutnya dalam list .

```
//List Definisi Tipe ADT
simpul struktur typedef
{
    batal *DataPtr;
    struct simpul *tautan;
} simpul;
struktur type def
{
    jumlah int;
    simpul *pos;
    simpul *kepala;
    Simpul *belakang;
    int (*bandingkan) (batal *argumen1, batal *argumen2)
} LIST ;
```

List Fungsi ADT diberikan di bawah ini:



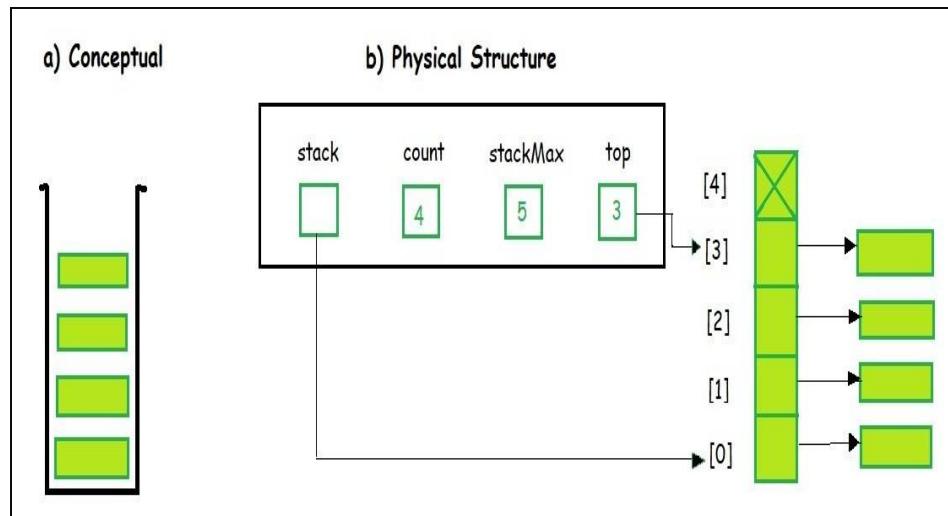
Gambar 3. 3 List Fungsi ADT

List berisi elemen dari jenis yang sama yang diatur dalam urutan berurutan dan operasi berikut dapat dilakukan pada list .

- 1) Get() – Mengembalikan elemen dari list pada posisi tertentu.
- 2) Insert() – Menyisipkan elemen pada posisi apapun dari list .
- 3) Remove() – Hapus kemunculan pertama elemen apa pun dari list yang tidak kosong.
- 4) Removeat() – Menghapus elemen di lokasi tertentu dari list yang tidak kosong.
- 5) Replace() – Mengganti elemen pada posisi apa pun dengan elemen lain.
- 6) Size() – Mengembalikan jumlah elemen dalam list .
- 7) Isempty() – Mengembalikan nilai true jika list kosong, jika tidak mengembalikan false.
- 8) Isfull() – Mengembalikan nilai true jika list penuh, jika tidak mengembalikan false.

b. Stack ADT

Dalam Implementasi Stack ADT alih-alih data disimpan di setiap node, penunjuk ke data disimpan. Program mengalokasikan memori untuk data dan alamat diteruskan ke stack ADT.



Gambar 3. 4 Stack ADT

Node kepala dan node data dienkapsulasi dalam ADT. Fungsi panggilan hanya dapat melihat penunjuk ke Stack an. Struktur kepala Stack an juga berisi penunjuk ke atas dan jumlah entri yang saat ini ada di Stack an.

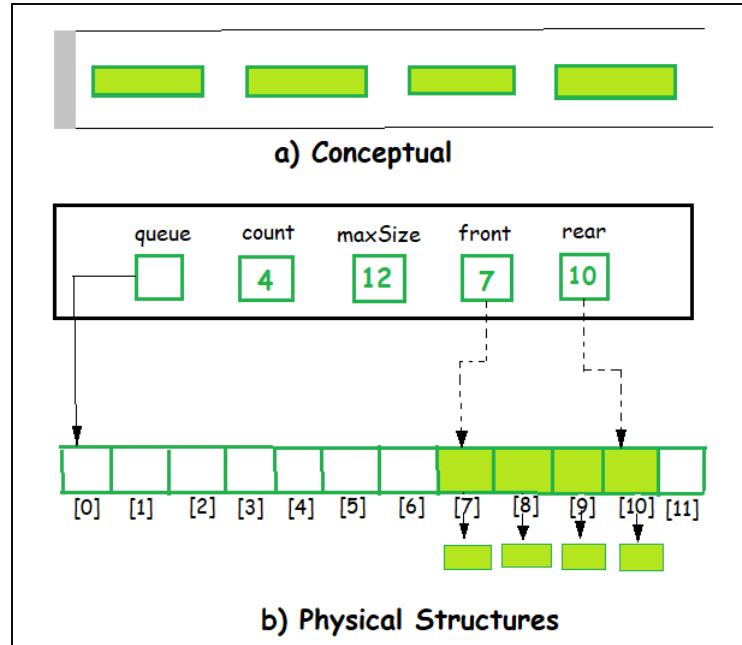
```
// Definisi Tipe ADT Stack
simpul struktur type def
{
    b a t a l *D a t a P t r;
    struct simpul *tautan;
} StackNode;
struktur type def
{
    j u m l a h i n t;
    StackNode *atas;
} S T A C K A N;
```

Stack berisi elemen dengan tipe yang sama yang disusun secara berurutan. Semua operasi berlangsung di satu ujung yang berada di atas Stack an dan operasi berikut dapat dilakukan:

- 1) push() – Masukkan elemen di salah satu ujung Stack an yang disebut top.
- 2) pop() – Hapus dan kembalikan elemen di bagian atas Stack an, jika tidak kosong.
- 3) peek() – Mengembalikan elemen di bagian atas Stack an tanpa menghapusnya, jika Stack an tidak kosong.
- 4) size() – Mengembalikan jumlah elemen dalam Stack an.
- 5) isEmpty() – Mengembalikan nilai true jika Stack an kosong, jika tidak mengembalikan false.
- 6) isFull() – Mengembalikan nilai true jika Stack an penuh, jika tidak mengembalikan false.

c. Queue ADT

Tipe data abstrak Queue (ADT) mengikuti desain dasar tipe data abstrak Stack an.

**Gambar 3. 5 Queue ADT**

Setiap node berisi pointer kosong ke data dan pointer link ke elemen berikutnya dalam Queue. Tanggung jawab program adalah mengalokasikan memori untuk menyimpan data.

```
//Definisi Jenis ADT Queue
simpul struktur typedef
{
    batal *DataPtr;
    struct simpul *berikutnya;
} Node Queue;
struktur typedef
{
    QueueNode *depan;
    QueueNode *belakang;
    jumlah int;
} ANTRIE;
```

Queue berisi elemen dari jenis yang sama diatur dalam urutan berurutan. Operasi berlangsung di kedua ujungnya, penyisipan dilakukan di ujung dan penghapusan dilakukan di depan. Operasi berikut dapat dilakukan:

- 1) Enqueue() – Menyisipkan elemen di akhir Queue.
- 2) Dequeue() – Hapus dan kembalikan elemen pertama dari Queue, jika Queue tidak kosong.

- 3) Peek() – Mengembalikan elemen Queue tanpa menghapusnya, jika Queue tidak kosong.
- 4) Size() – Mengembalikan jumlah elemen dalam Queue.
- 5) IsEmpty() – Mengembalikan nilai true jika Queue kosong

4. Jenis Struktur Data

Struktur data merupakan bagian dari bentuk suatu sistem program contohnya pada struktur maupun array dimana penerapan menginterpretasikan nilai data, hal tersebut dapat membuat operasi yang ada menjadi spesifik dapat dilakukan terhadap data pada program secara global jenis pada struktur data terbagi menjadi dua yakni Jenis data primitif diantaranya

- a. Integer
- b. Real
- c. Boolean
- d. Karakter
- e. String dimana tipe struktur data ini memiliki jenis data campuran

Penjelasan jenis data datas dibawah ini.

- a. Bilangan Bulat/Integer

(..... , -(n+1) , -n , , -2 , -1 , 0 , 1 , 2 , , n , n+1 ,)

Operasi mendasar terdapat pada interger diantaranya yaitu:

- 1) Jumlah
- 2) Kurang
- 3) Kali
- 4) Bagi
- 5) Pangkat

Data numerik selain Integer yang di kelompokkan dalam jenis data real dimana ditulis dengan titik atau koma pada desimal. Bilangan real tergolong pada memory komputer dimana menggunakan sistem floating point dimana versi itu sebagai Scientific Notation. Penyajian disini terdapat 2 yakni pecahan dan eksponen. Contoh pada interger dibawah ini.

Dalam sistem desimal, $456000 = 0.456 \times 10^6$

Dalam contoh tersebut 0.456 ialah pecahan sedang angka enam merupakan eksponen, pada umumnya suatu bilangan real X dapat ditulis dengan $M \times 10^E$ sebagai (M =Mentisa/Pecahan, E =real)

b. Boolean

Nama lain dari Boolean disebut dengan Logical. Unsur yang terdapat pada boolean memiliki satu dari True atau False. Operator yang dengan Boolean pada jenis data terdiri dari:

- 1) Operator OR sebagai penghasil TRUE, apabila satu atau keduanya memiliki nilai TRUE
- 2) Operator And sebagai penghasil TRUE, apabila AND dan OR memiliki FALSE atau kebalikannya
- 3) Operator NOT adalah Precedence yang berasal dari AND dan OR

Pada Suatu ekspresi dimana tidak menggunakan tanda kurung, operator NOT harus dievaluasi sebelum operator AND dan OR. Pada operator Relasional, yaitu : $>$, $<$, \geq , \leq , \neq dan $=$.

c. Karakter dan String

Jenis data karakter merupakan elemen dari suatu himpunan yang terdiri atas bilangan, abjad dan simbol-simbol khusus. Sedangkan jenis data string merupakan jenis data campuran, karena elemen-elemennya dibentuk dari karakter-karakter di atas. Karakter yang digunakan untuk membentuk suatu string disebut sebagai alphabet. Dalam penulisannya, suatu string berada dalam tanda "aphostrophe".

Contoh :

Terdapat suatu himpunan alphabet $A = \{ A, B, 2 \}$.

String yang dapat dibentuk dari alphabet tersebut yaitu 'AB2', 'ABB', 'BBA', 'ABA2', dan lainnya, termasuk "null string" atau "empty string". Himpunan yang memiliki anggota dari keseluruhan string dimana berasal dari himpunan alphabet yang dikenal dengan istilah "Vocabulary". Suatu Vocabulary V dimana menghasilkan himpunan alphabet A dinotasi dengan V_A atau A^* , apabila string juga terbentuk dari alphabet tersebut.

5. Memilih Struktur Data

Implementasi tipe data abstrak yang kompleks biasanya membutuhkan penggunaan struktur data untuk mengatur dan mengelola pengumpulan item data. Ada banyak struktur yang berbeda untuk dipilih. Jadi bagaimana kita tahu harus ke mana menggunakan? Kita harus mengevaluasi kesesuaian struktur data untuk mengimplementasikan diberi tipe data abstrak, yang kami berdasarkan pada kriteria berikut:

- a. Apakah struktur data menyediakan persyaratan penyimpanan seperti yang ditentukan oleh domain dari ADT(Abstract Data Type)? Tipe data abstrak didefinisikan untuk bekerja dengan sebuah domain spesifik dari nilai data. Struktur data yang kita pilih harus mampu menyimpan semua kemungkinan nilai dalam domain itu, dengan mempertimbangkan semua batasan atau batasan yang ditempatkan pada masing-masing item.
- b. Apakah struktur data menyediakan akses dan manipulasi data yang diperlukan fungsionalitas untuk sepenuhnya menerapkan ADT(Abstract Data Type)? Fungsionalitas abstrak tipe data disediakan melalui rangkaian operasi yang ditentukan. Struktur datanya harus memungkinkan implementasi ADT(Abstract Data Type) secara penuh dan benar tanpa harus untuk melanggar prinsip abstraksi dengan mengekspos detail implementasi ke pengguna.
- c. Apakah struktur data cocok untuk implementasi operasi yang efisien asi? Tujuan penting dalam implementasi tipe data abstrak adalah untuk memberikan solusi yang efisien. Beberapa struktur data memungkinkan lebih banyak implementasi yang efisien daripada yang lain, tetapi tidak setiap struktur data cocok untuk menerapkan setiap ADT. Pertimbangan efisiensi dapat membantu memilih yang terbaik struktur dari beberapa kandidat.

Mungkin ada beberapa struktur data yang cocok untuk mengimplementasikan tipe abstrak, tetapi kami berusaha untuk memilih yang terbaik berdasarkan konteksnya di mana ADT akan digunakan. Untuk mengakomodasi konteks yang berbeda, bahasa perpustakaan biasanya akan menyediakan beberapa implementasi dari beberapa ADT, memungkinkan programmer untuk memilih yang paling tepat. Mengikuti pendekatan ini, kami memperkenalkan sejumlah tipe data abstrak di seluruh teks dan menyajikan beberapa implementasi sebagai struktur data baru diperkenalkan.

Efisiensi implementasi didasarkan pada analisis kompleksitas. Jadi, kami menunda pertimbangan efisiensi implementasi dalam pemilihan struktur data hingga saat itu. Sementara itu, kami hanya mempertimbangkan kesesuaian struktur data berdasarkan penyimpanan dan persyaratan fungsional dari tipe data abstrak. Kami sekarang mengalihkan perhatian kami untuk memilih struktur data untuk menerapkan file bag ADT. Kandidat yang mungkin pada saat ini termasuk list dan kamus struktur. List tersebut dapat menyimpan semua jenis objek yang sebanding, termasuk duplikat. Setiap item disimpan satu per satu, termasuk duplikatnya, yang artinya referensi untuk setiap objek individu disimpan dan kemudian dapat diakses bila diperlukan. Ini persyaratan penyimpanan ADT Bag, membuat list struktur kandidat untuk implementasinya.

Kamus menyimpan pasangan kunci atau nilai di mana komponen kunci harus berada sebanding dan unik. Untuk menggunakan kamus dalam mengimplementasikan ADT Bag, kami harus memiliki cara untuk menyimpan barang duplikat seperti yang dipersyaratkan oleh definisi tipe data abstract. Untuk mencapai ini, setiap item unik dapat disimpan di kunci bagian dari pasangan kunci / nilai dan penghitung dapat disimpan di bagian nilai. Itu counter akan digunakan untuk menunjukkan jumlah kemunculan yang sesuai barang di dalam bag. Ketika item duplikat ditambahkan, penghitung bertambah; kapan duplikat dihapus, penghitung dikurangi.

Baik struktur list maupun kamus digunakan untuk mengimplementasikan Bag ADT. Untuk versi bag yang sederhana, bagaimanapun, list adalah pilihan yang lebih baik sejak saat itu kamus akan membutuhkan ruang dua kali lebih banyak untuk menyimpan isi bag dalam kasus di mana sebagian besar itemnya unik. Kamus itu luar biasa pilihan untuk implementasi variasi bag hitung dari ADT.

Setelah memilih list, kita harus memastikan list tersebut menyediakan sarana untuk mengimplementasikan set lengkap operasi tas. Saat mengimplementasikan ADT, kita harus menggunakan fungsionalitas yang disediakan oleh struktur data yang mendasarinya. Terkadang, opsi ADT identik dengan yang telah disediakan oleh struktur data. Dalam hal ini, file implementasi bisa sangat sederhana dan dapat terdiri dari satu panggilan ke operasi sponding struktur, sementara dalam kasus lain, penggunaan banyak operasi yang disediakan oleh struktur. Untuk membantu memverifikasi

implementasi yang benar dari Bag ADT menggunakan list, dapat menjelaskan bagaimana setiap pengoperasian bag nantinya dilaksanakan:

Bag kosong dapat diwakili oleh list kosong.

- a. Ukuran bag bisa ditentukan oleh ukuran list.
- b. Menentukan apakah bag berisi barang tertentu dapat dilakukan dengan ekuivalen operasi list. ^ Ketika item baru ditambahkan ke bag, itu dapat ditambahkan ke akhir after karena tidak ada pemesanan khusus barang di dalam bag.
- c. Menghapus item dari bag juga dapat ditangani dengan list yang setara operasi.
- d. Item dalam list dapat dilintasi menggunakan for loop dan Python menyediakannya iterator yang ditentukan pengguna yang digunakan dengan bag.

Dari list yang diperinci ini, terlihat bahwa setiap operasi bag ADT dapat diimplementasikan disebutkan menggunakan fungsionalitas list yang tersedia. Jadi, list cocok untuk menerapkan bag.

C. LATIHAN SOAL

1. Jelaskan menurut anda tentang tipe data dan manfaatnya!
2. Jelaskan dan sebutkan jenis mengenai ADT yang anda ketahui!
3. Bagaimana memilih tipe data yang sesuai dalam struktur data?
4. Jelaskan proses operasi yang ada dalam ADT stack!
5. Jelaskan dan sebutkan proses operasi yang ada pada ADT queue!

D. REFERENSI

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

Eriana, E. S. (2020). Pemilihan Ketua Himatif Universitas Pamulang Dengan Metode Simple Additive Weighting (Saw). *Jik(Jurnal Ilmu Komputer)*.

- Jodi, U. R. (2020). Algoritma Dan Struktur Data.
- 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.
- 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). Pendekripsi Kantuk Secara Real Time Menggunakan Pustaka Opencv Dan Dlib Python. *Sainstech : Jurnal Penelitian Dan Pengkajian Sains Dan Teknologi*.

PERTEMUAN 4

IF-ELSE

A. TUJUAN PEMBELAJARAN

Setelah mempelajari materi ini diharapkan mahasiswa mengerti penggunaan If-Else, Elif, If bersarang dan mampu membuat If dalam pemrograman Python.

B. URAIAN MATERI

1. Pernyataan IF

Secara detail terlihat pada pernyataan if Python dan penggunaannya dalam menerapkan pola pemrograman seperti satu arah, dua arah, dan multi arah keputusan. Dalam pertemuan ini membahas bagaimana struktur kontrol dengan sebuah tampilan rinci pada loop dan ekspresi Boolean. Pernyataan Python If menyediakan semacam perulangan, dimana memungkinkan untuk mengulang melalui urutan nilai.

Variabel indeks loop "var" mengambil setiap nilai yang berurutan dalam urutan, dan pernyataan di badan perulangan dieksekusi sekali untuk setiap nilai. Misalkan Apabila ingin menulis sebuah program yang dapat menghitung rata-rata suatu rangkaian nomor yang dimasukkan oleh pengguna. Untuk membuat program umum, itu harus bekerja untuk berbagai ukuran angka, bahwa rata-rata dihitung dengan menjumlahkan naikkan angka dan bagi dengan hitungan berapa banyak angka yang ada, yang demikian tidak perlu melacak semua nomor yang telah dimasukkan, hanya butuh jumlah berjalan sehingga dapat menghitung rata-rata di akhir.

Deskripsi masalah disarankan penggunaan beberapa pola desain yang telah terlihat sebelumnya. Dimana akan berurusan dengan serangkaian angka yang akan ditangani oleh beberapa bentuk loop. Jika ada angka, loop harus dieksekusi dan jika kali dapat menggunakan loop yang dihitung pola dan juga membutuhkan sejumlah akumulator loop. Pola dua ide bersama-sama yang dapat menghasilkan desain untuk masalah ini.

Python menyediakan metode bagaimana menyelesaikan algoritma melalui kondisi kondisional: if, if-else, elif, dan nested ifs. Di bawah ini, kami akan membahas kondisi tersebut secara lebih rinci. Mirip dengan bahasa pemrograman lain, Python juga menggunakan pernyataan if. Pernyataan ini berisi ekspresi logis yang menggunakan data yang dibandingkan dan membuat keputusan berdasarkan perbandingan.

Semua pernyataan diindentasi setelah bagian kondisional. Python menggunakan lekukan untuk mengelompokkan satu atau lebih pernyataan. Dalam pernyataan if, kondisi akan dieksekusi terlebih dahulu. Apabila kondisinya benar, pernyataan di blok true_statement akan dieksekusi. Pernyataan else biasanya digabung dengan pernyataan if di atas. Pernyataan else dapat berisi satu atau lebih blok pernyataan, yang akan dieksekusi jika kondisinya salah (tidak valid). Aturan sintaks pernyataan if sebagai berikut:

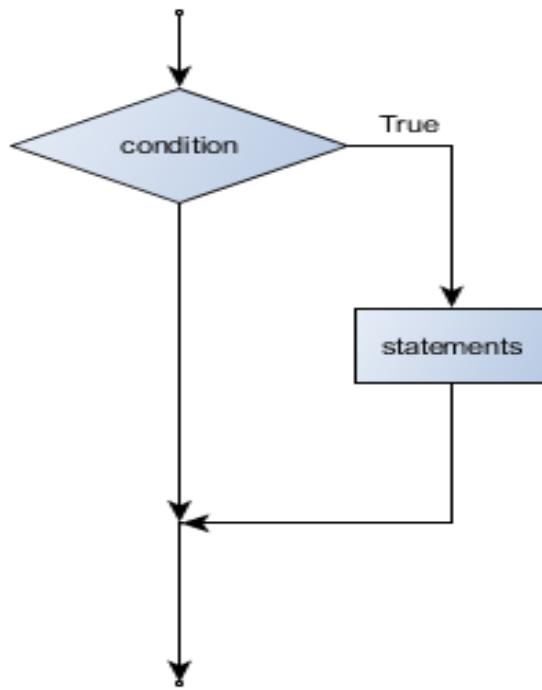
```
if <expr>:  
    <statement>
```

Makna dari bentuk if di atas:

- <expr> Adalah ekspresi yang dievaluasi dalam konteks Boolean, seperti yang dibahas di bagian tentang Operator Logis dalam tutorial Operator dan Ekspresi dalam Python.
- <statement> Adalah pernyataan Python yang valid, yang harus indentasi.

Jika benar (mengevaluasi nilai yang "kebenaran"), maka dieksekusi. Jika salah, maka dilewati dan tidak dieksekusi.<expr><statement><expr><statement>

Perhatikan bahwa colon () berikut diperlukan. Beberapa bahasa pemrograman perlu dilampirkan dalam tanda kurung, tetapi Python tidak.:<expr><expr>. Berikut adalah Diagram Alur dari pernyataan jika:



Gambar 4. 1 Diagram Alur IF

Seluruh pernyataan diindentasi setelah bagian kondisional. Python menggunakan lekukan untuk mengelompokkan satu atau lebih pernyataan. Dalam pernyataan if, kondisi akan dieksekusi terlebih dahulu. Jika kondisinya benar, pernyataan di blok true_statement akan dieksekusi. Sebagai contoh:

```
1  >>> x = 10
2  >>> if x>0:
3      ...     print 'Hello'
```

Pada contoh di atas, pernyataan if akan menampilkan teks 'Hello' jika nilai variabel x lebih besar dari 0.

```

1 x = 1
2 y = 5
3
4 if x < y:                                     # Truthy
5 print('yes')
6 #maka akan yes
7
8 if y < x:                                     # Falsy
9 print('yes')
10
11 if x:                                         # Falsy
12 print('yes')
13
14 #maka akan yes
15
16 if y:                                         # Truthy
17 print('yes')
18
19 #maka akan yes
20
21 if x or y:                                    # Truthy
22 print('yes')
23
24 #maka akan yes
25
26 if x and y:                                   # Falsy
27 print('yes')
28 if 'aul' in 'grault':                         # Truthy
29 print('yes')
30
31 #maka akan yes
32
33 if 'quux' in ['foo', 'bar', 'baz']: # Falsy
34 print('yes')
35

```

Catatan: Jika mencoba contoh-contoh ini secara interaktif akan menemukan bahwa, ketika menekan Enter setelah mengetikkan pernyataan, tidak ada yang terjadi.print('yes'). Karena ini adalah pernyataan multiline, dimana harus menekan Enter untuk kedua kalinya untuk memberi tahu interpreter bahwa sudah selesai. Newline tambahan ini tidak diperlukan dalam kode yang dijalankan dari file skrip.

2. Pernyataan Else

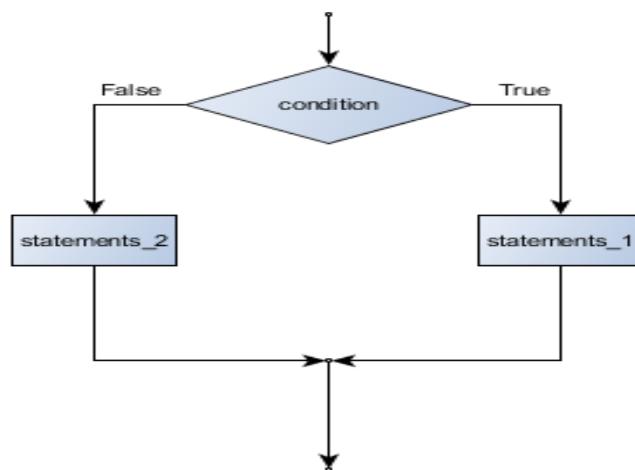
Pernyataan else biasanya digabung dengan pernyataan if di atas. Pernyataan else dapat berisi satu atau lebih blok pernyataan, dimana akan dieksekusi jika kondisinya tidak valid. Hal ini sering terjadi bahwa sesuatu terjadi ketika kondisi itu benar, dan sesuatu lain terjadi ketika itu salah. Untuk itu kami memiliki pernyataan if else

```

algoritma=1
if algoritma =='Struktur Data':
    print(' Belajar Algoritma dan Struktur Data!')
else:
    print(" Belum belajar")

```

Di sini, pernyataan cetak pertama akan dieksekusi jika sama dengan , dan pernyataan cetak yang indentasi di bawah klausa akan dieksekusi ketika tidak. Berikut adalah Diagram Alur dari pernyataan jika lain



Gambar 4. 2 Diagram Alur If-Else

Sintaks untuk pernyataan terlihat seperti ini:if else

if BOOLEAN EXPRESSION:

 STATEMENTS_1 # eksekusi If jika bernilai benar

else:

 STATEMENTS_2 # eksekusi If jika bernilai salah

Setiap pernyataan di dalam blok pernyataan dieksekusi jika ekspresi boolean mengevaluasi seluruh blok pernyataan di bawah Statement dieksekusi. if if else True False else. Tidak ada batasan jumlah pernyataan yang dapat muncul di bawah dua Statement pernyataan, tetapi harus ada setidaknya satu pernyataan di setiap blok. Kadang-kadang, adalah berguna untuk memiliki bagian tanpa pernyataan (biasanya sebagai penjaga tempat, atau perancah, untuk kode yang belum ditulis). Dalam hal ini dapat menggunakan pernyataan, yang tidak melakukan apa pun kecuali bertindak sebagai placeholder. if else pass

```
if True:      # This is always true
    pass      # so this is always executed, but it does nothing
else:
    pass
```

Berikut syntax untuk if -else

```
1  if kondisi:
2  ... pernyataan_benar
3  ... else:
4  ... pernyataan salah
```

Sebagai contoh, untuk mengetahui suatu bilangan merupakan genap atau ganjil, dapat digunakan kode seperti berikut ini,

```
>>> if (bilangan % 2 == 0):
1   ... print 'genap'
2   ... else:
3   ...     print 'ganjil'
```

contoh di atas, genap akan menampilkan apabila sisa dari pembagian bilangan dengan 2 sama dengan 0. dan sebaliknya apabila tidak sama dengan 0, maka akan menampilkan ganjil.

3. Pernyataan Elif

Terkadang ada lebih dari dua kemungkinan dan kami membutuhkan lebih dari dua cabang. Salah satu cara untuk mengekspresikan perhitungan seperti itu adalah kondisional yang terkait.

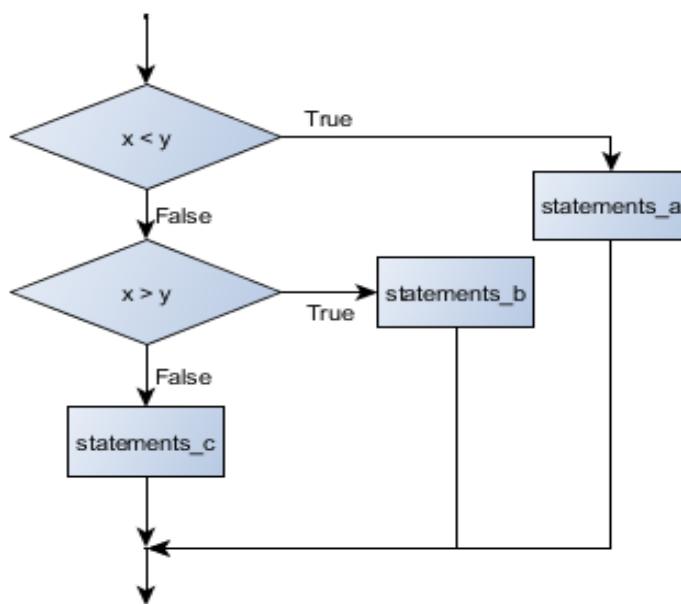
```

if x < y:
    STATEMENTS_A

elif x > y:
    STATEMENTS_B

else:
    STATEMENTS_C
  
```

Berikut adalah Flowchart dari rantai bersyarat



Gambar 4. 3 Diagram Alur Elif

elif Tidak ada batasan jumlah pernyataan tetapi hanya satu (dan opsional) pernyataan akhir diperbolehkan dan harus menjadi cabang terakhir dalam pernyataan:else if elif else

```
if choice == 'a':  
  
    print("You chose 'a'.")  
  
elif choice == 'b':  
  
    print("You chose 'b'.")  
  
elif choice == 'c':  
  
    print("You chose 'c'.")  
  
else:  
  
    print("Invalid choice.")
```

Setiap kondisi diperiksa secara berurutan. Jika yang pertama adalah palsu, yang berikutnya diperiksa, dan seterusnya. Jika salah satu dari mereka benar, cabang yang sesuai mengeksekusi, dan pernyataan berakhir. Bahkan jika lebih dari satu kondisi benar, hanya cabang sejati pertama yang mengeksekusi

Pernyataan elif memungkinkan untuk melakukan pengujian beberapa kondisi untuk setiap nilai dan mengeksekusi kode pernyataan dimana memenuhi kondisi yang diberlakukan. Berikut syntax elif, dibawah ini.

```
1  if kondisi_1:  
2      pernyataan_1_benar  
3  elif kondisi_2:  
4      pernyataan_2_benar  
5      :  
6      :  
7  elif kondisi_N:  
8      pernyataan_N_benar  
9  else:  
10     pernyataan_tidak_sesuai_syarat_di_atas
```

Contoh dibawah ini memakai pernyataan elif bertujuan menunjukkan apakah huruf yang kita masukkan adalah vokal. Apabila tidak, tampilan lain akan dimuncul.

```

1      huruf = raw_input("Masukkan sebuah huruf: ")
2
3      if (huruf == 'a'):
4          print "Ini adalah huruf vokal - a -"
5      elif (huruf == 'e'):
6          print "Ini adalah huruf vokal - e -"
7      elif (huruf == 'i'):
8          print "Ini adalah huruf vokal - i -"
9      elif (huruf == 'o'):
10         print "Ini adalah huruf vokal - o -"
11     elif (huruf == 'u'):
12         print "Ini adalah huruf vokal - u -"
13     else:
14         print "Ini bukan huruf vokal"

```

Pada kode diatas, ketika pengguna mengetik sebuah huruf, maka diproses terlepas apakah huruf tersebut merupakan huruf vokal atau bukan. Apabila pengguna mengetik huruf "a", "Ini adalah vokal -a-" akan ditampilkan sesuai dengan ketentuan yang berlaku. Jika kondisi tidak sesuai, misalnya pengguna memasukkan huruf "u", kondisi elif selanjutnya akan berlanjut hingga ditemukan kondisi yang sesuai, dan hasilnya akan ditampilkan sebagai "Ini adalah vokal -u-". Jika pengguna mengetikkan huruf "x", tidak ada kondisi yang cocok dengan kondisi if dan elif. Pernyataan else yang dieksekusi adalah "ini bukan vokal".

4. Pernyataan If Bersarang

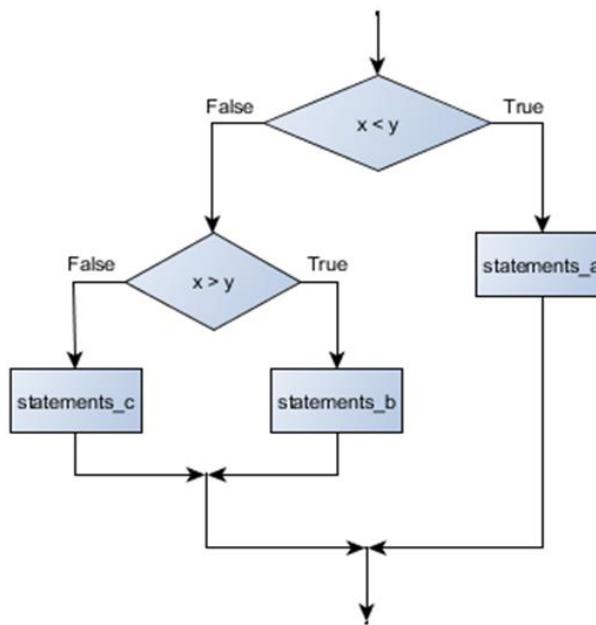
Satu kondisi dapat bersarang di dalam yang lain, contoh sebelumnya sebagai berikut:

```

if x < y:
    STATEMENTS_A
else:
    if x > y:
        STATEMENTS_B
    else:
        STATEMENTS_C

```

Berikut adalah Flowchart dari kondisional bersarang ini:



Gambar 4. 4 Diagram Alur If Bersarang

Kondisional luar berisi dua cabang. Cabang kedua berisi pernyataan lain, yang memiliki dua cabang sendiri. Kedua cabang itu juga bisa berisi pernyataan bersyarat if. Meskipun pernyataan membuat struktur jelas, kondisional bersarang sangat cepat menjadi sulit dibaca. Operator logis sering menyediakan cara untuk menyederhanakan pernyataan bersarang. Misalnya dapat menulis ulang kode berikut menggunakan satu bersyarat:

```

if 0 < x:      # assume x is an int here
    if x < 10:
        print("x is a positive single digit.")
  
```

Fungsi ini disebut hanya jika kita berhasil melewati kedua kondisional, sehingga dapat menggunakan operator:printand . Pernyataan Satu Barisif adalah kebiasaan untuk menulis pada satu baris dan indentasi pada baris berikut seperti ini: if <expr><statement>

```

if <expr>:
    <statement>
  
```

Tetapi diperbolehkan untuk menulis seluruh pernyataan pada satu baris. Berikut ini secara fungsional setara dengan contoh di atas:if

```
if <expr>: <statement>
```

Bahkan mungkin ada lebih dari satu pada baris yang sama, dipisahkan oleh titik koma:<statement>

```
if <expr>: <statement_1>; <statement_2>; ...; <statement_n>
```

Artinya ada dua interpretasi yang mungkin:

Jika benar, eksekusilah.<expr><statement_1>

Kemudian, eksekusi tanpa syarat, terlepas dari apakah itu benar atau tidak.<statement_2> ... <statement_n><expr>

Jika benar, jalankan semua. Jika tidak, jangan mengeksekusi salah satu dari mereka.<expr><statement_1> ... <statement_n>

Python mengambil interpretasi yang terakhir. Titik koma yang memisahkan memiliki prioritas yang lebih tinggi daripada titik kuut berikut dalam istilah komputer, titik koma dikatakan mengikat lebih erat daripada usus besar. Dengan demikian, yang diperlakukan sebagai suite, dan salah satu dari mereka dieksekusi, atau tidak satupun dari mereka adalah:

```
<statements><expr><statements>
```

```
if 'f' in 'foo': print('1'); print('2'); print('3')
```

```
1
```

```
2
```

```
3
```

```
if 'z' in 'foo': print('1'); print('2'); print('3')
```

Beberapa pernyataan dapat ditentukan pada baris yang sama dengan atau klausa juga:elif else,

```
1 x = 2
2 if x == 1: print('foo'); print('bar'); print('baz')
3 elif x == 2: print('qux'); print('quux')
4 else: print('corge'); print('grault')
5
```

Setelah dirun maka hasilnya

qux,

quux

```

1 x = 3
2 if x == 1: print('foo'); print('bar'); print('baz')
3 elif x == 2: print('qux'); print('quux')
4 else: print('corge'); print('grault')
5

```

Setelah dirun maka hasilnya

Corge

Grault

Sementara semua ini bekerja, dan interpreter menyetujui umumnya tidak disarankan karena hal itu mengarah pada readability yang buruk, terutama pernyataan yang kompleks. Seperti biasanya, kebanyakan orang akan menemukan hal-hal lebih menarik secara visual dan lebih mudah dipahami pada pandangan pertama daripada contoh di atas:

```

1 x = 3
2 if x == 1:
3     print('foo')
4     print('bar')
5     print('baz')
6 elif x == 2:
7     print('qux')
8     print('quux')
9 else:
10    print('corge')
11    print('grault')
12

```

Namun, jika sebuah pernyataan cukup sederhana, menempatkan semuanya pada satu baris mungkin masuk akal. Sesuatu seperti ini tidak akan meningkatkan terlalu banyak if. Terkadang menemukan bahwa menulis akan menempatkan blok kode yang belum terapkan. Dalam bahasa di mana delimiters digunakan untuk menentukan blok, seperti dalam pemrograman Perl and C dapat digunakan untuk menentukan code stub Misalnya, berikut ini adalah kode Perl atau C yang sah:

```
# This is not Python
```

```
if (x)
{
}
```

Pada suatu kondisi dimana diperlukan suatu if yang bersarang, maksud hal tersebut dapat dituliskan dengan if-else dalam if-else dibawah ini diberikan pencontohan dalam id bersarang pada program yang mana dapat menentukan huruf yang diinputkan User adalah huruf besar atau kecil.

```
1  huruf = raw_input("Masukkan sebuah huruf: ")
2
3  if (huruf >= 'A'):
4      if (huruf <= 'Z'):
5          print "Ini adalah Huruf Besar"
6      elif (huruf >= 'a'):
7          if (huruf <= 'z'):
8              print "Ini adalah huruf kecil"
9          else:
10             print "Huruf > z"
11     else:
12         print "Huruf > z tapi < a"
13 else:
14     print "Huruf < A"
```

Sebagai berikut hasilnya jika dipraktekan dalam python

```
4
5 a=input( "masukan a= ")
6 b=input( "masukan b= ")
7 c=input ("masukan c= ")
8
9 if a>b:
10    if a>c:
11        besar=a
12    else:
13        besar=c
14 elif b>c:
15    besar=b
16 else:
17    besar=c
18 print "bilangan yg terbesar adalah = ";
19 print besar
```

Jika di run maka hasilnya sebagai berikut.

```
masukan a= 12  
masukan b= 8  
masukan c= 15  
bilangan yg terbesar adalah =  
15
```

Dalam program Python bagaimana melakukan pengambilan keputusan dimana hal ini memungkinkan untuk eksekusi bersyarat dari pernyataan atau kelompok pernyataan berdasarkan nilai ekspresi.if. Alasan mempelajari if ialah:

- a. Hal pertama mendapatkan gambaran singkat tentang pernyataan dalam bentuknya yang paling sederhana.if
- b. Selanjutnya, dengan menggunakan pernyataan sebagai model, akan mengetahui alasan mengapa struktur kontrol memerlukan beberapa mekanisme untuk mengelompokkan pernyataan bersama-sama menjadi pernyataan majemuk atau blok.
- c. Terakhir adalah belajar lakukan pengkodean pengambilan keputusan yang kompleks.

Dengan menggunakan pernyataan bersyarat seperti pernyataan if dimana memiliki kontrol yang lebih besar atas apa yang dijalankan program. Pernyataan bersyarat memberitahu program untuk mengevaluasi apakah kondisi tertentu terpenuhi. Jika kondisi terpenuhi maka akan mengeksekusi kode tertentu, namun jika tidak terpenuhi program akan terus beralih ke kode lain if.

C. LATIHAN SOAL

1. Jelaskan menurut anda cara kerja pernyataan if dalam bahasa python!
2. Jelaskan menurut anda 2 perbedaan antara pernyataan if dengan if-else!
3. Bagaimana pendapat anda mengenai if-else majemuk?
4. Bagaimana pendapat anda elif dalam bahasa python?
5. Bagaimana pendapat anda mengenai pernyataan if bersarang?

D. REFERENSI

- Amanulhaq, A. A. (2021). Implementasi Algoritma Image Hashing Dan Hamming Distance Untuk Deteksi Kemiripan Gambar. *Repository Its*.
- Basant Agarwal, B. B. (2018). Hand-On Data Structures And Algorithms With Python. London: Packt Publishing.
- Emi Sita Eriana, A. Z. (2021). Praktikum Algoritma Dan Pemrograman. Tangerang Selatan: Unpam Press.
- Jodi, U. R. (2020). Algoritma Dan Struktur Data.
- 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.
- 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). Pendekripsi Kantuk Secara Real Time Menggunakan Pustaka Opencv Dan Dlib Python. Sainstech : Jurnal Penelitian Dan Pengkajian Sains Dan Teknologi.
- Zein, A. (2019). Pendekripsi Penyakit Malaria Menggunakan Medical Images Analisis Dengan Deep Learning Python. Jurnal Penelitian Dan Pengkajian Sains Dan Teknologi.

PERTEMUAN 5

ARRAY DAN LINKED LIST

A. TUJUAN PEMBELAJARAN

Setelah mempelajari materi ini diharapkan mahasiswa mengerti dan mampu menggunakan dan membuat program Array dan Linked List dengan Python

B. URAIAN MATERI

1. Array

Array adalah struktur penyimpanan data yang paling umum digunakan itu dibangun ke dalam sebagian besar bahasa pemrograman. Karena array sangat terkenal, mereka menawarkan lompatan yang nyaman tempat untuk memperkenalkan struktur data dan untuk melihat bagaimana pemrograman berorientasi objek dan struktur data berhubungan satu sama lain. Dalam bab ini kami akan memperkenalkan array di Java dan mendemonstrasikan kelas array buatan sendiri. Kami juga akan memeriksa jenis larik khusus, urutannya array, di mana data disimpan dalam ascending (atau descending) urutan kunci. Pengaturan ini memungkinkan cara yang cepat pencarian item data: pencarian biner. Struktur paling dasar dimana dapat menyimpan dan mengakses kumpulan data adalah larik. Array dapat digunakan untuk memecahkan berbagai macam masalah dalam ilmu komputer. Paling bahasa pemrograman menyediakan tipe data terstruktur ini sebagai primitif dan memungkinkan untuk pembuatan array dengan berbagai dimensi. Dalam bab ini, kami menerapkan struktur array untuk array satu dimensi dan kemudian menggunakan untuk mengimplementasikan sebuah array dua dimensi dan struktur matriks terkait.

Array merupakan suatu variabel yang khas dan memiliki kemampuan dalam memberi tempat yang dapat memberi tempungan nilai lebih dari satu dalam waktu bersamaan, apabila mempunya list item misalnya nama list sepeda motor, penyimpanan variabel biasa akan terdeklasi sebagai berikut:

```
Sepedamotor1="Ninja"
```

```
Sepedamotor1="Mio"
```

Sepedamotor1="Nmax"

Namun jika bagaimana jika menginginkan satu tipe diantara 3 sepeda motor diatas, dan bagaimana jika anda memiliki showroom sepeda motor dengan 500 jenis sepeda motor, dan ingin mencari satu tipe sepeda motor tertentu? Solusi permasalahan tersebut dengan array, karena array ini mampu menampung data yang berbeda dalam satu penamaan dan anda dimudahkan untuk dapat mengaksesnya dengan mengacu pada indeks, dalam array indeks merupakan tempat urutan data yang dimulai dengan 0,1,2 dan seterusnya sebanyak data yang akan disimpan.

2. Pengakses Elemen Array

Menrujuk elemen array melalui cara yakni mengacu pada penomoran indeksnya. Misalnya, dapatkan data dari jenis array pertama:

x = mobil [0]

misalkan Ubah nilai item array pertama:

mobil [0] = "Honda"

length Array

penggunaan metode len() bertujuan untuk dapat mengembalikan panjang array (jumlah elemen pada array).

Misal, Mengembalikan jumlah elemen pada larik mobil:

x = len (mobil)

Catatan: Panjang array selalu memiliki lebihan satu indeks dari array paling tinggi

a. Elemen Array Pendauran

Penggunaan for in loop untuk melakukan loop melalui semua elemen array.

Misal,

Cetak setiap item di deretan mobil:

untuk x di mobil:

cetak (x)

b. Menambahkan Elemen Array

Penggunaan metode append () dapat melaksanakan perintah tambah pada elemen ke array.

Misal,

Tambahkan satu elemen lagi ke deretan mobil:

```
cars.append ("Honda")
```

c. Menghapus Elemen Array

Penggunaan metode pop () dapat melaksanakan perintah delete elemen dari larik. Misal,

Menghapus elemen ke-2 dari deretan mobil:

```
mobil.pop (1)
```

Bisa menggunakan metode remove () untuk melaksanakan perintah delete elemen dari array. Misal,

Menghapus elemen dimana nama yang bernilai "Volvo":

```
cars.remove ("Volvo")
```

Catatan: Metode remove () dari list hanya menghapus kemunculan pertama dari nilai yang ditentukan.

d. Metode Array

Python memiliki seperangkat metode bawaan yang dapat Anda gunakan pada list / larik. Pendeskripsi Metode

append () Menambah elemen pada akhir list

clear () Menghapus seluruh elemen dalam list

copy () Melakukan pengembalian salinan list

count () Melakukan pengembalian jumlah elemen dengan nilai yang ditentukan

index () Melakukan pengembalian elemen pertama dengan nilai yang ditentukan

insert () Melakukan penambahan elemen pada posisi yang ditentukan

pop () Melakukan penghapusan elemen pada posisi yang ditentukan

remove () Menghapus item pertama dengan nilai yang ditentukan

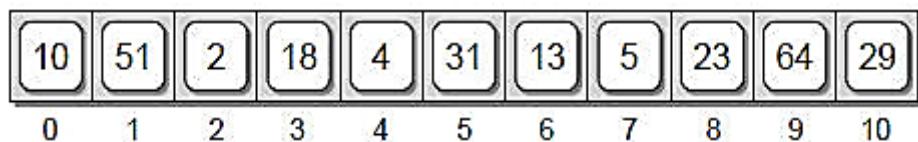
reverse () Membalik urutan list

sort () Melakukan pengurutan list

Catatan: Python tidak mempunyai support bawaan terhadap Array, akan tetapi List Python bisa diterapkan dalam penggunaannya sebagai penggantinya.

3. Struktur Array

Di tingkat perangkat keras, sebagian besar arsitektur komputer menyediakan mekanisme untuk ing dan menggunakan array satu dimensi. Array satu dimensi, seperti yang diilustrasikan pada terdiri dari beberapa elemen berurutan yang disimpan secara berdekatan byte memori dan memungkinkan akses acak ke elemen individu. Seluruh konten array diidentifikasi dengan satu nama. Individu elemen dalam array dapat diakses secara langsung dengan menentukan subskrip integer atau nilai indeks, yang menunjukkan posisi dari awal larik. Ini serupa ke notasi matematika (x_i), yang memungkinkan beberapa variabel yang sama nama. Perbedaannya adalah bahasa pemrograman biasanya menggunakan tanda kurung siku mengikuti nama array untuk menentukan subskrip, $x[i]$.



Gambar 3.1 Contoh array 1-D yang terdiri dari 11 elemen.

Jika akan melihat struktur array terlihat sangat mirip dengan struktur list Python. Itu karena kedua struktur tersebut sama-sama merupakan urutan yang tersusun dari banyak elemen sekuensial yang dapat diakses berdasarkan posisi. Tapi ada dua jurusan perbedaan antara larik dan list. Pertama, array memiliki jumlah yang terbatas operasi, yang biasanya mencakup pembuatan larik, membaca nilai dari elemen tertentu, dan menulis nilai ke elemen tertentu. Listnya, di sisi lain tangan, menyediakan sejumlah besar operasi untuk bekerja dengan konten file list. Kedua, list dapat bertambah dan menyusut selama eksekusi saat elemen ditambahkan atau dihapus saat ukuran array tidak dapat diubah setelah dibuat.

Mungkin bertanya-tanya, jika Python menyediakan struktur list sebagai struktur yang bisa berubah jenis urutan, mengapa kita repot-repot membahas struktur array, apalagi rencana mengimplementasikan tipe data abstrak untuk bekerja dengan array dengan Python? Yang pendek jawabannya adalah bahwa kedua struktur memiliki kegunaannya masing-masing. Ada banyak masalah itu hanya membutuhkan penggunaan array dasar yang jumlah elemennya diketahui

Sebelumnya dan rangkaian operasi yang fleksibel yang tersedia dengan list tidak diperlukan. Array paling cocok untuk masalah yang membutuhkan urutan di mana jumlah elemen ibu diketahui di depan, sedangkan list adalah pilihan yang lebih baik ketika ukuran urutan perlu diubah setelah dibuat. Seperti akan belajar nanti di bab ini, list berisi lebih banyak ruang penyimpanan daripada yang dibutuhkan simpan item yang saat ini ada dalam list. Ruang ekstra ini, yang ukurannya bisa sampai hingga dua kali lipat kapasitas yang diperlukan, memungkinkan perluasan yang cepat dan mudah sebagai item baru ditambahkan ke list. Tetapi ruang ekstra akan sia-sia saat menggunakan list untuk disimpan sejumlah elemen yang tetap. Misalnya, kita membutuhkan struktur urutan dengan 100; 000 elemen. Kita bisa membuat list dengan sejumlah elemen menggunakan operator replikasi:

```
nilai = [Tidak ada] * 100000
```

Namun di bawahnya, ini menghasilkan alokasi ruang hingga 200.000 elemen, setengahnya akan sia-sia. Dalam hal ini, array akan menjadi pilihan yang lebih baik.

Dalam mengambil suatu Keputusan, apakah sebuah larik atau list harus digunakan atau tidak, hal tersebut terbatas pada ukuran struktur urutan. Hal tersebut juga tergantung pada bagaimana akan digunakan. Itu list menyediakan sekumpulan besar operasi untuk mengelola item yang terdapat dalam list. Beberapa di antaranya termasuk memasukkan item di lokasi tertentu, mencari file item, menghapus item berdasarkan nilai atau lokasi, dengan mudah mengekstrak subset item, dan menyortir barang. Struktur array, di sisi lain, hanya menyediakan file serangkaian operasi terbatas untuk mengakses elemen individu yang terdiri dari larik. Jadi, jika masalah yang dihadapi membutuhkan jenis operasi ini, listnya lebih baik pilihan.

Array satu dimensi adalah kumpulan elemen yang bersebelahan di mana individu elemen diidentifikasi oleh subskrip integer unik yang dimulai dengan nol. Sekali array dibuat, ukurannya tidak dapat diubah.

- a. `Array (size)`: Membuat array satu dimensi yang terdiri dari elemen ukuran dengan setiap elemen awalnya disetel ke Tidak Ada. ukuran harus lebih besar dari nol.
- b. `length ()`: Mengembalikan panjang atau jumlah elemen dalam larik.
- c. `getitem (index)`: Mengembalikan nilai yang disimpan dalam array pada posisi elemen indeks. Argumen indeks harus berada dalam kisaran yang valid. Diakses menggunakan operator subskrip.
- d. `setitem (index, value)`: Memodifikasi konten elemen array di posisi index mengandung nilai. Indeks harus berada dalam kisaran yang valid. Diakses menggunakan operator subskrip.
- e. `clearing (nilai)`: Menghapus array dengan mengatur setiap elemen ke nilai.
- f. `iterator ()`: Membuat dan mengembalikan iterator yang bisa digunakan untuk melintasi elemen array.

Beberapa ilmuwan komputer menganggap array sebagai struktur fisik dan bukan abstraksi sejak array diimplementasikan pada tingkat perangkat keras. Tapi ingat, hanya ada tiga operasi dasar yang tersedia dengan perangkat keras yang diimplementasikan Array. Sebagai bagian dari Array ADT (abstract Data Type), maka telah menyediakan operasi ini tetapi telah juga menyertakan iterator dan operasi untuk mendapatkan ukuran array dan untuk mengatur setiap elemen ke nilai tertentu. Dalam hal ini, kami telah memberikan level yang lebih tinggi abstraksi daripada yang disediakan oleh larik yang diimplementasikan perangkat keras yang mendasarinya. Program sederhana berikut mengilustrasikan pembuatan dan penggunaan objek array berdasarkan Array ADT. Komentar disediakan untuk menyoroti penggunaan metode operator.

4. Implementasi Array

Python adalah bahasa skrip yang dibuat menggunakan bahasa C, bahasa tingkat tinggi yang membutuhkan kode sumber program dikompilasi menjadi kode yang dapat dieksekusi sebelum dapat digunakan. Bahasa C merupakan bahasa pemrograman dimana sangat kuat dapat menyediakan sintaks untuk bekerja dengan fungsionalitas lengkap yang tersedia oleh yang mendasarinya perangkat keras. Sintaks itu, bagaimanapun, bisa agak samar dibandingkan dengan Python, terutama untuk programmer Python yang mungkin belum familiar dengan C.

Banyak tipe data dan kelas yang tersedia dengan Python sebenarnya diimplementasikan menggunakan tipe yang sesuai dari bahasa C. Sedangkan Python tidak menyediakan file struktur array sebagai bagian dari bahasa itu sendiri, sekarang termasuk modul tipe seperti bagian dari Pustaka Standar Python. Modul ini menyediakan akses ke rangkaian yang beragam tipe data yang tersedia dalam bahasa C dan fungsionalitas lengkap yang disediakan oleh berbagai perpustakaan C. Modul `ctypes` menyediakan kemampuan untuk membuat ar- yang didukung perangkat keras. sinar seperti yang digunakan untuk mengimplementasikan string, list, tuple, dan kamus Python jenis koleksi. Tetapi modul `ctypes` tidak dimaksudkan untuk penggunaan sehari-hari dengan Python program seperti yang dirancang untuk digunakan oleh pengembang modul untuk membantu menciptakan lebih banyak modul Python portabel dengan menjembatani kesenjangan antara Python dan bahasa C.

Sebagian besar fungsionalitas yang disediakan oleh modul `ctypes` membutuhkan pengetahuan dari bahasa C. Dengan demikian, teknik yang diberikan oleh modul untuk membuat file array biasanya tidak boleh digunakan secara langsung dalam program Python. Tapi kita bisa menggunakannya dalam kelas `Array` kami untuk menyediakan fungsionalitas yang didefinisikan oleh `Array` ADT karena detailnya akan disembunyikan di dalam kelas.

Array suatu wadah yang dapat menampung sejumlah item dan item ini harus dari jenis yang sama. Sebagian besar struktur data menggunakan array untuk mengimplementasikan algoritmanya. Berikut ini adalah istilah-istilah penting untuk memahami konsep `Array`. Element setiap item yang disimpan dalam array disebut `elemen`. Indeks Setiap lokasi elemen dalam array memiliki indeks numerik, yang digunakan untuk mengidentifikasi elemen. Sesuai ilustrasi

di atas, berikut ini adalah poin-poin penting yang harus diperhatikan. Berikut ini adalah operasi dasar yang didukung oleh array.

- a. Traverse mencetak semua elemen array satu per satu.
- b. Penyisipan Menambahkan elemen pada indeks yang diberikan.
- c. Penghapusan Menghapus elemen pada indeks yang diberikan.
- d. Pencarian Mencari elemen menggunakan indeks yang diberikan atau berdasarkan nilai.
- e. Perbarui Memperbarui elemen pada indeks yang diberikan.

Array dibuat dengan Python dengan mengimpor modul array ke program python. Kemudian array dideklarasikan seperti gambar dibawah.

```
from array import *
arrayName = array(typecode, [Initializers])
```

Typecode adalah kode yang digunakan untuk menentukan tipe nilai yang akan disimpan oleh array. Beberapa typecode yang umum digunakan sebagai berikut.

Tabel 5.1 Typecode Array

Type code	Nilai
b	Mewakili bilangan bulat bertanda dengan ukuran 1 byte/td>
B	Mewakili unsigned integer ukuran 1 byte
c	Mewakili karakter berukuran 1 byte
i	Mewakili bilangan bulat yang ditandatangani dengan ukuran 2 byte

I	Mewakili bilangan bulat yang tidak ditandatangani dengan ukuran 2 byte
f	Mewakili floating point ukuran 4 byte
d	Mewakili floating point ukuran 8 byte

Sebelum melihat berbagai operasi array, mari buat dan cetak array menggunakan python. Kode di bawah ini membuat array yang disebut array 1

```
from array import *
array1 = array('i', [10,20,30,40,50])
for x in array1:
    print(x)
```

Ketika mengkompilasi dan mengeksekusi program di atas, menghasilkan hasil berikut yang menunjukkan elemen dimasukkan pada posisi indeks 1.

```
10
60
20
30
40
50
```

5. Tipe Data Abstrak Array

Struktur array umumnya ditemukan di sebagian besar bahasa pemrograman sebagai primitive type, tetapi Python hanya menyediakan struktur list untuk membuat sequences. Kita dapat mendefinisikan ADT Array untuk mewakili array satu dimensi digunakan dengan Python yang bekerja mirip dengan array yang ditemukan dalam bahasa lain. Boleh jadi digunakan di seluruh teks saat struktur array diperlukan.

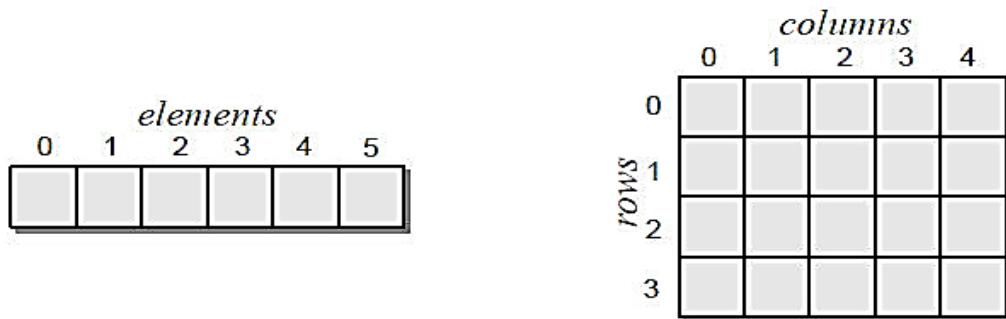
Array dua dimensi yakni suatu kumpulan elemen yang dapat dilakukan penyimpanan secara terurut di suatu tempat yang terbagi menjadi dua unsur yakni baris dan kolom. Array dua dimensi memiliki artian yakni dua dimensi data. Contoh sederhana adalah data yang tertulis dalam gambar tabel berikut ini :

Merk Mobil	1992	1993	1994	1995
1. Honda	35	45	80	120
2. Daihatsu	100	110	70	101
3. Suzuki	10	15	20	17

Gambar 5. 1 Array 2 Demensi

Array tidak terbatas pada satu dimensi. Beberapa masalah membutuhkan penggunaan array dua dimensi, yang mengatur data menjadi baris dan kolom yang serupa ke tabel atau kisi. Elemen individu diakses dengan menentukan dua indeks, satu untuk baris dan satu untuk kolom, $[i, j]$. Gambar 2.10 menunjukkan tampilan abstrak dari larik satu dan dua dimensi. Sedangkan arsitektur komputer menyediakan mekanisme di tingkat perangkat keras membuat dan menggunakan array satu dimensi, mereka biasanya tidak mendukung array dimensi yang lebih tinggi. Sebaliknya, bahasa pemrograman biasanya menyediakan bahasa mereka sendiri mekanisme untuk membuat dan mengelola array yang terdiri dari beberapa dimensi.

Gambar



Gambar 3.3 Contoh Array: (kiri Array 1-D dilihat sebagai list berurutan dan (kanan), Array 2-D dilihat sebagai tabel atau kisi persegi panjang.

Seperti yang kita lihat sebelumnya, Python tidak secara langsung mendukung array built-in dari dimensi apa pun. namun, di bagian sebelumnya, kami dapat menggunakan modul ctypes untuk membuat array yang didukung perangkat keras satu dimensi yang kami gunakan untuk mengimplementasikan Array ADT. Array dua dimensi juga sangat umum dalam pemrograman komputer, di mana mereka digunakan untuk memecahkan masalah yang membutuhkan data untuk diatur menjadi beberapa baris dan kolom. Karena array 2-D tidak disediakan oleh Python, kami mendefinisikan Array2D tipe data abstrak untuk membuat array 2-D. Ini terdiri dari serangkaian operasi terbatas mirip dengan yang disediakan oleh ADT Array satu dimensi.

6. Linked List

Linked list adalah salah satu struktur data dasar paling dasar di bidang ilmu komputer. Dengan menggunakan linked list, programmer dapat menyimpan data saat dibutuhkan. Linked listmirip dengan larik, kecuali bahwa data yang akan disimpan dalam Linked listdapat dialokasikan secara dinamis selama operasi program (waktu proses). Linked lista dalam struktur data yang berisi objek data yang dihubungkan oleh tautan. Setiap Linked listterdiri dari node yang memiliki bidang data dan referensi ke node berikutnya dalam daftar taut.

Dalam sebuah array, jika programmer ingin menyimpan data, programmer harus terlebih dahulu menentukan ukuran array, biasanya programmer mengalokasikan array yang sangat besar (misalnya, 100). Ini tidak efektif karena tidak terlalu besar dalam penggunaan umum. Dan jika programmer ingin menyimpan lebih dari seratus data, itu tidak mungkin karena sifat statis dari array. Linked listadalah struktur data yang dapat menutupi kelemahan ini. Biasanya, Linked listterdiri dari banyak blok data kecil yang ditautkan (biasanya melalui pointer). Linked list memiliki kemampuan untuk memvisualisasikan contoh yakni kereta api, terdapat bagian kepala linked list adalah mesin kereta, data yang disimpan adalah gerbong, dan pengait antar gerbong adalah pointer.

Mesin Data Data

(kepala) —> Pointer —> Pointer —

Programmer membaca data seperti kondektur yang melakukan pemeriksaan tiket. Pemrogram menelusuri Linked list melalui kepalanya, dan kemudian melanjutkan ke truk (data) berikutnya, dan seterusnya, hingga truk terakhir (biasanya ditandai dengan penunjuk yang menunjukkan alamat nol (NULL)). Pengambilan data diselesaikan satu per satu, sehingga traversal data bekerja efektif pada ON. Dibandingkan dengan array, ini adalah kelemahan terbesar dari linked list. Dalam array, jika programmer ingin mengakses data ke-n (indeks n), maka programmer dapat langsung mengaksesnya. Pada saat yang sama, untuk daftar tertaut, programmer harus terlebih dahulu melintasi n data. Linked list adalah sekelompok elemen dari jenis yang sama, mereka memiliki urutan tertentu, dan setiap elemen terdiri dari dua bagian. Bentuk umum:

infotype --> tipe yang ditentukan, yang menyimpan informasi dari elemen daftar
next -> alamat elemen berikutnya (penerus)

Apabila L ialah list, dan P adalah address, maka alamat elemen pertama list L dapat diacu dengan notasi berikut, yakni untuk pendeklarasian di kalukan lebih awal sebelum penggunaannya. Pada Elemen yang diacu oleh P dapat dikonsultasi informasinya dengan notasi yang di artikan dalam definisi berikut:

- List l adalah list kosong, jika First(L) = Nil
- Elemen terakhir dikenali, dengan salah satu cara adalah karena

Next(Last) = Nil

Nilai adalah pengganti Null, perubahan ini dituliskan dengan #define Nil Null.

Jenis-Jenis Linked List

- Singly linked list
- Double linked list
- Circular Linked List

Ruang yang disediakan untuk menyimpan data di area memori tertentu disebut node. Setiap node memiliki pointer ke node berikutnya, sehingga membentuk thread, sehingga hanya diperlukan satu variabel pointer. String

semacam ini disebut Linked list tunggal (NULL memiliki nilai khusus, yang berarti tidak menunjuk ke mana pun. Biasanya Linked list akhirnya akan menunjuk ke NULL). Ada dua cara untuk membuat Single Linked List: LIFO, yakni LIFO (Last In First Out), aplikasi: Stack dan FIFO(First In First Out) dengan aplikasi: Queue(antrian).

a. LIFO (Last In First Out)

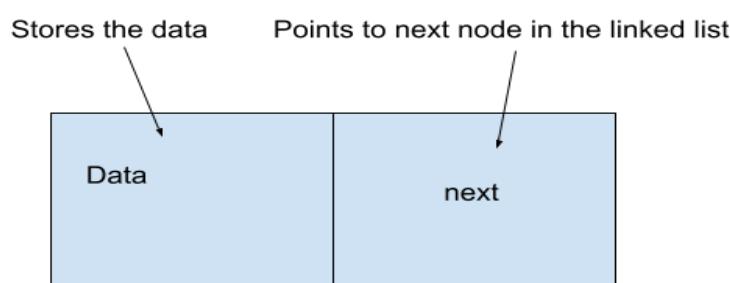
LIFO adalah metode untuk membuat daftar tertaut, di mana data masuk terakhir adalah data keluar paling awal. Ini mirip dengan cara Anda menumpuk item seperti yang dijelaskan di bawah ini. Buatlah node dari linked list sebagai berikut. Jika linked list dibuat metode LIFO, maka akan penambahan pada simpul terakhir, dikenal dengan istilah INSERT.

b. FIFO (First In First Out)

FIFO adalah suatu metode pembuatan Linked List yang data memiliki metode jika ymasuk paling utama merupakan data dimana akan keluar paling utama juga. Hal ini dapat di gambarkan), misalnya saat sekelompok orang yang datang (ENQUEUE) mengantri henddalam sehari-hari pada saat membeli tiket pada antrian loker. Apabila linked list dibuat secara metode FIFO, yang terjadi ialah adanya penambah pada simpul yang ada diposisi depan.

7. Node dalam Linked List

Node adalah objek yang memiliki bidang data dan pointer ke node lain dalam daftar tertaut. Struktur sederhana dari node dapat ditunjukkan pada gambar berikut.

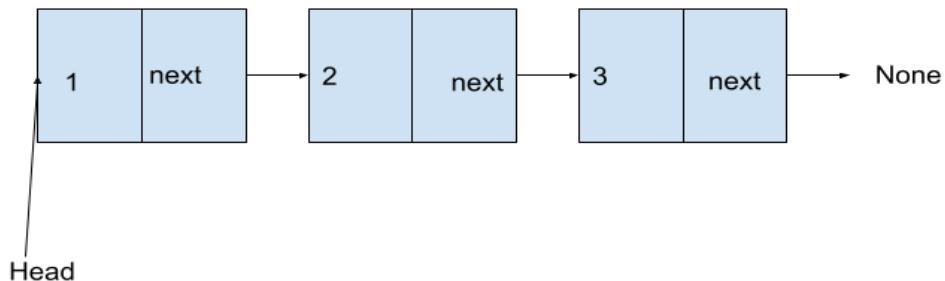


a. Node Linked list dalam python

Mengimplementasikan objek di atas menggunakan Node kelas yang memiliki dua bidang yaitu data dan selanjutnya sebagai berikut

```
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None
```

Linked list terdiri dari banyak node tersebut. Ada penunjuk kepala dalam Linked list yang menunjuk ke node pertama dalam Linked list atau Tidak ada saat Linked list kosong. Angka berikut menggambarkan daftar terkait yang memiliki tiga node.



Linked List dalam python dapat dilihat bahwa bidang berikutnya dari node terakhir menunjuk ke None dan referensi Head menunjuk ke Node pertama. Linked list kosong akan menjadi Linked list yang memiliki penunjuk kepala yang menunjuk ke None. Linked list kosong dapat dibuat dalam python sebagai berikut.

```
class linkedList:
    def __init__(self):
        self.head=None
```

b. Menyisipkan Elemen Linked List

Memasukkan elemen dalam Linked list di posisi pertama, di posisi terakhir di mana saja di antaranya. Untuk menyisipkan elemen dalam Linked

listdi awal, pertama-tama kita akan membuat node dan dengan data yang diberikan dan menetapkan referensi berikutnya ke node pertama yaitu di mana kepala menunjuk. Kemudian kita menunjuk referensi kepala ke node baru. Untuk melakukan operasi ini, kami menerapkan metode insertAtBeginning sebagai berikut.

```
def insertAtBeginning(self,data):  
    temp=Node(data)  
    if self.head==None:  
        self.head=temp  
    else:  
        temp.next=self.head  
        self.head=temp
```

Untuk menyisipkan elemen dalam Linked listdi akhir, kita hanya perlu menemukan node di mana elemen berikutnya mengacu pada None yaitu node terakhir. Kemudian kami membuat node baru dengan data yang diberikan dan mengarahkan elemen berikutnya dari node terakhir ke node yang baru dibuat. Untuk melakukan operasi ini, kami menerapkan metode insertAtEnd sebagai berikut.

```
def insertAtEnd(self,data):  
    temp=Node(data)  
    if self.head==None:  
        self.head=temp  
    else:  
        curr=self.head  
        while curr.next!=None:  
            curr=curr.next  
        curr.next=temp
```

Untuk memasukkan elemen pada posisi lain yang diberikan, kita dapat menghitung node sampai kita mencapai posisi itu dan kemudian kita mengarahkan elemen berikutnya dari node baru ke node berikutnya dari

node saat ini dan referensi berikutnya dari node saat ini ke node baru. Hal ini dapat diimplementasikan dengan menggunakan metode insertAtGivenPosition sebagai berikut.

```
def insertAtGivenPosition(self,data,position):  
    count=1  
    curr=self.head  
    while count<position-1 and curr!=None:  
        curr=curr.next  
        count+=1  
    temp=Node(data)  
    temp.next=curr.next  
    curr.next=temp
```

c. Melintasi Linked list

Untuk melintasi Linked list dalam python, kita akan mulai dari kepala, mencetak data dan pindah ke node berikutnya sampai kita mencapai None yaitu akhir dari daftar yang ditautkan. Metode traverse() berikut mengimplementasikan program untuk melintasi Linked list dalam python.

```
def traverse(self):  
    curr=self.head  
    while curr!=None:  
        print(curr.data)  
        curr=curr.next
```

d. Menghapus Node

Menghapus node baik dari awal atau akhir atau di antara dua node dari daftar tertaut. Untuk menghapus node pertama dari daftar tertaut, pertama-tama akan memeriksa apakah kepala Linked list menunjuk ke None, jika ya maka akan mengajukan pengecualian menggunakan python try

kecuali dengan pesan bahwa Linked list kosong. Jika tidak, akan dihapus node saat ini yang dirujuk dengan kepala dan memindahkan penunjuk kepala ke node berikutnya. Hal ini dapat dilaksanakan sebagai berikut.

```
def delFromBeginning(self):
    try:
        if self.head==None:
            raise Exception("Empty Linked List")
        else:
            temp=self.head
            self.head=self.head.next
            del temp
    except Exception as e:
        print(str(e))
```

Untuk menghapus node terakhir dari daftar tertaut, kita akan melintasi setiap node dalam Linked list dan memeriksa apakah pointer berikutnya dari node berikutnya dari node saat ini menunjuk ke None, jika ya maka node berikutnya dari node saat ini adalah node terakhir dan itu akan dihapus. Hal ini dapat dilaksanakan sebagai berikut.

```
def delFromEnd(self):
    try:
        if self.head==None:
            raise Exception("Empty Linked List")
        else:
            curr=self.head
            prev=None
            while curr.next!=None:
                prev=curr
                curr=curr.next
```

```
prev.next=curr.next  
del curr  
except Exception as e:  
    print(str(e))
```

Untuk menghapus node di antara daftar tertaut, di setiap node akan memeriksa apakah posisi node berikutnya adalah node yang akan dihapus, jika ya, kita akan menghapus node berikutnya dan menetapkan referensi berikutnya ke node berikutnya dari node yang akan dihapus. Hal ini dapat dilakukan sebagai berikut.

```
def delAtPos(self,position):  
    try:  
        if self.head==None:  
            raise Exception("Empty Linked List")  
        else:  
            curr=self.head  
            prev=None  
            count=1  
            while curr!=None and count<position:  
                prev=curr  
                curr=curr.next  
                count+=1  
            prev.next=curr.next  
            del curr  
        except Exception as e:  
            print(str(e))
```

8. Implementasi Linked List di Python

Berikut ini adalah kode python yang berfungsi untuk implementasi Linked list dalam python.

```
1 class Node:
2     def __init__(self,data):
3         self.data=data
4         self.next=None
5 class linkedList:
6     def __init__(self):
7         self.head=None
8     def insertAtBeginning(self,data):
9         temp=Node(data)
10        if self.head==None:
11            self.head=temp
12        else:
13            temp.next=self.head
14            self.head=temp
15     def insertAtEnd(self,data):
16        temp=Node(data)
17        if self.head==None:
18            self.head=temp
19        else:
20            curr=self.head
21            while curr.next!=None:
22                curr=curr.next
23            curr.next=temp
24     def insertAtGivenPosition(self,data,position):
25        count=1
26        curr=self.head
27        while count<position-1 and curr!=None:
28            curr=curr.next
29            count+=1
30        temp=Node(data)
31        temp.next=curr.next
32        curr.next=temp
33     def traverse(self):
34         curr=self.head
```

```

45         del temp
46     except Exception as e:
47         print(str(e))
48 def delFromEnd(self):
49     try:
50         if self.head==None:
51             raise Exception("Empty Linked List")
52         else:
53             curr=self.head
54             prev=None
55             while curr.next!=None:
56                 prev=curr
57                 curr=curr.next
58                 prev.next=curr.next
59                 del curr
60             except Exception as e:
61                 print(str(e))
62 def delAtPos(self,position):
63     try:
64         if self.head==None:
65             raise Exception("Empty Linked List")
66         else:
67             curr=self.head
68             prev=None
69             count=1
70             while curr!=None and count<position:
71                 prev=curr
72                 curr=curr.next
73                 count+=1
74             prev.next=curr.next
75             del curr
76         except Exception as e:
77             print(str(e))
78

```

C. LATIHAN SOAL

1. Jelaskan Menurut Anda Perbedaan Array Dan linked list!
2. Jelaskan 3 Kelebihan Dan Kekurangan Suatu Program Array!
3. Jelaskan bagaimana dan operasi-operasi pada liked List!
4. Buatlah contoh array dalam pemrograman Python!
5. Buatlah linked list dengan pemrograman python!

D. REFERENSI

- Amanulhaq, A. A. (2021). Implementasi Algoritma Image Hashing Dan Hamming Distance Untuk Deteksi Kemiripan Gambar. *Repository Its.*
- Basant Agarwal, B. B. (2018). Hand-On Data Structures And Algorithms With Python. London: Packt Publishing.
- Emi Sita Eriana, A. Z. (2021). Praktikum Algoritma Dan Pemrograman. Tangerang Selatan: Unpam Press.

- Jodi, U. R. (2020). Algoritma Dan Struktur Data.
- 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.
- 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). Pendekripsi Kantuk Secara Real Time Menggunakan Pustaka Opencv Dan Dlib Python. *Sainstech : Jurnal Penelitian Dan Pengkajian Sains Dan Teknologi*.

PERTEMUAN 6

FUNGSI REKURSIF

A. TUJUAN PEMBELAJARAN

Setelah mempelajari Materi ini diharapkan mahasiswa mengerti mengenai fungsi, dan mampu menggunakan dan membuat program dengan fungsi rekursif dengan Python.

B. URAIAN MATERI

1. Definisi Fungsi

Melakukan suatu proses yang memiliki kaitan adalah merupakan tujuan fungsi dimana adalah suatu blok kode yang terstruktur dan dapat memanggil kembali apabila diperlukan. Dalam Python terdapat fungsi bawaan seperti print() dan banyak lagi lainnya, akan tetapi bisa juga dengan fungsi yang dibuat sendiri oleh programmer

Python Functions suatu blok pernyataan terkait yang dirancang untuk melakukan tugas komputasi, logis, atau evaluatif. Idenya adalah untuk menempatkan beberapa tugas yang umum atau berulang kali dilakukan bersama-sama dan membuat fungsi sehingga alih-alih menulis kode yang sama lagi dan lagi untuk input yang berbeda, dimana dapat melakukan panggilan fungsi untuk menggunakan kembali kode yang terkandung di dalamnya berulang kali. Fungsi dapat built-in atau user-defined. Ini membantu program untuk menjadi ringkas, tidak berulang, dan terorganisir. Memanggil Fungsi dimana setelah membuat fungsi kita dapat menyebutnya dengan menggunakan nama fungsi diikuti oleh kurung yang mengandung parameter dari fungsi tertentu.

a. Argumen dari Sebuah Fungsi

Argumen adalah nilai-nilai yang dilewatkan di dalam kurung fungsi. Sebuah fungsi dapat memiliki sejumlah argumen yang dipisahkan oleh koma.

b. Jenis Argumen

Python mendukung berbagai jenis argumen yang dapat diteruskan pada saat panggilan fungsi. Mari kita bahas setiap jenis secara detail.

c. Argumen default

Argumen default ialah parameter dimana mengasumsikan nilai default apabila nilai tidak disediakan dalam pemanggilan fungsi untuk argumen tersebut. Contoh berikut ini menggambarkan argumen default. Seperti argumen default C ++, sejumlah argumen dalam suatu fungsi dapat memiliki nilai default. Tetapi begitu kita memiliki argumen default, semua argumen di sebelah kanannya juga harus memiliki nilai default.

d. Argumen kata kunci

Idenya adalah untuk memungkinkan penelepon untuk menentukan nama argumen dengan nilai-nilai sehingga penelepon tidak perlu mengingat urutan parameter.

e. Argumen panjang variabel

Dalam Python, kita dapat meneruskan sejumlah argumen variabel ke fungsi menggunakan simbol khusus. Ada dua simbol khusus:

args (Argumen Non-Kata Kunci)

**kwargs (Argumen Kata Kunci)

f. Docstring

String pertama setelah fungsi disebut string Dokumen atau Docstring secara singkat. Ini digunakan untuk menggambarkan fungsi fungsi. Penggunaan docstring dalam fungsi adalah opsional tetapi dianggap sebagai praktik yang baik. Sintaks di bawah ini dapat digunakan untuk mencetak docstring fungsi:

Syntax: `print(function_name.__doc__)`

g. Pernyataan kembali

Pernyataan pengembalian fungsi digunakan untuk keluar dari fungsi dan kembali ke fungsi mengembalikan nilai atau item data yang ditentukan ke penelepon.

h. Syntax: return [expression_list]

Pernyataan pengembalian dapat terdiri dari variabel, ekspresi, atau konstanta yang dikembalikan ke akhir eksekusi fungsi. Jika tidak ada yang di atas hadir dengan pernyataan kembali, objek None dikembalikan.

Apakah Fungsi Python Melewati Referensi atau melewati nilai? Di dalam Python setiap nama variabel adalah referensi. Ketika kita meneruskan variabel ke fungsi, referensi baru ke objek dibuat. Parameter yang lewat di Python sama dengan referensi yang lewat di Java. Dalam Python, fungsi anonim berarti bahwa fungsi tanpa nama. Seperti yang sudah kita ketahui, kata kunci def digunakan untuk menentukan fungsi normal dan kata kunci lambda digunakan untuk membuat fungsi anonim. Silakan lihat ini untuk detailnya.

2. Memanggil Function/Fungsi

Mendefinisikan fungsi hanya memberinya nama, penentuan parameter yang harus dimasukkan dalam fungsi dan struktur blok kode. Setelah struktur dasar fungsi diselesaikan, dapat menjalankannya dengan menyebutnya dari fungsi lain atau langsung dari prompt Python. Berikut ini adalah contoh untuk memanggil fungsi printme()

```
1 def printme( str ):
2     "This prints a passed string into this function"
3     print str
4     return;
5
6 # sekarang memanggil fungsi printme
7 printme("Ini panggilan pertama untuk fungsi user defined !")
8 printme("mamanggil kembali untuk fungsi yang sama")
9
```

Ketika kode di atas dieksekusi, ia menghasilkan hasil berikut :

Ini panggilan pertama untuk fungsi user defined !

mamanggil kembali untuk fungsi yang sama

3. Melewati Referensi Vs Nilai

Seluruh parameter (argumen) Python dilewatkkan dengan referensi. Ini berarti apabila mengubah apa parameter mengacu dalam fungsi, perubahan juga mencerminkan kembali dalam fungsi panggilan. Misalnya sebagai berikut.

```

1 def changeme( listku ):
2     "Mengubah list yang terdapat dalam fungsi ini"
3     listku.append([5,6,7,8]);
4     print "Nilai di dalam fungsi: ", listku
5     return
6
7 # sekarang memanggil fungsi changeme
8 listku = [50,60,70];
9 changeme( listku );
10 print "Nilai di luar fungsi: ", listku
11

```

Menjelaskan bahwa mempertahankan referensi objek yang dilewati dan menambahkan nilai dalam objek yang sama. Jadi hasilnya berikut :

Nilai di dalam fungsi: [50, 60, 70, [5, 6, 7, 8]]

Nilai di luar fungsi: [50, 60, 70, [5, 6, 7, 8]]

Ada satu contoh di mana argumen sedang dilewatkkan oleh referensi dan referensi sedang ditimpa di dalam fungsi yang disebut.

```

1 def changeme( listku ):
2     "This changes a passed list into this function"
3     listku = [5,6,7,8]; # This would assig new reference in myList
4     print "Nilai di dalam fungsi: ", listku
5     return
6
7 # sekarang dapat memanggil fungsi changeme
8 listku = [50,60,70];
9 changeme( listku );
10 print "Nilai diluar fungsi: ", listku
11

```

Parameter *listku* bersifat lokal untuk fungsi *changeme*. Mengubah *listku* dalam fungsi tidak mempengaruhi *listku*. Fungsi ini tidak menyelesaikan apa-apa dan akhirnya ini akan menghasilkan hasil berikut :

Nilai di dalam fungsi: [5, 6, 7, 8]

Nilai diluar fungsi: [50, 60, 70]

a. Argumen Fungsi

Programmer dapat memanggil fungsi dengan menggunakan jenis argumen formal berikut.

- 1) Argumen yang diperlukan
- 2) Argumen kata kunci
- 3) Argumen default
- 4) Argumen panjang variabel

b. Argumen yang diperlukan

Argumen yang diperlukan adalah argumen yang diteruskan ke fungsi dalam urutan posisi yang benar. Di sini, jumlah argumen dalam panggilan fungsi harus sesuai persis dengan definisi fungsi. Untuk memanggil fungsi `printme()`, Anda pasti perlu melewati satu argumen, jika tidak, itu memberikan kesalahan sintaks sebagai berikut

```
1 def printme( str ):
2     "This prints a passed string into this function"
3     print str
4     return;
5
6 # Now you can call printme function
7 printme()
8
```

c. Argumen kata kunci

Argumen kata kunci terkait dengan panggilan fungsi. Ketika Anda menggunakan argumen kata kunci dalam panggilan fungsi, penelepon mengidentifikasi argumen dengan nama parameter. Hal ini memungkinkan untuk melewatkannya atau menempatkan mereka keluar dari urutan karena interpreter Python dapat menggunakan kata kunci yang disediakan untuk mencocokkan nilai dengan parameter. Programmer juga dapat membuat panggilan kata kunci ke fungsi `printme()` dengan cara berikut :

```

1 def printme( str ):
2     "Mencetak sebuah posisi string ke dalam fungsi"
3     print str
4     return;
5
6 # Sekarang dapat memanggil fungsi printme
7 printme( str = " String Ku")
8
9
10

```

Ketika kode di atas dieksekusi, ia menghasilkan hasil berikut :

String Ku

Contoh berikut memberikan gambaran yang lebih jelas. Perhatikan bahwa urutan parameter tidak penting.

```

1 def printinfo( nama, umur ):
2     "Mencetak sebuah posisi string ke dalam fungsi"
3     print "Nama: ", nama
4     print "Umur: ", umur
5     return;
6
7 # Sekarang dapat memanggil fungsi printinfo
8 printinfo( umur=20, nama="Emi" )
9
10

```

Ketika kode di atas dieksekusi, ia menghasilkan hasil berikut :

Nama: Emi

Umur: 20

d. Argumen default

Argumen default adalah argumen yang mengasumsikan nilai default jika nilai tidak disediakan dalam fungsi panggilan untuk argumen tersebut. Contoh berikut memberikan ide tentang argumen default, mencetak usia default jika tidak lulus

```

1 def printinfo( nama, umur = 25 ):
2     "Mencetak sebuah posisi string ke dalam fungsi"
3     print "Nama: ", nama
4     print "umur ", umur
5     return;
6
7 # sekarang dapat memanggil fungsi printinfo
8 printinfo( umur=20, nama="Emi" )
9 printinfo( nama="Emi" )
10
11

```

Ketika kode di atas dieksekusi, ia menghasilkan hasil berikut :

Nama: Emi

umur 20

Nama: Emi

umur 25

e. Argumen Panjang Variabel

Kemungkinan memproses fungsi untuk lebih banyak argumen daripada yang ditentukan saat mendefinisikan fungsi. Argumen ini disebut argumen *variabel-panjang* dan tidak disebutkan dalam definisi fungsi, tidak seperti argumen yang diperlukan dan default. Sintaksis untuk fungsi dengan argumen variabel non-kata kunci adalah sebagai berikut.

```

def functionname([formal_args,] *var_args_tuple ):

    "function_docstring"

    function_suite

    return [expression]

```

Tanda bintang (*) ditempatkan di depan nama variabel yang memegang nilai semua argumen variabel nonkeyword. Tuple ini tetap kosong jika tidak ada argumen tambahan yang ditentukan selama panggilan fungsi. Berikut ini adalah contoh sederhana

```

1 def printinfo( arg1, *vartuple ):
2     " mencetak argumen melalui variabel"
3     print "Hasilnya: "
4     print arg1
5     for var in vartuple:
6         print var
7     return;
8
9 # sekarang dapat memanggil fungsi printinfo
10 printinfo( 20 )
11 printinfo( 90, 80, 70 )
12
13

```

Ketika kode di atas dieksekusi, ia menghasilkan hasil berikut :

Hasilnya:

20

Hasilnya:

90

80

70

f. Fungsi Anonim

Fungsi-fungsi ini disebut anonim karena mereka tidak dinyatakan dengan cara standar dengan menggunakan kata kunci *def*. Anda dapat menggunakan kata kunci *lambda* untuk membuat fungsi anonim kecil.

- 1) Bentuk Lambda dapat mengambil sejumlah argumen tetapi mengembalikan hanya satu nilai dalam bentuk ekspresi. Mereka tidak dapat berisi perintah atau beberapa ekspresi.
- 2) Fungsi anonim tidak bisa menjadi panggilan langsung untuk dicetak karena lambda membutuhkan ekspresi.
- 3) Fungsi Lambda memiliki ruang nama lokal mereka sendiri dan tidak dapat mengakses variabel selain yang ada dalam daftar parameter mereka dan yang ada di ruang nama global.
- 4) Meskipun tampaknya lambda adalah versi satu baris dari suatu fungsi, mereka tidak setara dengan pernyataan inline di C atau C ++, yang

tujuannya adalah dengan melewati alokasi tumpukan fungsi selama pemanggilan karena alasan kinerja.

g. Sintaksis

Sintaks fungsi *lambda* hanya berisi satu pernyataan, yaitu sebagai berikut :

```
lambda [arg1 [,arg2,.....argn]]:expression
```

Berikut ini adalah contoh untuk menunjukkan bagaimana bentuk fungsi *lambda* bekerja

```
1 jumlah = lambda arg1, arg2: arg1 + arg2;
2
3 # sekarang dapat memanggil sebuah fungsi sum as
4 print "Total nilai: ", jumlah( 5, 10 )
5 print "Total nilai: ", jumlah( 15, 22 )
6
```

Ketika kode di atas dieksekusi, ia menghasilkan hasil berikut :

Total nilai: 15

Total nilai: 37

h. Pernyataan kembali

Pernyataan kembali [ekspresi] keluar dari fungsi, opsional melewati kembali ekspresi ke penelepon. Pernyataan kembali tanpa argumen sama dengan mengembalikan tidak ada. Semua contoh di atas tidak mengembalikan nilai apa pun. Anda dapat mengembalikan nilai dari fungsi sebagai berikut :

```
1 def jumlah( arg1, arg2 ):
2     # Menambah kedua parameter dan mengembalikan.
3     total = arg1 + arg2
4     print "di dalam function : ", total
5     return total;
6
7 # Now you can call sum function
8 total = jumlah( 7, 21 );
9 print "di luar function : ", total
10
```

Ketika kode di atas dieksekusi, ia menghasilkan hasil berikut :

di dalam function : 28

di luar function : 28

i. Lingkup Variabel

Semua variabel dalam suatu program mungkin tidak dapat diakses di semua lokasi dalam program itu. Ini tergantung di mana Anda telah menyatakan variabel. Ruang lingkup variabel menentukan bagian dari program di mana Anda dapat mengakses pengenal tertentu. Ada dua lingkup dasar variabel dalam Python yaitu variabel global dan lokal

j. Variabel global versi Lokal

Variabel diartikan dalam badan fungsi mempunyai ruang lingkup lokal, dan yang didefinisi di luar yang mempunyai ruang lingkup global. Artinya dalam variabel lokal hanya mampu diakses di dalam fungsi dimana dideklarasikan, sedangkan variabel global mampu diakses pada semua badan program oleh semua fungsi. Ketika memanggil fungsi, variabel yang dinyatakan di dalamnya dibawa ke ruang lingkup. Berikut ini adalah contoh sederhana dibawah ini.

```

1 total = 0; # ini variabel global
2 # disini mendefinisikan fungsi
3 def jumlah( arg1, arg2 ):
4     # Add both the parameters and return them.
5     total = arg1 + arg2; # Here total is local variable.
6     print "Di dalam total fungsi lokal : ", total
7     return total;
8
9 # Now you can call sum function
10 jumlah( 7, 12 );
11 print "Di luar total fungsi lokal : ", total
12
13

```

Ketika kode di atas dieksekusi, ia menghasilkan hasil berikut :

Di dalam total fungsi lokal : 19

Di luar total fungsi lokal : 0

4. Rekursif

Python menerima rekursi fungsi, yang berarti fungsi dapat memanggil dirinya sendiri. Rekursi merupakan dasar ilmu matematika dan pemrograman umum. Berarti sebuah fungsi memanggil dirinya sendiri. dan punya manfaat

dapat mengulang/looping data untuk mencapai hasil. Pengembang harus sangat berhati-hati dengan rekursi karena bisa sangat mudah untuk menyelinap ke dalam menulis fungsi yang tidak pernah berakhir, atau salah satu yang menggunakan kelebihan jumlah memori atau kekuatan prosesor. Namun, ketika ditulis rekursi benar dapat memberikan pendekatan yang sangat efisien dan matematis-elegan untuk pemrograman.

Pada contohnya, tri_recursion () merupakan fungsi dimana telah definisikan untuk memanggil dirinya sendiri ("recurse"). Penggunaan variabel k sebagai data, yang decrements(-1) setiap kali berulang. Rekursi berakhir ketika kondisinya tidak lebih besar dari 0 (yaitu ketika 0). Untuk pengembangan baru, perlu beberapa waktu untuk mencari tahu cara kerjanya, cara terbaik untuk mengetahuinya ialah mengujinya dan memodifikasi.

Penjelasan tentang sesuatu yang mengacu pada dirinya sendiri disebut definisi rekursif. Di formulasi terakhir kami, algoritma pencarian biner menggunakan deskripsinya sendiri. Sebuah "panggilan" untuk pencarian biner "berulang" di dalam definisi karenanya, label "Definisi rekursif." Sekilas, Anda mungkin berpikir definisi rekursif hanya omong kosong. Pasti Anda pernah memiliki seorang guru yang bersikeras bahwa Anda tidak dapat menggunakan satu kata pun di dalamnya definisi? Itu disebut definisi melingkar dan biasanya tidak terlalu berarti kredit pada ujian.

Dalam matematika, bagaimanapun, definisi rekursif tertentu digunakan sepanjang waktu. Selama kita berhati-hati dalam perumusan dan penggunaan definisi rekursif, mereka bisa sangat berguna dan sangat kuat. Rekursif klasik Contoh dalam matematika adalah definisi faktorial. Didefinisikan faktorial dari nilai seperti ini:

$$n! = n(n - 1)(n - 2) \dots (1)$$

Misalnya, menghitung 5!

$$5! = 5(4)(3)(2)(1)$$

Harus dingat bahwa mengimplementasikan program untuk menghitung faktorial menggunakan loop sederhana yang mengakumulasi produk faktorial.

Melihat perhitungan 5! hubungan ini memberi cara lain untuk mengungkapkan apa adanya yang dimaksud dengan faktorial secara umum.

Rekursi sebagai contohnya, saat dua cermin berada paralel antara satu dengan yang lain, gambar yang tertangkap adalah suatu bentuk rekursi tak-terbatas. Secara spesifik hal ini mendefinisikan suatu instansi tidak terbatas dengan menggunakan ekspresi terbatas beberapa instansi bisa merujuk ke instansi lainnya, tetapi dengan suatu cara sehingga tidak ada perulangan atau keterkaitan yang tidak terbatas dapat terjadi. Berikut ini contoh sederhana fungsi rekursif.

Python:

```
def fungsiRekursif():
    print("Hello, ini fungsi rekursif")
    fungsiRekursif()
```

Javascript:

```
function fungsiRekursif(){
    console.log("Hello, ini fungsi rekursif");
    fungsiRekursif();
}
```

Contoh fungsi di atas menampilkan teks "Hello, ini fungsi rekursif" terus menerus, disebabkan pemanggilan diri sendiri tanpa berhenti. Fungsi rekursif mampu menyelesaika berbagai persoalan sebagai contoh dalam perhitungan bilangan fibbonaci dan faktorial. Rekursif dapat menggunakan untuk menghitung deret fibonacci. Deret fibonacci merupakan deret angka sederhana yang susunan angka penjumlahan dari dua angka sebelumnya (0,1,1,2,3,5,8,13,21,...dst).

Deret Fibonacci didefinisikan sebagai:

$$F_n = F_{n-1} + F_{n-2}$$

dimana $F_0 = 0$

and $F_1 = 1$

Membuat fungsi fibonacci pada python cukup mudah. berikut syntaxnya.

```

def fib(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fib(n-1) + fib(n-2)

fib(7)

```

Output yang dihasilkan adalah 13.untuk fungsi fibonacci diatas.

Dapat dilihat bahwa definisi rekursif tidak melingkar karena setiap aplikasi menyebabkan kita meminta faktorial dari bilangan yang lebih kecil. Akhirnya kita dapatkan down to, yang tidak memerlukan aplikasi definisi lain. Ini adalah disebut kasus dasar untuk rekursi. Ketika rekursi berhenti, kita mendapatkan sebuah ekspresi tertutup yang bisa langsung dihitung. Semua definisi rekursif yang baik memiliki karakteristik utama berikut:

- Ada satu atau lebih kasus dasar yang tidak memerlukan rekursi.
- Semua rantai rekursi akhirnya berakhir di salah satu kasus dasar.

Cara termudah untuk menjamin bahwa kedua syarat ini terpenuhi adalah dengan membuatnya pastikan bahwa setiap rekursi selalu terjadi pada versi yang lebih kecil dari masalah aslinya. Versi sangat kecil dari masalah yang dapat diselesaikan tanpa rekursi kemudian menjadi kasus dasar. Persis seperti inilah definisi faktorial bekerja.

Program rekursif harus dinyatakan dalam prosedur atau fungsi, karena hanya prosedur dan fungsi yang dapat dipanggil dalam sebuah program. Fungsi atau Prosedur Rekursif disusun oleh dua komponen :

- Basis (komponen basis) fungsi : Menghentikan rekursif dan memberi nilai yang terdefinisi
- Aturan rekursif (komponen induksi) fungsi : mendefinisikan dengan dirinya sendiri

Aturan rekursif dapat memungkinkan komputasi tak berhingga bila komputasi tidak mencapai komponen basis. Pemrogram harus berhati-hati dalam membuat definisi rekursif. Definisi harus menunjukkan pergerakan yang semakin menuju ke objek yang lebih sederhana sehingga memungkinkan

definisi tersebut berhenti. Studi Kasus Rekursif pada contoh Program menghitung factorial :

Permasalahan : $5! = 5 \times 4 \times 3 \times 2 \times 1$

$$= 120$$

```

2
3 a = raw_input("masukkan nilai faktorial yang ingin dicari =")
4 c = int(a)
5 for i in range(c):
6     if i==0:
7
8         c=c*1
9     else :
10        c=c*i
11    print (c)
```

```

In [4]: %run "D:\kuliah\STRUKTUR-DATA\Python\faktorial.py"
masukkan nilai faktorial yang ingin dicari =5
5
10
30
120
In [5]:
```

Hal perlu diperhatikan saat melakukan proses rekursif yakni dengan memastikan batas atau kondisi berhenti berproses pada rekursif. Pada fungsi fact di atas kondisi berhenti jika $n=0$. Sebelum menerapkan proses rekursif harus yakin bahwa syarat berhenti dapat dicapai, jangan tidak henti. Untuk itu, berbagai kondisi yang dapat menyebabkan rekursif tak berujung harus dapat ditangani (contoh apabila masukkan $n<0$). Proses rekursif tak berujung akan menyebabkan memori habis, dan akan mengalami gangguan.

5. Anonymous/Lambda Function

Anonymous Function atau Lambda Function adalah fungsi yang didefinisikan tanpa nama. fungsi biasa diawali dengan def, akan tetapi lambda function diawali dengan lambda. pada dasarnya fungsi lambda memiliki perilaku yang sama dengan fungsi biasa, kita bisa memasukkan berapapun parameter akan tetapi hanya dapat memiliki satu ekspresi atau perintah dan

mengembalikan nilainya. bahkan fungsi ini ditulis dalam satu baris kode saja. berikut adalah syntax lambda function.

Diawali dengan keyword lambda lalu parameter, titik dua, dan perintah. Kita dapat memasukkan fungsi ini kedalam variabel agar dapat digunakan. berikut contoh fungsi lambda.

```
double = lambda x: x * 2
x = lambda a, b : a * b
print(double(5))
print(x(5, 6))
```

Output yang dihasilkan 10 30 untuk fungsi lambda diatas. biasanya lambda function digunakan pada fungsi yang parameternya adalah fungsi seperti map(), dan filter()

6. Fungsi dalam Python

Fungsi yang didefinisikan dalam fungsi lain dikenal sebagai fungsi dalam atau fungsi bersarang. Fungsi bersarang dapat mengakses variabel dari lingkup melampirkan. Fungsi bersarang digunakan sehingga dapat dilindungi dari segala sesuatu yang terjadi di luar fungsi, yang berguna dalam penentuan fungsi dimana sebagai penyedia yang dierlukan programmer. Berikut adalah aturan sederhana untuk menentukan fungsi dalam Python.

- a. Blok fungsi diawali dengan key word def selanjutnya diikui dengan nama fungsi dan tanda kurung ()).
- b. Masing-masing parameter input atau argumen penempatannya dalam kurung. Programmer bisa membuat parameter di tanda kurung
- c. Pernyataan kesatu dari fungsi berupa pernyataan opsional - string dokumentasi fungsi atau *docstring*.
- d. Blok kode dalam setiap fungsi diawali dengan dengan tanda (:) serta diindentasinya.
- e. Pernyataan kembali [ekspresi] keluar dari fungsi, opsional melewati kembali ekspresi celled. Pernyataan kembali tanpa argumen sama dengan mengembalikan tidak ada.

Sintaksis

```
def namafungsi( parameter ):  
    "function_docstring"  
    function_suite  
    return [ekspresi]
```

Secara default, parameter memiliki perilaku posisional dan Anda perlu memberi tahu mereka dalam urutan yang sama dengan yang didefinisikan.

Contoh

Fungsi berikut mengambil string sebagai parameter input dan mencetaknya pada layar standar.

```
def printme( str ):  
    "This prints a passed string into this function"  
    print str  
    return
```

C. LATIHAN SOAL

1. Jelaskan menurut Anda mengenai fungsi!
2. Jelaskan perbedaan program dengan rekursif dan interatif!
3. Jelaskan kekurangan dan kelebihan apabila menggunakan rekursif dan interatif!
4. Buatlah program pemangkatan dengan menggunakan rekursif!
5. Tuliskanlah sebuah fungsi yang menjumlahkan dua buah integer, dan menghasilkan sebuah integer, dengan membuat definisi rekursif dari penjumlahan!

D. REFERENSI

Amanulhaq, A. A. (2021). Implementasi Algoritma Image Hashing Dan Hamming Distance Untuk Deteksi Kemiripan Gambar. *repository ITS*.

Basant Agarwal, B. B. (2018). Hand-On Data Structures and Algorithms With Python. London: Packt Publishing.

- Emi Sita Eriana, A. Z. (2021). Praktikum Algoritma dan Pemrograman. Tangerang Selatan: Unpam Press.
- Jodi, u. R. (2020). Algoritma Dan Struktur Data.
- 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.
- 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). Pendekripsi Kantuk Secara Real Time Menggunakan Pustaka OPENCV dan DLIB PYTHON. Sainstech : Jurnal Penelitian dan Pengkajian Sains dan Teknologi.

PERTEMUAN 7

TEKNIK SEARCH

A. TUJUAN PEMBELAJARAN

Setelah mempelajari Materi ini diharapkan mahasiswa mengerti dan paham teknik search, teknik binary, pencarian binary dan mampu membuat pencarian binary dengan pemrograman python.

B. URAIAN MATERI

1. Konsep Dasar Search

Pencarian atau search, yang telah dibahas di seluruh teks, adalah pilihan yang sangat umum. operasi dan telah dipelajari secara ekstensif. Pencarian linier dari sebuah array atau Python list sangat lambat, tetapi itu dapat ditingkatkan dengan pencarian biner. Bahkan dengan waktu pencarian yang lebih baik, array dan list Python memiliki kelemahan untuk penyisipan dan penghapusan tombol pencarian. Ingat, pencarian biner hanya bisa dilakukan pada urutan yang diurutkan. Saat kunci ditambahkan atau dihapus dari file array atau list Python, urutannya harus dipertahankan. Ini bisa memakan waktu karena kunci harus digeser untuk memberi ruang saat menambahkan kunci baru atau untuk menutup celah saat menghapus kunci yang ada. Penggunaan list tertaut memberikan penyisipan yang lebih cepat dan penghapusan tanpa harus menggeser tombol yang ada. Sayangnya, satu-satunya jenis pencarian yang dapat dilakukan pada list tertaut adalah pencarian linier, meskipun file list diurutkan. Dalam bab ini, kami mengeksplorasi beberapa dari banyak cara dalam struktur pohon dapat digunakan dalam melakukan pencarian yang efisien.

Struktur pohon dapat digunakan mengatur data dinamis secara hierarkis. Pohon memiliki berbagai bentuk dan ukuran tergantung pada aplikasinya dan hubungan antar node. Saat digunakan untuk mencari, setiap node berisi kunci pencarian sebagai bagian dari datanya entri (terkadang disebut payload) dan node diatur berdasarkan hubungan antar kunci. Ada banyak jenis pohon pencarian, beberapa di antaranya hanya variasi yang lain, dan beberapa di

antaranya dapat digunakan untuk mencari data disimpan secara eksternal. Tetapi tujuan utama dari semua pohon pencarian adalah untuk menyediakan efisiensi operasi pencarian untuk dengan cepat menemukan item spesifik yang terdapat di pohon.

Pohon pencarian dapat digunakan untuk mengimplementasikan berbagai jenis wadah, beberapa di antaranya mungkin hanya perlu menyimpan kunci pencarian dalam setiap node pohon. Lebih umum, bagaimanapun, aplikasi mengasosiasikan data atau muatan dengan masing-masing cari dan gunakan struktur dengan cara yang sama seperti ADT peta bekas. Peta ADT pada saat itu kami menerapkannya itu menggunakan struktur list. Latihan dalam beberapa bab menawarkan kesempatan untuk menyediakan implementasi baru menggunakan berbagai struktur data.

Konsep teknik pencarian dikembangkan karena keterbatasan yang diberlakukan oleh teknologi pencarian kata kunci Boolean klasik saat menangani kumpulan teks digital yang besar dan tidak terstruktur. Pencarian kata kunci sering kali memberikan hasil yang menyertakan banyak item yang tidak relevan (positif palsu) atau yang mengeluarkan terlalu banyak item relevan (negatif palsu) karena efek sinonim dan polisemi. Sinonimi berarti bahwa satu dari dua atau lebih kata dalam bahasa yang sama memiliki arti yang sama, dan polisemi berarti bahwa banyak kata individual memiliki lebih dari satu arti. Polisemi merupakan kendala utama bagi semua sistem komputer yang berusaha menangani bahasa manusia. Dalam bahasa Inggris, istilah yang paling sering digunakan memiliki beberapa arti yang sama. Misalnya, kata api dapat berarti: aktivitas pembakaran; untuk menghentikan pekerjaan; untuk meluncurkan, atau untuk menggairahkan (seperti dalam api). Untuk 200 istilah paling polysemous dalam bahasa Inggris, kata kerja tipikal memiliki lebih dari dua belas arti atau pengertian umum. Kata benda khas dari himpunan ini memiliki lebih dari delapan pengertian umum. Untuk 2000 istilah paling polysemous dalam bahasa Inggris, kata kerja tipikal memiliki lebih dari delapan pengertian umum dan kata benda tipikal memiliki lebih dari lima.

Selain masalah polisema dan sinonim, penelusuran kata kunci dapat mengecualikan kata yang salah eja secara tidak sengaja serta variasi pada batang (atau akar) kata (misalnya, serang vs. mencolok). Pencarian kata kunci juga rentan terhadap kesalahan yang disebabkan oleh proses pemindaian

pengenalan karakter optik (OCR), yang dapat menyebabkan kesalahan acak ke dalam teks dokumen (sering disebut sebagai teks berisik) selama proses pemindaian.

Pencarian konsep dapat mengatasi tantangan ini dengan menggunakan disambiguasi arti kata (WSD), dan teknik lainnya, untuk membantunya mendapatkan arti sebenarnya dari kata-kata tersebut, dan konsep dasarnya, bukan hanya dengan mencocokkan string karakter seperti teknologi pencarian kata kunci .

2. Teknik Pencarian

Proses pencarian suatu data tertentu pada sekumpulan data bertipe sama. Dalam proses pemrograman searching/pencarian biasanya digunakan untuk pemrosesan dalam update maupun hapus data sebelum melakukan pemrosesan pencarian data. Kedua yatu cara menyediakan data dalam kumpulan data, apabila data berada maka proses penyisipan tidak diperkenankan. Jika data tidak ada maka proses penyisipan dilakukan untuk tujuan menghindari duplikat data. Tehnik Pencarian Tunggal meliput tehnik Sequential Search / Linier Search dan tehnik Binary Search

a. Pencarian Sekuensial

Teknik dengan menngunakan algoritma searching yang sederhana dimana pencarian secara runtun dengan melakukan perbandingan antar elemen satu sama secara urut diawali element pertama sampai ditemukannya data yang diinginkan. Pencarian adalah kebutuhan yang sangat mendasar ketika Anda menyimpan data dalam struktur data yang berbeda. Pendekatan yang paling sederhana adalah untuk pergi di setiap elemen dalam struktur data dan mencocokkannya dengan nilai yang Anda cari. Ini dikenal sebagai pencarian linear. Ini tidak efisien dan jarang digunakan, tetapi membuat program untuk itu memberikan gambaran tentang bagaimana kita dapat menerapkan beberapa algoritma pencarian canggih.

b. Pencarian Linear

Dalam jenis pencarian ini, pencarian berurutan dilakukan di semua item satu per satu. Setiap item diperiksa dan jika kecocokan ditemukan maka item

tertentu dikembalikan, jika tidak, pencarian berlanjut sampai akhir struktur data. Contoh pencarian linier dalam python seperti dibawah ini.

```

1 def linear_search(values, search_for):
2     search_at = 0
3     search_res = False
4 # Match the value with each data element
5     while search_at < len(values) and search_res is False:
6         if values[search_at] == search_for:
7             search_res = True
8         else:
9             search_at = search_at + 1
10    return search_res
11 l = [64, 34, 25, 12, 22, 11, 90]
12 print(linear_search(l, 12))
13 print(linear_search(l, 91))
14

```

Hasil

Ketika kode di atas dieksekusi, ia menghasilkan hasil berikut :

True

False

c. Pencarian Interpolasi

Algoritma pencarian ini bekerja pada posisi menyelis nilai yang diperlukan. Agar algoritma ini berfungsi dengan baik, pengumpulan data harus dalam bentuk yang diurutkan dan didistribusikan secara merata. Awalnya, posisi probe adalah posisi item paling tengah dari koleksi. Jika kecocokan terjadi, maka indeks item dikembalikan. Jika item tengah lebih besar dari item, maka posisi probe kembali dihitung dalam sub-array di sebelah kanan item tengah. Jika tidak, item dicari di subarray di sebelah kiri item tengah. Proses ini berlanjut pada sub-array juga sampai ukuran subarray berkurang menjadi nol. Pencarian Biner menggunakan Rekursi Menerapkan algoritma pencarian biner menggunakan python seperti yang ditunjukkan di bawah ini. Kami menggunakan daftar item yang dipesan dan merancang fungsi rekursif untuk mengambil dalam daftar bersama dengan indeks awal dan akhir sebagai input. Kemudian, fungsi pencarian biner memanggil dirinya sendiri sampai

menemukan item yang dicari atau menyimpulkan tentang ketidakdirinya dalam daftar. Contoh sebagai berikut.

```

1 def bsearch(list, idx0, idxn, val):
2     if (idxn < idx0):
3         return None
4     else:
5         midval = idx0 + ((idxn - idx0) // 2)
6     # Compare the search item with middle most value
7     if list[midval] > val:
8         return bsearch(list, idx0, midval-1, val)
9     elif list[midval] < val:
10        return bsearch(list, midval+1, idxn, val)
11    else:
12        return midval
13 list = [8,11,24,56,88,131]
14 print(bsearch(list, 0, 5, 24))
15 print(bsearch(list, 0, 5, 51))
16

```

Hasil

Ketika kode di atas dieksekusi, ia menghasilkan hasil berikut :

2

None

d. Pencarian biner

Pencarian Biner adalah algoritma pencarian yang digunakan untuk mencari elemen dari array yang diurutkan. Ini tidak dapat digunakan untuk mencari dari array yang tidakurut. Pencarian biner adalah algoritma yang efisien dan lebih baik daripada pencarian linier dalam hal kompleksitas waktu.

Kompleksitas waktu pencarian linier adalah $O(n)$. Sedangkan kompleksitas waktu pencarian biner adalah $O(\log n)$. Oleh karena itu, pencarian biner adalah algoritma yang efisien dan lebih cepat mencari tetapi hanya dapat digunakan untuk mencari dari array yang diurutkan.

3. Cara Kerja Binary Search

Ide dasar di balik pencarian biner adalah bahwa alih-alih membandingkan elemen yang diperlukan dengan semua elemen array, kita akan membandingkan elemen yang diperlukan dengan elemen tengah array. Jika ini

ternyata menjadi elemen yang kita cari, kita selesai dengan pencarian dengan sukses. Lain, jika elemen yang kita cari kurang dari elemen tengah, dapat dipastikan bahwa elemen tersebut terletak pada bagian pertama atau kiri dari array, karena array diurutkan. Demikian pula, jika elemen yang kita cari lebih besar dari elemen tengah, dapat dipastikan bahwa elemen tersebut terletak di paruh kedua array.

Dengan demikian, pencarian biner terus mengurangi array menjadi dua. Proses di atas diterapkan secara rekursif pada bagian yang dipilih dari array sampai kita menemukan elemen yang kita cari. Akan mulai mencari dengan indeks kiri 0 dan indeks kanan sama dengan indeks terakhir dari array. Indeks elemen tengah (tengah) dihitung yang merupakan jumlah indeks kiri dan kanan dibagi dengan 2. Jika elemen yang diperlukan kurang dari elemen tengah, maka indeks yang tepat diubah menjadi pertengahan 1, yang berarti kita sekarang akan melihat paruh pertama array saja. Demikian juga, jika elemen yang diperlukan lebih besar dari elemen tengah, maka indeks kiri diubah menjadi pertengahan + 1, yang berarti kita sekarang akan melihat paruh kedua array saja. Kami akan mengulangi proses di atas untuk setengah array yang dipilih.

Bagaimana tahu jika elemen tidak ada dalam array. Kita perlu memiliki beberapa kondisi untuk berhenti mencari lebih lanjut yang akan menunjukkan bahwa elemen tidak ada dalam array. Kami akan secara berulang mencari elemen dalam array selama indeks kiri kurang dari atau sama dengan indeks yang tepat. Setelah kondisi ini berubah menjadi salah dan kami belum menemukan elemennya, ini berarti bahwa elemen tersebut tidak ada dalam array. Contoh nya dibawah ini mari ambil array yang diurutkan berikut dan kita perlu mencari elemen 6.

2		5	6	8	10	11	13	15	16
---	--	---	---	---	----	----	----	----	----

L=0 H=8 Mid=4

2	5	6	8	10	11	13	15	16
---	---	---	---	----	----	----	----	----

$6 < 10$, oleh karena itu mengambil babak pertama.

$H = \text{Pertengahan}-1$

$L=0 H=3 \text{ Mid}=1$

2	5	6	8	10	11	13	15	16
---	---	---	---	----	----	----	----	----

$6 > 5$, jadi pilih babak kedua.

$L=\text{Tengah}+1$

$L=2 H=3 \text{ Mid}=2$

2	5	6	8	10	11	13	15	16
---	---	---	---	----	----	----	----	----

$6 == 6$, elemen yang ditemukan

Oleh karena itu elemen 6 ditemukan pada indeks 2.

Pelaksanaan

Dari array yang diurutkan tertentu, cari elemen yang diperlukan dan cetak indeksnya jika elemen hadir dalam array. Jika elemen tidak ada, cetak -1.

Kode untuk implementasi pencarian biner diberikan di bawah ini.

Contoh

```

1 def binary_search(arr,x):
2     l=0
3     r=len(arr)-1
4     while(l<=r):
5         mid=(l+r)//2
6         if(arr[mid]==x):
7             return mid
8         elif(x<arr[mid]):
9             r=mid-1
10        elif(x>arr[mid]):
11            l=mid+1
12    return -1
13 array=[1,2,3,4,5,6,7,8,9,10]
14 a=7
15 print(binary_search(array,a))
16 b=15
17 print(binary_search(array,b))
18

```

Hasilnya jika di run.

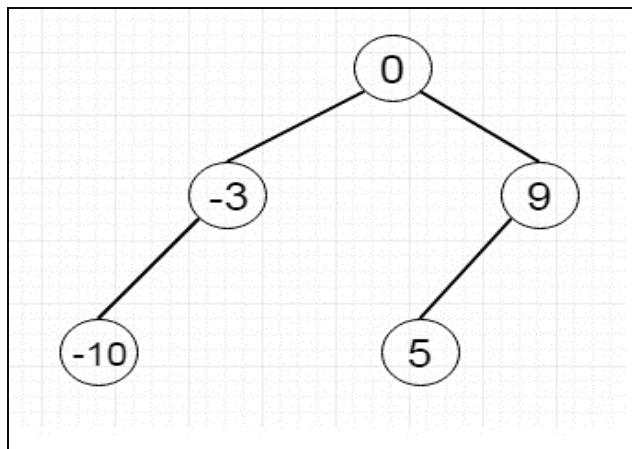
6

-1

Elemen 7 hadir di indeks 6.

Elemen 15 tidak ada dalam array, maka -1 dicetak.

Misalkan kita memiliki satu array A yang diurutkan. Kita harus menghasilkan satu pencarian biner yang seimbang tinggi. Dalam masalah ini, pohon biner yang seimbang tinggi sebenarnya adalah pohon biner di mana kedalaman dua subtrees dari setiap node tidak pernah berbeda lebih dari 1. Misalkan array seperti [-10, -3, 0, 5, 9]. Jadi satu output yang mungkin akan seperti: [0, -3, 9, -10, null, 5]



Untuk mengatasi hal ini, kami akan mengikuti langkah-langkah ini.

Jika A kosong, maka kembalikan Null

menemukan elemen tengah, dan menjadikannya root

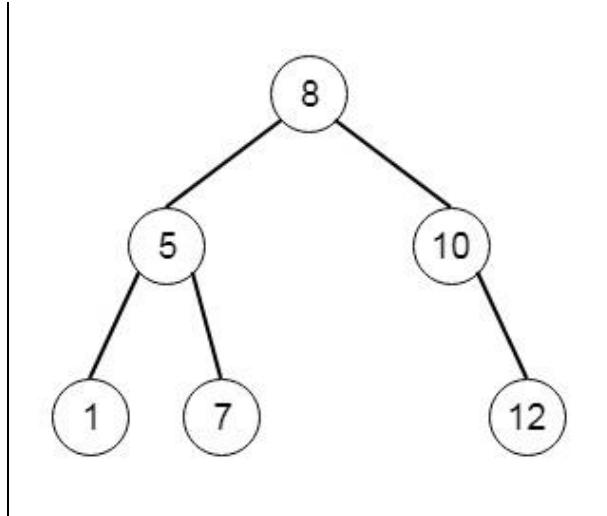
Bagi array menjadi dua sub-array, bagian kiri elemen tengah, dan bagian kanan elemen tengah

rekursif melakukan tugas yang sama untuk subarray kiri dan subarray kanan.

Mari kita lihat implementasi berikut untuk mendapatkan pemahaman yang lebih baik.

Misalkan kita harus membuat pohon pencarian biner yang cocok dengan traversal preorder yang diberikan. Jadi jika traversal pre-order seperti

[8,5,1,7,10,12], maka outputnya akan [8,5,10,1,7,7,12], jadi pohon akan menjadi



Untuk mengatasi hal ini, ikuti langkah-langkah ini :

akar := 0th node dari daftar traversal preorder

tumpukan := tumpukan, dan dorong akar ke dalam tumpukan

untuk setiap elemen i dari elemen kedua dari daftar preorder

i := node dengan nilai i

jika nilai i < bagian atas elemen atas tumpukan, maka

kiri dari node atas tumpukan := i

masukkan i ke dalam tumpukan

sebaliknya

sementara tumpukan tidak kosong, dan menumpuk nilai elemen atas < nilai i

terakhir := bagian atas tumpukan

elemen pop dari stack

kanan node terakhir := i

masukkan i ke dalam tumpukan

akar pengembalian

4. Pencarian Binery di Python

Penerapan algoritma pencarian binery menggunakan Python untuk menemukan posisi indeks elemen dalam daftar yang diberikan. Pencarian biner untuk menemukan elemen tertentu dalam daftar. Misalkan kita memiliki daftar seribu elemen, dan kita perlu mendapatkan posisi indeks dari elemen tertentu. Kita dapat menemukan posisi indeks elemen dengan sangat cepat menggunakan algoritma pencarian biner. Ada banyak algoritma pencarian tetapi pencarian biner paling populer di antara mereka. Elemen dalam daftar harus diurutkan untuk menerapkan algoritma pencarian biner. Jika elemen tidak diurutkan maka urutkan terlebih dahulu.

Konsep Pencarian Biner Dalam algoritma pencarian biner, kita dapat menemukan posisi elemen menggunakan metode berikut.

- a. Metode Rekursif
- b. Metode Iteratif

Teknik pendekatan divide and conquer diikuti dengan metode rekursif. Dalam metode ini, fungsi disebut sendiri lagi dan lagi sampai menemukan elemen dalam daftar. Satu set pernyataan diulang beberapa kali untuk menemukan posisi indeks elemen dalam metode berulang. Sementara loop digunakan untuk menyelesaikan tugas ini. Pencarian biner lebih efektif daripada pencarian linier karena kita tidak perlu mencari setiap indeks daftar. Daftar harus diurutkan untuk mencapai algoritma pencarian biner. Langkah demi langkah implementasi pencarian biner, memiliki daftar elemen yang diurutkan, dan kami mencari posisi indeks 45.

[12, 24, 32, 39, 45, 50, 54]

Jadi penetapan dua petunjuk dalam daftar kami. Satu pointer digunakan untuk menunjukkan nilai yang lebih kecil yang disebut rendah dan pointer kedua digunakan untuk menunjukkan nilai tertinggi yang disebut tinggi. Selanjutnya, menghitung nilai elemen tengah dalam array.

$$\text{pertengahan} = (\text{rendah}+\text{tinggi})/2$$

Di sini, rendah adalah 0 dan tinggi adalah 7.

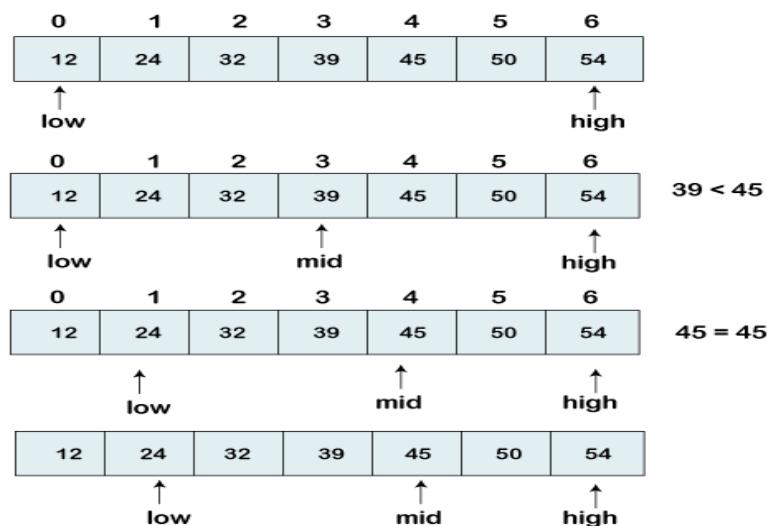
$$\text{pertengahan} = (0+7)/2$$

$$\text{pertengahan} = 3 \text{ (Integer)}$$

Sekarang akan membandingkan elemen yang dicari dengan nilai indeks menengah. Dalam hal ini, 32 tidak sama dengan 45. Jadi perlu melakukan perbandingan lebih lanjut untuk menemukan elemen. Jika angka yang dicari sama dengan pertengahan. Kemudian kembali pertengahan sebaliknya pindah ke perbandingan lebih lanjut. Angka yang akan dicari lebih besar dari angka tengah, membandingkan n dengan elemen tengah elemen di sisi kanan pertengahan dan diatur rendah ke rendah

= pertengahan + 1.

Jika tidak, bandingkan n dengan elemen tengah elemen di sisi kiri tengah dan atur tinggi ke tinggi = pertengahan - 1.



Ulangi sampai nomor yang dicari ditemukan.

Penerapan pencarian biner dengan metode berulang kemudian mengulangi serangkaian pernyataan dan mengulangi setiap item dari daftar, menemukan nilai tengah sampai pencarian selesai. Menciptakan fungsi yang disebut fungsi `binary_search()` yang membutuhkan dua argumen - daftar untuk diurutkan dan nomor yang akan dicari. Dua variabel untuk menyimpan nilai terendah dan tertinggi dalam daftar. Yang rendah diberi nilai awal ke 0, tinggi ke `len(list1) - 1` dan pertengahan sebagai 0.

Selanjutnya, pernyatakan loop sementara dengan kondisi bahwa yang terendah sama dan lebih kecil dari yang tertinggi. Sementara loop akan berulang jika jumlahnya belum ditemukan. Sementara loop, menemukan nilai tengah dan membandingkan nilai indeks dengan jumlah yang cari. Jika nilai

indeks menengah lebih kecil dari n,kita meningkatkan nilai tengah sebesar 1 dan menetapkannya ke Pencarian bergerak ke sisi kiri. Jika tidak, kurangi nilai tengah dan tetapkan ke yang tinggi. Pencarian bergerak ke sisi kanan. Jika n sama dengan nilai tengah maka kembali pertengahan. Ini akan terjadi sampai rendah sama dan lebih kecil dari tinggi.

Jika mencapai di akhir fungsi, maka elemen tidak hadir dalam daftar. Kembali 1 ke fungsi panggilan.

Kompleksitas algoritma pencarian biner adalah $O(1)$ untuk kasus terbaik. Ini terjadi jika elemen yang dicari ditemukan dalam perbandingan pertama. $O(\log n)$ adalah yang terburuk dan kompleksitas kasus rata-rata pencarian biner. Hal ini tergantung pada jumlah pencarian yang dilakukan untuk menemukan elemen yang cari. Algoritma pencarian biner adalah cara yang paling efisien dan cepat untuk mencari elemen dalam daftar. Ini melewati perbandingan yang tidak perlu. Seperti namanya, pencarian dibagi menjadi dua bagian. Ini berfokus pada sisi daftar, yang dekat dengan jumlah yang kami cari.

C. LATIHAN SOAL

1. Jelaskan yang anda ketahui teknik search
2. sebutkan 2 dan jelaskan metode dalam teknik search!
3. Jelaskan kelebihan dan kekurangan linier search!
4. Jelaskan bagaimana cara kerja dalam pencarian binary!
5. Buatlah contoh implementasi teknik search dengan bahasa python!

D. REFERENSI

Basant Agarwal, B. B. (2018). *Hand-On Data Structures And Algorithms With Python*. London: Packt Publishing.

Jodi, U. R. (2020). Algoritma Dan Struktur Data.

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*
- 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.

PERTEMUAN 8

BUBBLE SORT

A. TUJUAN PEMBELAJARAN

Setelah mempelajari ini diharapkan mahasiswa mengerti dan paham teknik sorting dan bubble sort, cara kerja dan membuat program bubble sort dengan Python.

B. URAIAN MATERI

1. Sorting

Sorting adalah proses pengurutan data yang sebelumnya disusun secara acak sehingga menjadi tersusun secara teratur menurut suatu aturan tertentu.

Ada dua jenis pengurutan :

- a. Ascending (naik)
- b. Descending (turun)

Masalah pengurutan dan menjelajahi pengurutan dasar algoritma, sebagian besar algoritme pengurutan dapat dibagi menjadi dua kategori: urutan perbandingan dan urutan distribusi. Sebagai perbandingan sort, item data dapat diatur dalam bentuk menaik (dari terkecil hingga terbesar) atau urutan turun (dari terbesar ke terkecil) dengan melakukan kompromi logika berpasangan “parisons” di antara tombol sortir. Perbandingan berpasangan biasanya didasarkan pada baik urutan numerik saat bekerja dengan bilangan bulat dan real atau leksikografis atau order saat bekerja dengan string dan urutan. Semacam distribusi, di sisi lain tangan, mendistribusikan atau membagi kunci pengurutan menjadi kelompok atau koleksi perantara berdasarkan nilai kunci individu. Misalnya, perhatikan masalah pengurutan sebuah list nilai numerik berdasarkan nilai huruf yang setara, bukan yang sebenarnya nilai numerik. Nilai dapat dibagi menjadi beberapa kelompok berdasarkan korespondensi nilai huruf tanpa harus membuat perbandingan antara nilai numerik.

Algoritme pengurutan menggunakan loop berulang bersarang untuk mengurutkan urutan nilai. Dalam bab ini, kami menjelajahi dua jenis

perbandingan tambahan algoritma, keduanya menggunakan rekursi dan menerapkan strategi divide and conquer untuk mengurutkan urutan. Banyak jenis perbandingan juga dapat diterapkan ke list tertaut, yang kami jelajahi bersama dengan salah satu jenis distribusi yang lebih umum.

2. Bubble Sort

Bubble Sort adalah algoritma pengurutan dimana untuk mengurutkan item daftar dalam urutan naik dengan membandingkan dua nilai yang berdekatan. Jika nilai pertama lebih tinggi dari nilai kedua, nilai pertama mengambil posisi nilai kedua, sedangkan nilai kedua mengambil posisi nilai pertama. Jika nilai pertama lebih rendah dari nilai kedua, maka tidak ada swapping yang dilakukan. Proses ini diulang sampai semua nilai dalam daftar telah dibandingkan dan ditukar jika perlu. Setiap iterasi biasanya disebut pass. Jumlah pass dalam jenis gelembung sama dengan jumlah elemen dalam daftar minus satu.

Implementasi menjadi tiga (3) langkah, yaitu masalah, solusi, dan algoritma yang dapat kami gunakan untuk menulis kode untuk bahasa apa pun. Masalahnya daftar item diberikan secara acak, dan ingin mengatur item dengan tertib.

Pertimbangkan daftar berikut:

[21,6,9,33,3]

Solusinya

Iterasi melalui daftar dimana dilakukan pembandingan dua elemen yang berdekatan dan menukar apabila nilai pertama lebih tinggi dari nilai kedua. Hasilnya harus sebagai berikut:

[3,6,9,21,33]

a. Cara Kerja Algoritma Bubblesort

Algoritma Bubblesort bekerja sebagai berikut

Langkah 1) Dapatkan jumlah total elemen. Dapatkan jumlah total item dalam daftar yang diberikan

Langkah 2) Tentukan jumlah outer pass ($n - 1$) yang harus dilakukan. Panjangnya adalah daftar minus satu

Langkah 3) Lakukan umpan dalam ($n - 1$) kali untuk outer pass 1. Dapatkan nilai elemen pertama dan bandingkan dengan nilai kedua. Jika nilai kedua kurang dari nilai pertama, maka swap posisi

Langkah 4) Ulangi langkah 3 melewati sampai Anda mencapai outer pass ($n - 1$). Dapatkan elemen berikutnya dalam daftar kemudian ulangi proses yang dilakukan pada langkah 3 sampai semua nilai telah ditempatkan dalam urutan naik yang benar.

Langkah 5) Kembalikan hasilnya ketika semua umpan telah dilakukan. Mengembalikan hasil daftar yang diurutkan

Langkah 6) Optimalkan Algoritma

Hindari umpan dalam yang tidak perlu jika daftar atau nilai yang berdekatan sudah diurutkan. Misalnya, jika daftar yang disediakan sudah berisi elemen yang telah diurutkan dalam urutan naik, maka kita dapat memecah loop lebih awal.

b. Mengoptimalkan Algoritma Bubble Sort

Secara default, algoritma untuk jenis gelembung di Python membandingkan semua item dalam daftar terlepas dari apakah daftar sudah diurutkan atau tidak. Jika daftar yang diberikan sudah diurutkan, membandingkan semua nilai adalah buang-buang waktu dan sumber daya. Mengoptimalkan jenis gelembung membantu kita menghindari iterasi yang tidak perlu dan menghemat waktu dan sumber daya. Misalnya, jika item pertama dan kedua sudah diurutkan, maka tidak perlu iterasi melalui sisa nilai. Iterasi dihentikan, dan yang berikutnya dimulai sampai proses selesai seperti yang ditunjukkan pada contoh Bubble Sort di bawah ini. Optimasi dilakukan dengan menggunakan langkah-langkah berikut

Langkah 1) Membuat variabel bendera yang memantau jika ada swapping yang terjadi di loop dalam

Langkah 2) Jika nilai telah bertukar posisi, lanjutkan ke iterasi berikutnya

Langkah 3) Jika manfaatnya belum bertukar posisi, hentikan loop dalam, dan lanjutkan dengan loop luar.

Jenis gelembung yang dioptimalkan lebih efisien karena hanya menjalankan langkah-langkah yang diperlukan dan melewatkkan langkah-langkah yang tidak diperlukan.

3. Representasi Visual

Mengingat daftar lima elemen, gambar berikut menggambarkan bagaimana bubble sort iterates melalui nilai saat menyortirnya

Gambar berikut menunjukkan daftar yang tidak disorting



Iterasi Pertama

Langkah 1)



Nilai 21 dan 6 dibandingkan dengan memeriksa mana yang lebih besar dari yang lain.



21 lebih besar dari 6, jadi 21 mengambil posisi yang ditempati oleh 6 sementara 6 mengambil posisi yang ditempati oleh 21



Daftar modifikasi kami sekarang terlihat seperti yang di atas.

Langkah 2)



Nilai 21 dan 9 dibandingkan.



21 lebih besar dari 9, jadi kita menukar posisi 21 dan 9



Daftar baru sekarang seperti di atas

Langkah 3)



Nilai 21 dan 33 dibandingkan untuk menemukan yang lebih besar.



Nilai 33 lebih besar dari 21, jadi tidak ada swapping yang terjadi.

Langkah 4)



Nilai 33 dan 3 dibandingkan untuk menemukan yang lebih besar.



Nilai 33 lebih besar dari 3, jadi kami menukar posisi mereka.



Daftar yang diurutkan di akhir iterasi pertama adalah seperti yang di atas.

Iterasi Kedua

Daftar baru setelah iterasi kedua adalah sebagai berikut:



Iterasi Ketiga

Daftar baru setelah iterasi ketiga adalah sebagai berikut:



Iterasi Keempat

Daftar baru setelah iterasi keempat adalah sebagai berikut:



4. Keuntungan dan Kekurangan bubble sort

Berikut ini adalah beberapa keuntungan dari algoritma jenis gelembung.

- a. Sangat mudah untuk memahami
- b. Ini berkinerja sangat baik ketika daftar sudah atau hampir diurutkan.
- c. Tidak memerlukan memori yang luas.
- d. Sangat mudah untuk menulis kode untuk algoritma.
- e. Persyaratan ruang minimal dibandingkan dengan algoritma penyortiran lainnya.

Berikut ini adalah beberapa kelemahan dari algoritma jenis gelembung.

- a. Itu tidak berkinerja baik saat menyortir daftar besar. Dibutuhkan terlalu banyak waktu dan sumber daya.
- b. Ini sebagian besar digunakan untuk tujuan akademik dan bukan aplikasi dunia nyata.
- c. Jumlah langkah yang diperlukan untuk mengurutkan daftar adalah urutan n^2

5. Analisis Kompleksitas Bubble Sort

Ada tiga jenis kompleksitas adalah:

- a. Urutkan kompleksitas

Kompleksitas jenis digunakan untuk mengekspresikan jumlah waktu eksekusi dan ruang yang diperlukan untuk mengurutkan daftar. Jenis gelembung membuat $(n - 1)$ iterasi untuk mengurutkan daftar di mana n adalah jumlah total elemen dalam daftar.

- b. Kompleksitas waktu

Kompleksitas waktu dari jenis gelembung adalah $O(n^2)$

Kompleksitas waktu dapat dikategorikan sebagai:

- 1) Kasus terburuk – disinilah daftar yang disediakan berada dalam urutan menurun. Algoritma ini melakukan jumlah maksimum eksekusi yang dinyatakan sebagai [Big-O] $O(n^2)$
- 2) Kasus terbaik – ini terjadi ketika daftar yang disediakan sudah diurutkan. Algoritma melakukan jumlah minimum eksekusi yang dinyatakan sebagai [Big-Omega] $O(n)$
- 3) Kasus rata-rata – ini terjadi ketika daftar dalam urutan acak. Kompleksitas rata-rata direpresentasikan sebagai [Big-theta] $\Theta(n^2)$

c. Space complexity

Kompleksitas ruang mengukur jumlah ruang ekstra yang diperlukan untuk menyortir daftar. Jenis gelembung hanya membutuhkan satu (1) ruang ekstra untuk variabel temporal yang digunakan untuk bertukar nilai. Oleh karena itu, ia memiliki kompleksitas ruang $O(1)$.

Jadi secara ringkas metode bubble sort ialah:

- a. Algoritma Bubble sort bekerja dengan membandingkan dua nilai yang berdekatan dan menukar jika nilai di sebelah kiri kurang dari nilai di sebelah kanan.
- b. Menerapkan algoritma bubble sort relatif lurus ke depan dengan Python. Yang perlu di gunakan adalah untuk loop dan jika pernyataan.
- c. Masalah yang dipecahkan algoritma jenis gelembung adalah mengambil daftar item acak dan mengubahnya menjadi daftar yang dipesan.
- d. Algoritma jenis gelembung dalam struktur data berkinerja terbaik ketika daftar sudah diurutkan karena melakukan sejumlah minimal iterasi.
- e. Algoritma penggontor gelembung tidak berkinerja baik ketika daftar dalam urutan terbalik.
- f. Jenis bubbler memiliki kompleksitas waktu $O(n^2)$ dan kompleksitas ruang $O(1)$
- g. Algoritma jenis bubbler paling cocok untuk tujuan akademik dan bukan aplikasi dunia nyata.

- h. Jenis gelembung yang dioptimalkan membuat algoritma lebih efisien dengan melewatan iterasi yang tidak perlu saat memeriksa nilai yang telah diurutkan

6. Insertion Sort (Metode Penyisipan)

Dalam kebanyakan kasus, jenis penyisipan adalah yang terbaik dari jenis dasar yang dijelaskan di sini bab. Ini masih dijalankan dalam waktu $O(N^2)$, tetapi sekitar dua kali lebih cepat dari jenis sorting bubble/gelembung dan agak lebih cepat daripada jenis pemilihan dalam situasi normal. Ini juga tidak terlalu rumit, meskipun sedikit lebih terlibat daripada gelembung dan jenis seleksi. Ini sering digunakan sebagai tahap akhir dari jenis yang lebih canggih, seperti quicksort. Sortir Penyisipan pada jenis penyisipan jika kita mulai di tengah proses, saat tim sedang setengah diurutkan.

Insertion Sort merupakan algoritma yang efisien untuk mengurutkan angka yang mempunyai jumlah elemen sedikit. Dimana Input : deretan angka sejumlah n buah dan Output ialah permutasi (pengurutan) sejumlah n angka dari input yang sudah terurut secara ascending maupun descending. Metode penyisipan (Insertion sort) bertujuan untuk menjadikan bagian sisi kiri array terurutkan sampai dengan seluruh array berhasil diurutkan. Metode ini mengurutkan bilangan-bilangan yang telah dibaca; dan berikutnya secara berulang akan menyisipkan bilangan-bilangan dalam array yang belum terbaca ke sisi kiri array yang telah terurut.

Insertion Sort bekerja seperti banyak orang yang sedang mengurutkan kartu di tangan. Dimulai dengan tangan kiri yang kosong dan kartunya tertumpuk di meja. Selanjutnya kita ambil satu persatu kartu di meja dan diletakkan di tangan kiri dengan posisi yang benar (terurut). Untuk menemukan posisi yang benar, maka kita harus membandingkan satu persatu kartu yang ada (di tangan kiri) secara berurutan.

7. Visualisasi Insertion Sort

Visualisasi pada insertsort digambarkan di bawah ini.

3	10	4	6	8	9	7	2	1	5
---	----	---	---	---	---	---	---	---	---

Bagian biru/abu-abu (dua bilangan pertama) sekarang dalam keadaan terurut secara relatif.

3	10	4	6	8	9	7	2	1	5
---	----	---	---	---	---	---	---	---	---

Berikutnya, kita perlu menyisipkan bilangan ketiga (4) ke dalam bagian biru/abu-abu sehingga setelah penyisipan tersebut, bagian biru/abu-abu tetap dalam keadaan terurut secara relatif.

Caranya :

pertama : Ambil bilangan ketiga (4).

3	10		6	8	9	7	2	1	5
4									

Kedua : Geser bilangan kedua (10) shg ada ruang untuk disisipi.

3		10	6	8	9	7	2	1	5
4									

Ketiga : Sisipkan bilangan 4 ke posisi yang tepat

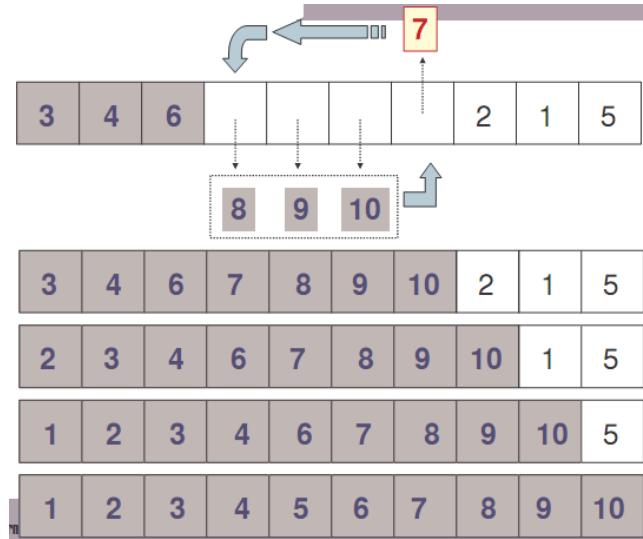
3	4	10	6	8	9	7	2	1	5
3 4 6 10 8 9 7 2 1 5									

Sekarang, tiga bilangan pertama sudah terurut secara relatif dan kita sisipkan bilangan keempat kepada tiga bilangan pertama tsb. Setelah penyisipan, empat bilangan pertama haruslah dalam keadaan terurut secara relatif.

3	4	6	10	8	9	7	2	1	5
3 4 6 10 8 9 7 2 1 5									

Ulangi proses tersebut sampai bilangan terakhir disisipkan

3	4	6	8	10	9	7	2	1	5
3	4	6	8	9	10	7	2	1	5



8. Implementasi Bubble Sort dan Insert Sort di Python

Kode berikut menunjukkan cara menerapkan algoritma Bubble Sort di Python.

```

1 def bubbleSort( theSeq ):
2     n = len( theSeq )
3     for i in range( n - 1 ) :
4         flag = 0
5         for j in range(n - 1) :
6             if theSeq[j] > theSeq[j + 1] :
7                 tmp = theSeq[j]
8                 theSeq[j] = theSeq[j + 1]
9                 theSeq[j + 1] = tmp
10            flag = 1
11            if flag == 0:
12                break
13        return theSeq
14 el = [21,6,9,33,3]
15 result = bubbleSort(el)
16 print (result)
```

Hasilnya [3, 6, 9, 21, 33]

Penjelasan untuk kode program Python Bubble Sort adalah sebagai berikut:

```

1  def bubbleSort( theSeq ): 1
2      n = len( theSeq ) 2
3
4          for i in range( n - 1 ) : 3
5              flag = 0 4
6
7                  for j in range(n - 1) : 5
8
9                      if theSeq[j] > theSeq[j + 1] : 6
10                         tmp = theSeq[j] 7
11                         theSeq[j] = theSeq[j + 1] 8
12                         theSeq[j + 1] = tmp 9
13                         flag = 1 10
14
15                  if flag == 0: 11
16                      break 12
17
18          return theSeq 13
19
20 el = [21,6,9,33,3] 14
21
22 result = bubbleSort(el) 15
23
24 print (result) 16

```

Penjelasannya sebagai berikut.

- Mendefinisikan fungsi bubbleSort yang menerima parameter theSeq. Kode tidak menghasilkan apapun.
- Menperoleh panjang array dan menetapkan nilai ke variabel n. Kode tidak menghasilkan apa-apa
- Memulai loop untuk yang menjalankan algoritma bubble sort ($n - 1$) kali. Ini adalah loop luar.
- Mendefinisikan variabel yang akan digunakan untuk menentukan apakah swap telah terjadi atau tidak. Ini untuk tujuan optimasi
- Mulai loop dalam yang membandingkan semua nilai dalam daftar dari yang pertama hingga yang terakhir.
- Menggunakan pernyataan if untuk memeriksa apakah nilai di sisi kiri lebih besar dari yang ada di sisi kanan langsung.

- g. Memberikan nilai theSeq[j] ke tmp variabel temporal jika kondisi mengevaluasi ke true.
- h. Nilai theSeq[j +1] ditugaskan ke posisi TheSeq[j].
- i. Nilai tmp variabel ditugaskan untuk memposisikan Seq[j +1].
- j. Variabel bendera diberi nilai 1 untuk menunjukkan bahwa swap telah terjadi.
- k. Menggunakan pernyataan jika untuk memeriksa apakah nilai bendera variabel adalah 0.
- l. Jika nilainya adalah 0, maka kita sebut pernyataan istirahat yang melangkah keluar dari lingkaran dalam.
- m. Mengembalikan nilai Seq setelah diurutkan. Kode tersebut menampilkan daftar yang diurutkan.
- n. Mendefinisikan el variabel yang berisi daftar angka acak.
- o. Menetapkan nilai gelembung fungsiSort ke hasil variabel.
- p. Mencetak nilai hasil variabel.

Pada contoh pemrograman Bubble sort pada Python seperti dibawah ini

```

3
4 def bubbleSort(alist):
5     for passnum in range(len(alist)-1,0,-1):
6         for i in range(passnum):
7             if alist[i]>alist[i+1]:
8                 temp = alist[i]
9                 alist[i] = alist[i+1]
10                alist[i+1] = temp
11
12 alist = [54,26,93,17,77,31,44,55,20]
13 bubbleSort(alist)
14 print(alist)

```

Hasil jika dirun sebagai berikut [17, 20, 26, 31, 44, 54, 55, 77, 93]

Pada contoh 2 pemrograman Insertsort pada Python seperti dibawah ini

```

3
4 def insertionSort(alist):
5     for index in range(1,len(alist)):
6
7         currentvalue = alist[index]
8         position = index
9
10        while position>0 and alist[position-1]>currentvalue:
11            alist[position]=alist[position-1]
12            position = position-1
13
14        alist[position]=currentvalue
15
16 alist = [54,26,93,17,77,31,44,55,20]
17 insertionSort(alist)
18 print(alist)

```

Hasil [17, 20, 26, 31, 44, 54, 55, 77, 93]

C. LATIHAN SOAL

1. Jelaskan yang anda ketahui teknik Bubble Sort dan Insertsort!
2. Buatlah analogi/ visual bubble sort dalam 6 angka yang akan dilakukan pengurutan!
3. Jelaskan kelebihan dan kekurangan dengan Bubble sort dan Insertsort!
4. Jelaskan bagaimana mengopsimalkan Bubble Sort!
5. Buatlah contoh implementasi teknik Insertsort dengan bahasa python!

D. REFERENSI

Basant Agarwal, B. B. (2018). *Hand-On Data Structures And Algorithms With Python*. London: Packt Publishing.

Jodi, U. R. (2020). Algoritma Dan Struktur Data.

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*
- 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.

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 mampu 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 $O(n \log n)$. 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 list disalin kembali ke seurutan. 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 $O(n \log n)$ 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 ketengah+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 pembagian 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 $O(n\log n)$.

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 ($\text{start} + \text{end} / 2$), 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

```

1 # Python program for implementation of MergeSort
2 # Merges two subarrays of arr[].
3 # First subarray is arr[1..m]
4 # Second subarray is arr[m+1..r]
5
6 def merge(arr, l, m, r):
7     n1 = m - l + 1
8     n2 = r - m
9     # Create temp arrays
10    L = [0] * (n1)
11    R = [0] * (n2)
12
13    # Copy data to temp arrays L[] and R[]
14    for i in range(0, n1):
15        L[i] = arr[l + i]
16
17    for j in range(0, n2):
18        R[j] = arr[m + 1 + j]
19
20    # Merge the temp arrays back into arr[l..r]
21    i = 0      # Initial index of first subarray
22    j = 0      # Initial index of second subarray
23    k = l      # Initial index of merged subarray
24
25    while i < n1 and j < n2:
26        if L[i] <= R[j]:
27            arr[k] = L[i]
28            i += 1
29        else:
30            arr[k] = R[j]
31            j += 1
32        k += 1
33
34    # Copy the remaining elements of L[], if there
35    # are any
36    while i < n1:
37        arr[k] = L[i]
38        i += 1
39        k += 1
40

```

```
41 # Copy the remaining elements of R[], if there
42 # are any
43 while j < n2:
44     arr[k] = R[j]
45     j += 1
46     k += 1
47
48 # l is for left index and r is right index of the
49 # sub-array of arr to be sorted
50
51 def mergeSort(arr, l, r):
52     if l < r:
53
54         # Same as (l+r)//2, but avoids overflow for
55         # large l and h
56         m = l+(r-l)//2
57
58         # Sort first and second halves
59         mergeSort(arr, l, m)
60         mergeSort(arr, m+1, r)
61         merge(arr, l, m, r)
62
63 # Driver code to test above
64 arr = [121, 10, 18, 51, 6, 73]
65 n = len(arr)
66 print("Given array is")
67 for i in range(n):
68     print("%d" % arr[i]),
69
70 mergeSort(arr, 0, n-1)
71 print("\n\nSorted array is")
72 for i in range(n):
73     print("%d" % arr[i]),
74
```

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.

```
4 def mergeSort(alist):
5     print("Splitting ",alist)
6     if len(alist)>1:
7         mid = len(alist)//2
8         lefthalf = alist[:mid]
9         righthalf = alist[mid:]
10        mergeSort(lefthalf)
11        mergeSort(righthalf)
12
13        i=0
14        j=0
15        k=0
16        while i<len(lefthalf) and j<len(righthalf):
17            if lefthalf[i]<righthalf[j]:
18                alist[k]=lefthalf[i]
19                i=i+1
20            else:
21                alist[k]=righthalf[j]
22                j=j+1
23                k=k+1
24            while i<len(lefthalf):
25                alist[k]=lefthalf[i]
26                i=i+1
27                k=k+1
28            while j<len(righthalf):
29                alist[k]=righthalf[j]
30                j=j+1
31                k=k+1
32    print("Merging ",alist)
33 alist = [54,26,93,17,77,31,44,55,20]
34 mergeSort(alist)
35 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 Databases.
- 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). Pendekripsi Kantuk Secara Real Time Menggunakan Pustaka Opencv Dan Dlib Python. *Sainstech : Jurnal Penelitian Dan Pengkajian Sains Dan Teknologi*.

PERTEMUAN 10

TEKNIK SELECTION SORT

A. TUJUAN PEMBELAJARAN

Setelah mempelajari materi pertemuan ini diharapkan mahasiswa mengerti dan paham teknik selection sort, kompleksitas waktu, kelebihan dan kekurangan selectin sort dan mampu membuat program selection sort

B. URAIAN MATERI

1. Selection Sort

Pemanggilan data dengan suatu metode yang disebut sortir pada list untuk mengurutkan item dalam list dalam urutan menaik. Urutan naik ditentukan oleh kurang dari operator seperti yang didefinisikan pada item. Jadi bagaimana cara kerja algoritma pengurutan. Satu dari algoritma pengurutan awal disebut Pengurutan Seleksi dan berfungsi sebagai permulaan yang baik tempat untuk memahami algoritma pengurutan.

Selection Sort adalah algoritma penyortiran perbandingan yang digunakan untuk mengurutkan daftar item acak dalam urutan naik. Perbandingan tidak membutuhkan banyak ruang ekstra. Ini hanya membutuhkan satu ruang memori tambahan untuk variabel temporal. Ini dikenal sebagai penyortiran di tempat. Jenis seleksi memiliki kompleksitas waktu $O(n^2)$ di mana n adalah jumlah total item dalam daftar. Kompleksitas waktu mengukur jumlah iterasi yang diperlukan untuk mengurutkan daftar. Daftar ini dibagi menjadi dua partisi: Daftar pertama berisi item yang diurutkan, sedangkan daftar kedua berisi item yang tidak disortir.

Secara default, daftar yang diurutkan kosong, dan daftar yang tidak disortir berisi semua elemen. Daftar yang tidak disortir kemudian dipindai untuk nilai minimum, yang kemudian ditempatkan dalam daftar yang diurutkan. Proses ini diulang sampai semua nilai telah dibandingkan dan diurutkan. Bagaimana cara kerja tipe seleksi?

Item pertama dalam partisi yang tidak disorting dibandingkan dengan semua nilai ke sisi kanan untuk memeriksa apakah itu adalah nilai minimum.

Jika bukan nilai minimum, maka posisinya ditukar dengan nilai minimum. Misalnya, jika indeks nilai minimum adalah 3, maka nilai elemen dengan indeks 3 ditempatkan pada indeks 0 sedangkan nilai yang berada di indeks 0 ditempatkan pada indeks 3. Jika elemen pertama dalam partisi yang tidak disortir adalah nilai minimum, maka ia mengembalikan posisinya.

Elemen yang telah ditentukan sebagai nilai minimum kemudian dipindahkan ke partisi di sisi kiri, yang merupakan daftar yang diurutkan. Sisi yang dipartisi sekarang memiliki satu elemen, sedangkan sisi yang tidak dipartisi memiliki $(n - 1)$ elemen di mana n adalah jumlah total elemen dalam daftar. Proses ini diulang berulang-ulang sampai semua item telah dibandingkan dan diurutkan berdasarkan nilai-nilai mereka. Contoh pada selection sort

Daftar elemen yang dalam urutan acak perlu diurutkan dalam urutan naik. Pertimbangkan daftar berikut sebagai contoh.

[21,6,9,33,3]

10 Pertanyaan dan Jawaban Wawancara Perilaku Teratas

Daftar di atas harus diurutkan untuk menghasilkan hasil berikut

[3,6,9,21,33]

Solusi (Algoritma)

Langkah 1) Dapatkan nilai n yang merupakan ukuran total array

Langkah 2) Partisi daftar menjadi bagian yang diurutkan dan tidaksortir. Bagian yang diurutkan awalnya kosong sementara bagian yang tidak disortir berisi seluruh daftar

Langkah 3) Pilih nilai minimum dari bagian yang tidak dipartisi dan letakkan ke bagian yang diurutkan.

Langkah 4) Ulangi proses $(n - 1)$ kali sampai semua elemen dalam daftar telah diurutkan.

Algoritma urutan pemilihan cukup sederhana untuk dijelaskan. Algoritma dimulai dengan mencari nilai terkecil untuk ditempatkan di posisi pertama dalam list. Ini dilakukan dengan melakukan pencarian linier melalui list dan sepanjang jalan mengingat index barang terkecil yang ditemukannya. Algoritma

menggunakan pola tebak dan periksa terlebih dahulu menebak bahwa item terkecil adalah item pertama dalam list dan kemudian memeriksa item berikutnya untuk melihat apakah itu membuat tebakan yang salah. Bagian dari Algoritma ini adalah bagian seleksi. Fungsi pemilihan melakukan pemilihan ini. Fungsi pada Selection Sort sebagai berikut.

```
def select(seq, start):
    minIndex = start

    for j in range(start+1, len(seq)):
        if seq[minIndex] > seq[j]:
            minIndex = j
    return minIndex
```

Argumen start memberi tahu fungsi select tempat mulai mencari yang terkecil barang. Ini mencari dari awal hingga akhir urutan untuk item terkecil. Algoritma pengurutan pilihan bekerja dengan menemukan item terkecil menggunakan pilihan berfungsi dan menempatkan item itu ke posisi pertama dari urutan. Sekarang nilainya masuk posisi pertama harus diletakkan di tempat lain. Itu hanya ditukar dengan lokasi dari nilai yang sedang dipindahkan. Algoritma dilanjutkan dengan mencari nilai terkecil kedua dalam urutan tersebut. Karena nilai terkecil sekarang ada di yang pertama lokasi dalam urutan, algoritma pengurutan seleksi mulai melihat dari yang kedua posisi dalam list untuk nilai terkecil. Ketika nilai terkecil ditemukan (yaitu benar-benar nilai terkecil kedua untuk list) nilai di posisi kedua dan nilai ini ditukar. Kemudian algoritma sortir seleksi mencari item terkecil dimulai dari lokasi ketiga secara berurutan. Pola ini berulang sampai semua item secara berurutan telah diurutkan. Fungsi selSort melakukan penyortiran sebenarnya algoritma.

Selection sort mencari elemen yang tepat untuk diletakkan di posisi yang telah diketahui, dan meletakkannya di posisi tersebut setelah data tersebut ditemukan Selection Sort Membandingkan elemen yang sekarang dengan elemen yang berikutnya sampai dengan elemen yang terakhir. Jika ditemukan elemen lain yang lebih kecil dari elemen sekarang maka dicatat posisinya dan kemudian ditukar. Pengurutan data dalam struktur data sangat penting untuk data yang beripe data numerik ataupun karakter.Pengurutan dapat dilakukan secara ascending (urut naik) dan descending (urut turun) Pengurutan (Sorting)

adalah proses menyusun kembali data yang sebelumnya telah disusun dengan suatu pola tertentu, sehingga tersusun secara teratur menurut aturan tertentu.

Metode ini memiliki konsep memilih data yang maksimum/minimum dari suatu kumpulan data larik L, lalu menempatkan data tersebut ke elemen paling akhir atau paling awal sesuai pengurutan yang diinginkan. Data maksimum/minimum yang diperoleh, diasingkan ke tempat lain, dan tidak diikutsertakan pada proses pencarian data maksimum/minimum berikutnya. Perhatikan ilustrasi berikut : Misalkan ada sekumpulan data acak berjumlah n elemen yang disimpan di dalam larik L, akan diurut menaik, maka langkah-langkah yang harus dilakukan adalah :

- a. Menentukan jumlah iterasi, yaitu pass = $n-2$
- b. Untuk setiap pass ke- $i = 0, 1, 2, \dots, n-2$, lakukan
- c. Cari elemen terbesar (maks) dari elemen ke- i sampai ke- $(n-1)$
- d. Pertukaran maks dengan elemen ke- i
- e. Kurangi n satu ($n = n - 1$)

Rincian tiap-tiap pas adalah sebagai berikut :

- a. pass 0

Cari elemen maksimum di dalam $L[0 \dots (n-1)]$.

Pertukarkan elemen maksimum dengan elemen $L[n-1]$

- b. pass 1

Cari elemen maksimum di dalam $L[0 \dots (n-2)]$

Pertukarkan elemen maksimum dengan elemen $L[n-2]$

- c. pass 2

Cari elemen maksimum di dalam $L[0 \dots (n-3)]$

Pertukarkan elemen maksimum dengan elemen $L[n-3]$

- d. pass 3

Cari elemen maksimum di dalam $L[0 \dots 1]$

Pertukarkan elemen maksimum dengan elemen $L[1]$

Metode ini dapat divisualisasikan algoritma pengurutan seleksi dengan menjalankan animasi pengurutannya. Animasi digambarkan pada Gambar diatas setelah menyortir lebih dari setengah nilai dalam sebuah urutan. Titik hijau mewakili item yang sekarang berada di lokasi yang tepat di urutan. Tinggi titik dari sumbu x (yaitu nilai y) adalah nilainya. Itu sumbu x adalah posisi dalam list. Dalam animasi ini semua nilai antara 17 dan 93 sedang diurutkan ke dalam urutan menaik. Sudut kanan atas mewakili nilai-nilai itu yang belum disortir. Algoritma mulai mencari nilai terkecil berikutnya tepat di sebelah kanan garis diagonal hijau. Ia menemukan nilai minimum (yaitu terdekat ke sumbu x) dengan menelusuri semua titik yang tidak disortir. Setelah menemukan file kecil, titik terpendek, ini menukar dengan titik paling kiri untuk memasukkannya ke dalam posisi yang diurutkan dalam list.

2. Representasi Visual

Dalam algoritma penyortiran pilihan, array diurutkan dengan rekursif menemukan elemen minimum dari bagian yang tidak disortir dan memasukkannya di awal. Dua subarray terbentuk selama pelaksanaan jenis Seleksi pada array tertentu.

- a. Subarray, yang sudah diurutkan
- b. Subarray, yang tidak sortir.

Selama setiap iterasi jenis seleksi, elemen minimum dari subarray yang tidak disortir muncul dan dimasukkan ke dalam subarray yang diurutkan. Mari melihat representasi visual dari algoritma dibawah ini.

**Gambar 10. 1** Representasi Visual Selection Sort

Mengingat daftar lima elemen, gambar berikut menggambarkan bagaimana algoritma sortasi pilihan iterates melalui nilai-nilai saat menyortirnya.

Gambar berikut menunjukkan daftar yang tidak disorting



Langkah 1)



Nilai pertama 21 dibandingkan dengan sisa nilai untuk memeriksa apakah itu adalah nilai minimum.



3 adalah nilai minimum, sehingga posisi 21 dan 3 ditukar. Nilai dengan latar belakang hijau mewakili partisi yang diurutkan dari daftar.

Langkah 2)



Nilai 6 yang merupakan elemen pertama dalam partisi yang tidak disortortasi dibandingkan dengan sisa nilai untuk mengetahui apakah ada nilai yang lebih rendah.



Nilai 6 adalah nilai minimum, sehingga mempertahankan posisinya.

Langkah 3)



Elemen pertama dari daftar yang tidak disortort dengan nilai 9 dibandingkan dengan sisa nilai untuk memeriksa apakah itu adalah nilai minimum.



Nilai 9 adalah nilai minimum, sehingga mempertahankan posisinya dalam partisi yang diurutkan.

Langkah 4)



Nilai 33 dibandingkan dengan nilai lainnya.



Nilai 21 lebih rendah dari 33, sehingga posisi ditukar untuk menghasilkan daftar baru di atas.

Langkah 5)



Kami hanya memiliki satu nilai yang tersisa dalam daftar yang tidak dipartisi. Oleh karena itu, sudah disortir.



Daftar terakhir seperti yang ditunjukkan pada gambar di atas.

Sekarang lihat implementasi algoritma dalam contoh pemrograman dibawah ini.

```

1 A = ['t', 'u', 't', 'o', 'r', 'i', 'a', 'l']
2 for i in range(len(A)):
3     min_= i
4     for j in range(i+1, len(A)):
5         if A[min_] > A[j]:
6             min_ = j
7     #swap
8     A[i], A[min_] = A[min_], A[i]
9 # main
10 for i in range(len(A)):
11     print(A[i])
12
13

```

Silahkan mencoba dari soure code diatas sampai hasilnya seperti dibawah

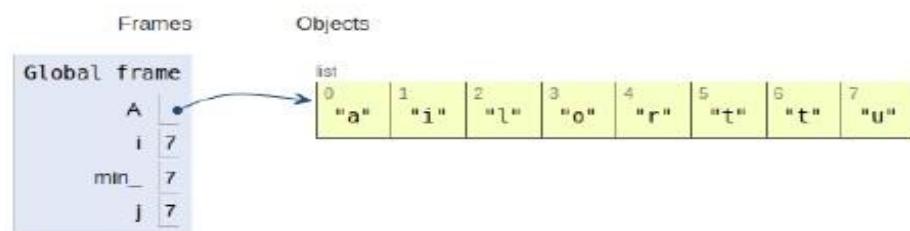
a
i
l
o
r
t
t
u

Output dari algoritma dalam urutan naik. Min_ adalah nilai saat ini yang semakin dibandingkan dengan semua nilai lainnya. Parameter analisis algoritma tercantum di bawah ini

Kompleksitas Waktu – $O(n^2)$

Ruang Tambahan – $O(1)$

Di sini semua variabel dinyatakan dalam bingkai global seperti yang ditunjukkan pada gambar di bawah ini



Gambar 10. 2 Parameter Variable Global

3. Kompleksitas Waktu Selection Sort

Kompleksitas jenis digunakan untuk mengekspresikan jumlah waktu eksekusi yang diperlukan untuk mengurutkan daftar. Implementasinya memiliki dua loop. Loop luar yang memilih nilai satu per satu dari daftar dieksekusi n kali di mana n adalah jumlah total nilai dalam daftar. Loop dalam, yang membandingkan nilai dari loop luar dengan sisa nilai, juga dieksekusi n kali di mana n adalah jumlah total elemen dalam daftar. Oleh karena itu, jumlah eksekusi adalah $(n * n)$, yang juga dapat dinyatakan sebagai $O(n)^2$. Jenis seleksi memiliki tiga kategori kompleksitas yaitu;

- a. Kasus terburuk – di sinilah daftar yang disediakan berada dalam urutan menurun. Algoritma ini melakukan jumlah maksimum eksekusi yang dinyatakan sebagai [Big-O] $O(n^2)$
- b. Kasus terbaik – ini terjadi ketika daftar yang disediakan sudah diurutkan. Algoritma melakukan jumlah minimum eksekusi yang dinyatakan sebagai [Big-Omega] $\Omega(n^2)$
- c. Kasus rata-rata – ini terjadi ketika daftar dalam urutan acak. Kompleksitas rata-rata dinyatakan sebagai [Big-theta] $\Theta(n^2)$

Jenis seleksi memiliki kompleksitas ruang $O(1)$ karena membutuhkan satu variabel temporal yang digunakan untuk bertukar nilai.

Kapan menggunakan tipe pilihan? Jenis pilihan paling baik digunakan saat Anda ingin:

- a. Anda harus mengurutkan daftar kecil item dalam urutan naik
- b. Ketika biaya pertukaran nilai tidak signifikan
- c. Hal ini juga digunakan ketika Anda perlu untuk memastikan bahwa semua nilai dalam daftar telah diperiksa.

Keuntungan dari tipe pilihan, Berikut ini adalah keuntungan dari selection sort.

- a. Ini berkinerja sangat baik pada daftar kecil
- b. Ini adalah algoritma di tempat. Tidak membutuhkan banyak ruang untuk menyortir. Hanya satu ruang ekstra yang diperlukan untuk memegang variabel temporal.
- c. Ini berkinerja baik pada item yang telah disortir.

Kelemahan dari jenis seleksi, berikut ini adalah kerugian dari selection sort

- a. Ini berkinerja buruk ketika bekerja pada daftar besar.
- b. Jumlah iterasi yang dibuat selama penyortiran adalah n^2 , di mana n adalah jumlah total elemen dalam daftar.
- c. Algoritma lain, seperti quicksort, memiliki kinerja yang lebih baik dibandingkan dengan jenis seleksi.

4. Implementasi Selection Sort di Python

Implementasi pada bahasa pemrograman Python dicontohkan pada Program Selection sort dibawah ini.

```
4
5 def selectionSort(alist):
6     for fillslot in range(len(alist)-1,0,-1):
7         positionOfMax=0
8         for location in range(1,fillslot+1):
9             if alist[location]>alist[positionOfMax]:
10                 positionOfMax = location
11
12             temp = alist[fillslot]
13             alist[fillslot] = alist[positionOfMax]
14             alist[positionOfMax] = temp
15
16 alist = [54,26,93,17,77,31,44,55,20]
17 selectionSort(alist)
18 print(alist)
```

Hasilnya [17, 20, 26, 31, 44, 54, 55, 77, 93]

Program Urutkan Pilihan

Kode berikut menunjukkan implementasi urutkan pilihan menggunakan

```

1 def selectionSort( itemsList ):
2     n = len( itemsList )
3     for i in range( n - 1 ):
4         minValueIndex = i
5
6         for j in range( i + 1, n ):
7             if itemsList[j] < itemsList[minValueIndex] :
8                 minValueIndex = j
9
10        if minValueIndex != i :
11            temp = itemsList[i]
12            itemsList[i] = itemsList[minValueIndex]
13            itemsList[minValueIndex] = temp
14
15    return itemsList
16
17
18 el = [29,16,8,7,35]
19
20 print(selectionSort(el))
21

```

Menjalankan kode di atas menghasilkan hasil berikut

[7, 8, 16, 29, 35]

Penjelasan Kode

Penjelasan untuk kode adalah sebagai berikut:

```

1  def selectionSort( itemsList ): 1
2  n = len( itemsList ) 2
3  for i in range( n - 1 ): 3
4      minValueIndex = i
5
6      for j in range( i + 1, n ): 4
7          if itemsList[j] < itemsList[minValueIndex] : 5
8              minValueIndex = j 6
9
10     if minValueIndex != i : 7
11         temp = itemsList[i] 8
12         itemsList[i] = itemsList[minValueIndex] 9
13         itemsList[minValueIndex] = temp 10
14
15     return itemsList 11
16
17
18 el = [21,6,9,33,3] 12
19
20 print(selectionSort(el)) 13

```

Berikut penjelasan sourcecode diatas.

- a. Mendefinisikan fungsi bernama selectionSort
- b. Mendapatkan jumlah total elemen dalam daftar. Kita perlu ini untuk menentukan jumlah pass yang akan dibuat ketika membandingkan nilai.
- c. Lingkaran luar. Menggunakan loop untuk iterasi melalui nilai-nilai daftar. Jumlah iterasi adalah $(n - 1)$. Nilai n adalah 5, jadi $(5 - 1)$ memberi kita 4. Ini berarti iterasi luar akan dilakukan 4 kali. Dalam setiap iterasi, nilai variabel i ditugaskan ke variabel minValueIndex
- d. Loop dalam. Menggunakan loop untuk membandingkan nilai paling kiri dengan nilai lain di sisi kanan. Namun, nilai untuk j tidak dimulai pada indeks 0. Dimulai dari $(i + 1)$. Ini tidak termasuk nilai-nilai yang telah diurutkan sehingga kita fokus pada item yang belum diurutkan.
- e. Menemukan nilai minimum dalam daftar yang tidak disortort dan menempatkannya di posisi yang tepat
- f. Memperbarui nilai minValueIndex saat kondisi swapping benar
- g. Membandingkan nilai angka indeks minValueIndex dan saya untuk melihat apakah mereka tidak sama
- h. Nilai paling kiri disimpan dalam variabel temporal
- i. Nilai yang lebih rendah dari sisi kanan mengambil posisi posisi pertama
- j. Nilai yang disimpan dalam nilai temporal disimpan dalam posisi yang sebelumnya dipegang oleh nilai minimum.
- k. Mengembalikan daftar yang diurutkan sebagai hasil fungsi
- l. Membuat daftar el yang memiliki angka acak
- m. Cetak daftar yang diurutkan setelah memanggil fungsi Urutkan pilihan yang lewat di el sebagai parameter.

Ringkasan teknik selection sort dalam pembahasan diaatas adalah sebagai berikut.

- a. Selection sort adalah algoritma perbandingan di tempat yang digunakan untuk mengurutkan daftar acak ke dalam daftar yang dipesan. Ini memiliki kompleksitas waktu $O(n^2)$

- b. Daftar ini dibagi menjadi dua bagian, diurutkan dan tidak diurutkan. Nilai minimum dipilih dari bagian yang tidak diurutkan dan ditempatkan ke bagian yang diurutkan.
- c. Hal ini diulang sampai semua item telah disortir.
- d. Menerapkan pseudocode di Python 3 melibatkan penggunaan dua untuk loop dan jika pernyataan untuk memeriksa apakah swapping diperlukan
- e. Kompleksitas waktu mengukur jumlah langkah yang diperlukan untuk mengurutkan daftar.
- f. Kompleksitas waktu terburuk terjadi ketika daftar berada dalam urutan menurun. Ini memiliki kompleksitas waktu [Big-O] $O(n^2)$
- g. Kompleksitas waktu kasus terbaik terjadi ketika daftar sudah dalam urutan naik. Ini memiliki kompleksitas waktu [Big-Omega] (n^2)
- h. Kompleksitas waktu rata-rata kasus terjadi ketika daftar dalam urutan acak. Ini memiliki kompleksitas waktu [Big-theta] (n^2)
- i. Jenis pilihan paling baik digunakan ketika Anda memiliki daftar kecil item untuk diurutkan, biaya nilai tukar tidak masalah, dan memeriksa semua nilai adalah wajib.
- j. Jenis pilihan tidak berkinerja baik pada daftar besar
- k. Algoritma penyortiran lainnya, seperti quicksort, memiliki kinerja yang lebih baik bila dibandingkan dengan jenis seleksi.

C. LATIHAN SOAL

1. Jelaskan yang anda ketahui teknik selection Sort!
2. Buatlah analogi deret bilangan dengan selection sort dalam 6 angka yang akan dilakukan pengurutan!
3. Jelaskan kelebihan dengan menggunakan selection sort dengan metode sort lainnya!
4. Jelaskan kelemahan dengan menggunakan teknik selection sort pada pengurutan data dalam jumlah besar!

5. Buatlah contoh implementasi teknik selection 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). Pendekripsi Kantuk Secara Real Time Menggunakan Pustaka Opencv Dan Dlib Python. *Sainstech : Jurnal Penelitian Dan Pengkajian Sains Dan Teknologi*.

PERTEMUAN 11

TEKNIK QUICK SORT

A. TUJUAN PEMBELAJARAN

Setelah mempelajari materi pertemuan ini diharapkan mahasiswa mengerti dan paham teknik merge sort, logika dan teknik divide and conquer pada quick sort dan mempu membuat program quick sort dengan python

B. URAIAN MATERI

1. Quick Sort

Dalam arti tertentu, algoritme quicksort adalah kebalikan dari algoritme pengurutan gabungan. Algoritma ini juga yang paling banyak digunakan dan salah satu algoritma penyortiran paling efisien yang dikenal. Quicksort sekali lagi merupakan algoritma divide and conquer dan biasanya ditulis secara rekursif. Tapi, di mana merge sort memisahkan list sampai kita mencapai ukuran 1 dan kemudian merge sort list, algoritma quicksort melakukan penggabungan terlebih dahulu, lalu memisahkan list. Kami tidak bisa menggabungkan list yang tidak disortir. Sebagai gantinya, kami mempartisi menjadi dua list. Apa yang kita inginkan adalah mempersiapkan listnya agar quicksort dapat dipanggil secara rekursif. Tapi, jika kita ingin sukses, kedua sublist harus lebih mudah diurutkan daripada yang asli. Persiapan ini untuk pemisahan disebut pemartision. Untuk mempartisi list kami memilih elemen pivot. Pikirkan quicksort mempartisi list menjadi semua item yang lebih besar dari pivot dan semua elemen lebih kecil dari poros. Kami meletakkan semua barang yang lebih besar di sebelah kanan poros dan semua item yang lebih kecil di sebelah kiri poros. Setelah kita melakukan ini, ada dua hal yang benar:

- a. Poros berada di lokasi akhirnya dalam list.
- b. Kedua sublist sekarang lebih kecil dan oleh karena itu dapat diurutkan dengan cepat.

Begini keduanya sublist diurutkan, ini akan menyebabkan seluruh list menjadi dalam urutan terurut, karena kiri akan menjadi nilai yang naik ke poros, poros

berada di tempat yang benar, dan nilai yang lebih besar dari pivot semuanya akan berada di lokasi yang benar juga.

Quicksort adalah algoritma bagi dan taklukkan. Untuk mendapatkan performa terbaik, kami ingin membagi urutan tepat di tengah menjadi dua berukuran sama urutan. Ini berarti bahwa kita harus memilih nilai persis ditengah menjadi poros karena poros digunakan untuk membagi dua list. Sayangnya ini tidak mungkin jika kita melakukannya dengan efisien. Untuk quicksort memiliki kompleksitas $O(n \log n)$, likemerge sort, itu harus mempartisi list dalam waktu $O(n)$. Kita harus memilih pivot cepat dan kita harus memilihnya dengan baik. Jika kita tidak memilih poros yang dekat dengan tengah, kita tidak akan mendapatkan kompleksitas $O(n \log n)$ yang kita harapkan. Ternyata memilih sebuah poros acak dari list sudah cukup baik. Salah satu cara untuk menjamin pilihan acak pivotnya adalah membuat algoritme quicksort dimulai dengan mengacak urutannya.



Gambar 11.1 Ilustrasi Quick Sort

Quick Sort dimana algoritma pengurutan datanya menggunakan teknik pemecahan data menjadi partisi-partisi, sehingga metode ini disebut juga dengan nama partition exchange sort. Untuk memulai iterasi pengurutan, pertama-tama sebuah elemen dipilih dari data, kemudian elemen-elemen data

akan diurutkan diatur sedemikian rupa, sehingga nilai variabel *Sementara* berada di suatu posisi ke *I* yang memenuhi kondisi sebagai berikut :

- a. Semua elemen di posisi ke 1 sampai dengan ke *I*-1 adalah lebih kecil atau sama dengan *Sementara*.
- b. Semua elemen di posisi ke *I*+1 sampai dengan ke *N* adalah lebih besar atau sama dengan *Sementara*.

2. Logika Quick Sort

Algoritma tipe cepat adalah algoritma penyortiran di tempat tanpa perlu ruang ekstra atau array tambahan karena semua operasi akan dilakukan dalam daftar yang sama, karena kami membagi daftar yang diberikan menjadi tiga bagian sebagai elemen pivot, elemen kurang dari pivot sebagai satu sub-daftar dan elemen yang lebih besar dari pivot sebagai sub-daftar lain. Ini juga disebut sebagai jenis pertukaran partisi. Quicksort adalah algoritma penyortiran yang lebih baik dan, di sebagian besar bahasa pemrograman, tersedia sebagai algoritma penyortiran bawaan.

Diberikan di bawah ini adalah langkah-langkah utama untuk logika implementasi tipe cepat:

- a. Pertama, kita akan memilih elemen pivot dari daftar yang diberikan.
 - 1) Kita dapat memilih elemen pivot dengan cara yang berbeda seperti di bawah ini:
 - 2) Kita dapat memilih elemen pertama dari daftar sebagai pivot.
 - 3) Kita dapat memilih elemen terakhir dari daftar sebagai pivot.
 - 4) Kita dapat memilih beberapa elemen acak dari daftar sebagai pivot.
 - 5) Akhirnya, kita dapat memilih median elemen daftar sebagai pivot.
- b. Dalam partisi, kami akan mengatur ulang daftar sedemikian rupa sehingga semua elemen daftar yang kurang dari pivot akan berada di sisi kiri, dan semua elemen daftar yang lebih besar dari pivot akan berada di sisi kanan, dan elemen yang sama akan berada di salah satu sisi pivot. Proses ini disebut partisi. Setelah akhir proses partisi, elemen pivot akan berada di posisi akhir pada daftar.

- c. Kami akan menerapkan langkah-langkah di atas secara rekursif pada kedua sub-array sampai semua sub-array diurutkan.

Contoh 1 penggambaran logika sebagai berikut:

Ambil daftar dengan 6 elemen sebagai 9 7 15 10 2 5. Di sini akan mengambil elemen pertama dari daftar sebagai elemen pivot dan memulai daftar dan mengakhiri daftar.

Langkah 1: Pivot = 9 mulai = 9 ujung = 5

9 7 15 10 2 5

Sekarang kita akan memanggil proses partisi pada daftar yang diberikan, dan kita akan mendapatkan daftar yang diatur ulang dengan elemen pivot berada di posisi akhir seperti di bawah ini:

5 7 2 9 15 10

Seperti yang kita lihat elemen pivot berada dalam posisi terurutkan terakhirnya. Sekarang kita akan menerapkan langkah-langkah yang sama ke sub-daftar kiri dan kanan elemen pivot.

Langkah 2: pivot = 5 start = 5 end = 2 (sub-list kiri)

2 5 7

pivot = 15 start = 15 end = 10 (sub-list kanan)

10 15

Pada langkah di atas, kami telah memanggil metode partisi pada sub-daftar kiri dan kanan, dan kami mengatur ulang seperti di bawah ini:

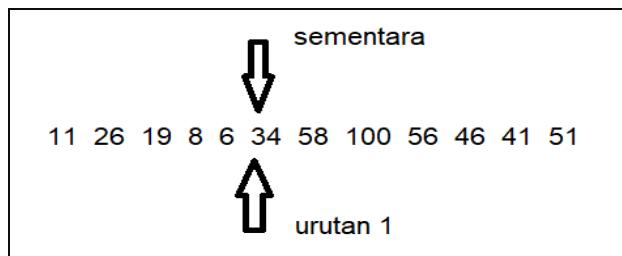
2 5 7 9 10 15

Jika mengamati daftar yang kita dapatkan di langkah di atas, semua elemen berada dalam posisi yang diurutkan. Dengan mengikuti logika di atas, kita dapat menerapkan jenis cepat, dan ini adalah salah satu cara untuk menerapkan jenis cepat dengan kompleksitas waktu kasus rata-rata $O(N \log N)$ dan kompleksitas waktu terburuk menjadi $O(n^2)$. Jika kita mengamati hal di atas, penyortiran terjadi di tempat tanpa menggunakan ruang ekstra.

Sebagai contoh 2, data yang akan diurutkan sejumlah 12 elemen sebagai berikut.

34 46 19 8 6 100 58 26 56 11 41 51

Misalnya element yang dipilih adalah element yang pertama, maka variabel Sementara bernilai 34. Setelah diatur, maka nilai 34 akan menempati posisi ke 1, yaitu posisi urutan ke 6 sebagai berikut :



Tampak bahwa kondisi berikut terpenuhi, yaitu :

- a. Semua elemen di posisi ke 1 sampai dengan posisi ke 5 (11, 26, 19, 8, dan 6) akan lebih kecil atau sama dengan nilai 34 yang dipilih.
- b. Semua elemen di posisi ke 7 sampai dengan ke 12 (58, 100, 56, 46, 41 dan 51) aka lebih besar atau sama dengan nilai 34 yang dipilih.

Dengan demikian, data tersebut akan terpecah menjadi 2 partisi, sebagai berikut :

(11 26 19 8 6) 34 (58,100,56,46,41 51)

Proses ini diulangi kembali untuk masing-masing partisi data, yaitu untuk data (11 26 19 8 6) dan data (58,100,56,46,41 51). Untuk partisi yang pertama, bila nilai Sementara yang diambil adalah data pertama kembali dalam partisi bersangkutan, yaitu 10 dan diatur kembali sedemikian rupa, maka nilai data yang dipilih akan terletak di posisi sebagai berikut:

(6 8) 11 (19 26) 34 (58 100 56 46 41 51)

Untuk mengurutkan masing-masing partisi, maka proses tersebut diulangi kembali dan tiap-tiap partisi dipecah-pecah kembali lebih lanjut. Kurung yang menutupi partisi menunjukkan data yang belum urut dan perlu diurutkan kembali. Sedang data yang tidak berada diantara tanda kurung merupakan data

yang sudah diurut. Iterasi selanjutnya sampai didapatkan data yang telah urut semuanya adalah sebagai berikut ini.

6 (8) 11 (19 26) 34 (58 100 56 46 41 51)

6 8 11 19 (26) 34 (58 100 56 46 41 51)

6 8 11 19 26 34 (51 41 56 46) 58 (100)

6 8 11 19 26 34 (51 41 56 46) 58 100

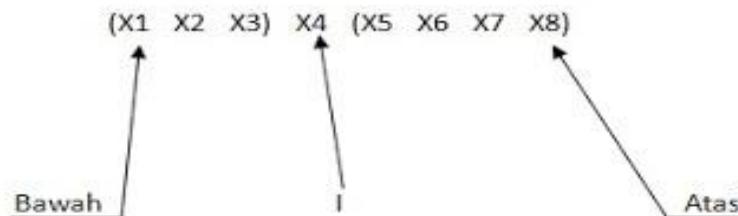
6 8 11 19 26 34 (46 41) 51 (56) 58 100

6 8 11 19 26 34 (46 41) 51 56 58 100

6 8 11 19 26 34 41 (46) 51 56 58 100

6 8 11 19 26 34 41 46 51 56 58 100

Bila diamati lebih lanjut, maka quick sort dapat didefinisikan dengan lebih mudah menggunakan prosedur rekursi. Misalnya untuk partisi sebagai berikut



Maka Proses dari rekursi terlihat dalam pengurutan data dari bawah sampai I-1 dan pengurutan I+1 sampai atas.

Setelah list diacak, memilih poros acak menjadi lebih mudah. Partisi fungsi mengambil item pertama dalam urutan sebagai poros. Partisi dimulai dari kedua ujungnya dan bekerja sampai ke tengah. Intinya setiap kali nilainya semakin besar daripada pivot ditemukan di sisi kiri dan nilai yang lebih kecil dari pivot ditemukan di sisi kanan, kedua nilai tersebut ditukar. Begitu kita mencapai tengah dari keduanya sisi, poros ditukar ke tempatnya. Setelah urutan dipartisi, quicksort

algoritma dipanggil secara rekursif pada dua bagian. Variabel i dan j adalah indeks dari nilai kiri dan kanan, masing-masing, selama proses partisi.

Jika Anda melihat kode partisi, keduanya berkomentar sementara kondisi loop sedang mungkin lebih mudah dipahami daripada kode yang tidak diberi komentar. Namun, yang tidak diberi komentar kode hanya menggunakan operator less than. Quicksort adalah algoritma pengurutan yang digunakan dengan metode sortir pada list. Ini hanya membutuhkan operator kurang dari yang didefinisikan antar item dalam urutan. Dengan menulis dua while loop seperti yang kita miliki, satu-satunya pemesanan yang diperlukan didefinisikan oleh operator less than seperti yang disyaratkan Python. Potret pada Gambar 4.7 menunjukkan efek partisi pada suatu urutan. Di dalam gambar, urutan telah dipartisi dua kali. Partisi pertama dipilih poros yang hampir mati. Namun, partisi kedua memilih poros itu tidak terlalu bagus. Garis merah menunjukkan bagian dari urutan yang sekarang sedang dipartisi. Lihat bagaimana nilai paling kiri di sub-urutan itu menjadi nilai pivot. Dua titik hijau adalah nilai pivot yang sudah berada di lokasi yang benar. Semua nilai di atas poros akan berakhir di partisi di sebelah kanan poros dan semuanya nilai di sebelah kiri pivot lebih kecil dari pivot. Inilah sifat quicksort. Sekali lagi, dengan kompleksitas yang diamortisasi kita dapat menemukan bahwa algoritma quicksort berjalan $O(n \log n)$ waktu. Pertimbangkan untuk menyortir list [5 8 2 6 9 1 0 7] menggunakan quicksort. Gambar 4.8 menggambarkan list setelah setiap panggilan ke fungsi partisi. Pivot di setiap panggilan diidentifikasi oleh item berwarna oranye. Fungsi partisi mempartisi list yang meluas ke kanan dari porosnya. Setelah dipartisi, pivot dipindahkan ke lokasi akhirnya dengan menukar itu dengan item terakhir yang kurang dari poros. Kemudian partisi dilakukan pada file menghasilkan dua sublist.

Pengacakan yang dilakukan pada langkah pertama quicksort membantu memilih yang lebih acak nilai poros. Ini memiliki konsekuensi nyata dalam algoritme quicksort, terutama saat urutan yang diteruskan ke quicksort memiliki peluang untuk diurutkan. Jika berurutan diberikan ke quicksort diurutkan, atau hampir diurutkan, baik dalam ascending atau descending order, maka quicksort tidak akan mencapai kompleksitas $O(n \log n)$. Faktanya, kasus terburuk kompleksitas algoritma adalah $O(n^2)$. Jika poros yang dipilih adalah yang terkecil atau terbesar berikutnya nilai, maka partisi tidak akan membagi masalah menjadi sublist yang lebih kecil sebagai terjadi ketika 9 dipilih sebagai poros untuk sublist pada Gambar 4.8. Algoritme akan cukup letakkan satu nilai pada tempatnya dan akhiri dengan satu partisi besar dari semua yang lainnya

nilai. Jika ini terjadi setiap kali pivot dipilih, itu akan mengarah ke $O(n^2)$ kompleksitas. Mengacak daftar sebelum quicksorting akan membantu untuk memastikan bahwa ini tidak terjadi.

Urutan penggabungan tidak dipengaruhi oleh pilihan pivot, karena tidak ada pilihan yang diperlukan. Oleh karena itu, jenis gabungan tidak memiliki kasus pertama atau kasus terbaik untuk dipertimbangkan. Itu akan selalu mencapai kompleksitas $O(n \log n)$. Meski begitu, quicksort berkinerja lebih baik dalam praktiknya daripada merge sort karena algoritme quicksort tidak perlu menyalin ke daftar baru, lalu kembali lagi. Algoritme quicksort adalah standar de facto untuk algoritme pengurutan.

Algoritma penyortiran quicksort yang mengikuti algoritma divide and conquer. Pertama, kita akan belajar apa itu membagi dan menaklukkan algoritma.

3. Teknik Divide and Conquer

Divide and Conquer adalah paradigma algoritmik yang sama dengan Greedy dan Dynamic Programming. Algoritma membagi dan menaklukkan standar mengikuti tiga langkah untuk memecahkan masalah.

- a. Bagi: Pecahkan masalah yang diberikan menjadi subproblems yang termasuk dalam jenis yang sama.
- b. Conquer: Selesaikan subproblems secara rekursif.
- c. Gabungkan: Gabungkan semua subproblems di akhir untuk mendapatkan jawabannya.

Algoritma Quicksort memilih elemen sebagai pivot dan partisi array yang diberikan di sekitar elemen pivot yang dipilih. Ada banyak cara elemen pivot dapat dipilih. Selalu pilih elemen pertama sebagai pivot.

- a. Selalu pilih elemen terakhir sebagai pivot.
- b. Pilih elemen acak sebagai pivot.
- c. Pilih elemen tengah atau median sebagai pivot.

Setelah memilih elemen pivot, algoritma mengatur ulang sedemikian rupa sehingga semua elemen yang lebih kecil dari elemen pivot yang dipilih bergeser

ke sisi kiri elemen pivot dan semua elemen yang lebih besar dari elemen pivot yang dipilih bergeser ke sisi kanan elemen pivot. Akhirnya, algoritma secara rekursif mengurutkan subarrays di sisi kiri dan kanan elemen pivot.

4. Quicksort dengan Python

Dalam kehidupan nyata, kita harus selalu menggunakan jenis builtin yang disediakan oleh Python. Namun, memahami algoritma quicksort adalah instruktif.

Tujuan di sini adalah untuk memecah subjek sehingga mudah dipahami dan ditiru oleh pembaca tanpa harus kembali ke bahan referensi. Algoritma quicksort pada dasarnya adalah sebagai berikut:

- a. Pilih titik data pivot.
- b. Pindahkan semua titik data kurang dari (di bawah) pivot ke posisi di bawah pivot - pindahkan yang lebih besar dari atau sama dengan (di atas) pivot ke posisi di atasnya.
- c. Terapkan algoritma ke area di atas dan di bawah pivot
- d. Jika data didistribusikan secara acak, pilih titik data pertama sebagai pivot setara dengan pilihan acak.

Implementasi sebelumnya menciptakan banyak daftar tambahan yang tidak perlu. Jika kita bisa melakukan ini di tempat, kita akan menghindari membuang-buang ruang.

Algoritma ini sering diajarkan dalam kursus ilmu komputer dan diminta untuk wawancara kerja. Ini membantu kita berpikir tentang rekursi dan membagi-dan-menaklukkan. Quicksort tidak terlalu praktis dalam Python karena algoritma timsort builtin cukup efisien, dan kami memiliki batas rekursi. Kami berharap untuk mengurutkan daftar di tempat dengan list.sort atau membuat daftar yang diurutkan baru dengan diurutkan yang keduanya mengambil dan argumen

Dalam python, quick sort didefinisikan sebagai algoritma pemilahan yang mengikuti metode divide and conquers untuk memilih elemen pivot berdasarkan mana array yang tersisa akan dibagi menjadi dua elemen sub-array yang kurang dari pivot akan berada di sub-array kiri dan elemen yang lebih besar dari

pivot akan berada di sub-array kanan dan proses akan berulang secara rekursif sampai semua sub-array diurutkan tanpa menggunakan array tambahan atau ekstra. Ruang disebut quick sort. Quick sort adalah algoritma penyortiran yang efisien dan paling banyak digunakan yang lebih baik daripada algoritma serupa jika diimplementasikan dengan baik. Ini memiliki kompleksitas waktu kasus rata-rata $O(N \log N)$, dan kompleksitas waktu terburuk adalah $O(n^2)$.

```
4 def quickSort(alist):
5     quickSortHelper(alist,0,len(alist)-1)
6
7 def quickSortHelper(alist,first,last):
8     if first<last:
9
10        splitpoint = partition(alist,first,last)
11
12        quickSortHelper(alist,first,splitpoint-1)
13        quickSortHelper(alist,splitpoint+1,last)
```

```
16 def partition(alist,first,last):
17     pivotvalue = alist[first]
18
19     leftmark = first+1
20     rightmark = last
21
22     done = False
23     while not done:
24
25         while leftmark <= rightmark and \
26             alist[leftmark] <= pivotvalue:
27             leftmark = leftmark + 1
28
29         while alist[rightmark] >= pivotvalue and \
30             rightmark >= leftmark:
31             rightmark = rightmark -1
32
33         if rightmark < leftmark:
34             done = True
35         else:
36             temp = alist[leftmark]
37             alist[leftmark] = alist[rightmark]
38             alist[rightmark] = temp
39
40         temp = alist[first]
41         alist[first] = alist[rightmark]
42         alist[rightmark] = temp
43
44
45     return rightmark
46
47 alist = [54,26,93,17,77,31,44,55,20]
48 quickSort(alist)
49 print(alist)
```

Hasilnya jika di run,

[17, 20, 26, 31, 44, 54, 55, 77, 93]

Contoh 2 implementasi Python dengan teknik quick sort

```

1 def quickSort(alist):
2     quickSortHelper(alist,0,len(alist)-1)
3
4 def quickSortHelper(alist,first,last):
5     if first<last:
6         splitpoint = partition(alist,first,last)
7         quickSortHelper(alist,first,splitpoint-1)
8         quickSortHelper(alist,splitpoint+1,last)
9     def partition(alist,first,last):
10        pivotvalue = alist[first]
11        leftmark = first+1
12        rightmark = last
13        done = False
14        while not done:
15            while leftmark <= rightmark and alist[leftmark] <= pivotvalue:
16                leftmark = leftmark + 1
17            while alist[rightmark] >= pivotvalue and rightmark >= leftmark:
18                rightmark = rightmark -1
19            if rightmark < leftmark:
20                done = True
21            else:
22                temp = alist[leftmark]
23                alist[leftmark] = alist[rightmark]
24                alist[rightmark] = temp
25                temp = alist[first]
26                alist[first] = alist[rightmark]
27                alist[rightmark] = temp
28        return rightmark
29
30 alist = [55,20,9,27,37,31,64,15,50]
31 quickSort(alist)
32 print(alist)
33

```

Hasilnya

[9, 20, 15, 27, 31, 37, 50, 64, 55]

Pada Quick Sort di atas adalah pivot yang diambil adalah data paling kiri. Berikut Quick Sort dengan pivot yang digunakan adalah data yang terakhir. contoh 3 implementasi quick sort pada python sebagai berikut.

```
1 def quickSort(alist):
2     quickSortHelper(alist,0,len(alist)-1)
3
4 def quickSortHelper(alist,first,last):
5     if first<last:
6         splitpoint = partition(alist,first,last)
7         quickSortHelper(alist,first,splitpoint-1)
8         quickSortHelper(alist,splitpoint+1,last)
9 def partition(alist,first,last):
10    pivotvalue = alist[last]
11    leftmark = first
12    rightmark = last-1
13    done = False
14    while not done:
15        while leftmark <= rightmark and alist[leftmark] <= pivotvalue:
16            leftmark = leftmark + 1
17        while alist[rightmark] >= pivotvalue and rightmark >= leftmark:
18            rightmark = rightmark -1
19        if rightmark < leftmark:
20            done = True
21        else:
22            temp = alist[leftmark]
23            alist[leftmark] = alist[rightmark]
24            alist[rightmark] = temp
25            temp = alist[last]
26            alist[last] = alist[leftmark]
27            alist[leftmark] = temp
28    return leftmark
29 alist = [55,20,9,27,37,31,64,15,50]
30 quickSort(alist)
31 print(alist)
32
33 |
```

Hasilnya

[50, 15, 9, 20, 27, 31, 37, 55, 64]

Contoh 4 quick Sort dengan pivot ditengah

```
1 def quickSort(alist):
2     print('Dengan Quick Sort')
3     quickSortHelper(alist,0,len(alist)-1)
4 def quickSortHelper(alist,start,end):
5     if start >= end:
6         return
7     i,j = start, end
8     pivot = (start + (end - start)//2)
9     while i<=j:
10        while(alist[i] < alist[pivot]):
11            i+=1
12        while(alist[j] > alist[pivot]):
13            j-=1
14        if(i<=j):
15            alist[i],alist[j] = alist[j],alist[i]
16            i+=1
17            j-=1
18    quickSortHelper(alist,start,j)
19    quickSortHelper(alist,i,end)
20    return alist
21
22 alist = [55,20,9,27,37,31,64,15,50]
23
24 quickSort(alist)
25
26 print(alist)
27
```

Hasilnya

Dengan Quick Sort

[9, 15, 20, 27, 31, 37, 50, 55, 64]

Akhirnya, ini semua tentang algoritma tipe cepat dalam python. Sejauh ini, kita telah melihat definisi jenis cepat, logika di balik implementasi tipe cepat dengan penjelasan langkah demi langkah, dan seberapa cepat penyortiran dapat diimplementasikan menggunakan berbagai metode python dengan contoh dan output yang sesuai.

C. LATIHAN SOAL

1. Jelaskan yang anda ketahui teknik quick Sort!
2. Buatlah analogi deret bilangan dengan quick sort dalam 6 angka yang akan dilakukan pengurutan!
3. Jelaskan teknik Teknik Divide and Conquer pada quick sort!
4. Jelaskan Bagaimana pengecekan kondisi diawal antrian
5. Buatlah contoh implementasi teknik quick sort dengan bahasa python!

D. REFERENSI

- Distance Untuk Deteksi Kemiripan Gambar. *Repository Its.*
- Basant Agarwal, B. B. (2018). *Hand-On Data Structures And Algorithms With Python*. London: Packt Publishing.
- Emi Sita Eriana, A. Z. (2021). Praktikum Algoritma Dan Pemrograman. Tangerang Selatan: Unpam Press.
- Jodi, U. R. (2020). Algoritma Dan Struktur Data.
- 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.
- 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). Pendekripsi Kantuk Secara Real Time Menggunakan Pustaka Opencv Dan Dlib Python. Sainstech : Jurnal Penelitian Dan Pengkajian Sains Dan Teknologi.
- Zein, A. (2019). Pendekripsi Penyakit Malaria Menggunakan Medical Images Analisis Dengan Deep Learning Python. Jurnal Penelitian Dan Pengkajian Sains Dan Teknologi.

PERTEMUAN 12

STACK

A. TUJUAN PEMBELAJARAN

Setelah mempelajari materi pertemuan ini diharapkan mahasiswa mengerti dan paham stack, proses LIFO, dan ampu membuat program stack dalam Python

B. URAIAN MATERI

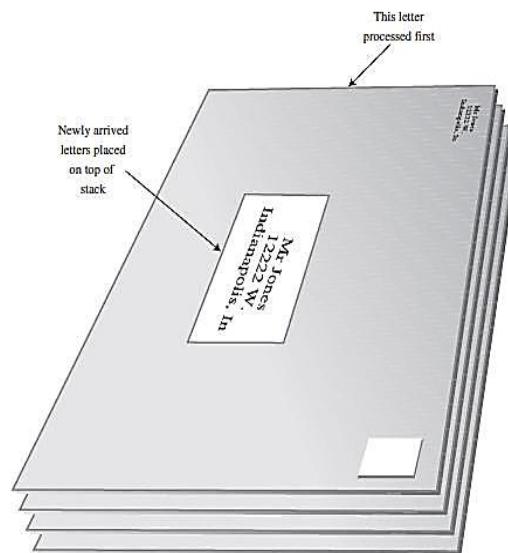
1. Stack

Stack atau tumpukan memungkinkan akses hanya ke satu item data: item terakhir yang dimasukkan. Jika Anda menghapus ini item, Anda dapat mengakses item di sebelah terakhir yang disisipkan, dan seterusnya. Kemampuan ini berguna dalam banyak situasi pemrograman. Di bagian ini kita akan melihat bagaimana tumpukan bisa digunakan untuk memeriksa apakah tanda kurung di komputer file sumber program. Tumpukan memainkan peran penting dalam menganalisis ekspresi aritmatika seperti $3 * (4 + 5)$. Tumpukan juga merupakan bantuan praktis untuk algoritme yang diterapkan pada struktur data kompleks tertentu.

Kebanyakan mikroprosesor menggunakan arsitektur berbasis tumpukan. Ketika sebuah metode dipanggil, itu alamat pengirim dan argumen didorong ke tumpukan, dan ketika dikembalikan, mereka muncul. Operasi stack dibangun ke dalam mikroprosesor. Beberapa kalkulator saku yang tua menggunakan arsitektur berbasis tumpukan. Bukannya masuk ekspresi aritmatika menggunakan tanda kurung mendorong hasil antara ke sebuah stack atau tumpukan.

Banyak orang, ketika mereka menerima surat mereka, melemparkannya ke tumpukan di meja aula atau ke dalam keranjang "dalam" di tempat kerja. Kemudian, ketika mereka punya waktu luang, mereka memproses kumpulan surat dari atas ke bawah. Pertama, mereka membuka huruf di atas tumpukan dan mengambil tindakan yang sesuai membayar tagihan, membuangnya, atau apa pun. Setelah surat pertama dibuang, mereka memeriksa surat berikutnya,

yang sekarang menjadi bagian atas tumpukan, dan tangani itu. Akhirnya, turun ke di bagian bawah tumpukan (yang sekarang menjadi paling atas).



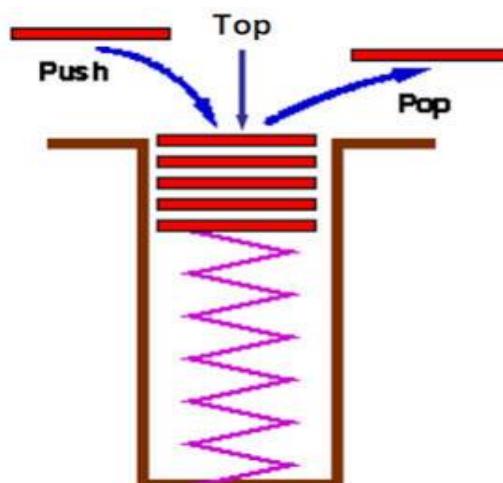
Gambar 12. 1 Ilustrasi Tumpukan Surat

Pendekatan "lakukan yang teratas dulu" ini bekerja dengan baik selama dapat memprosesnya dengan mudah semua email dalam waktu yang wajar. Jika tidak bisa, ada bahaya bahwa huruf-huruf itu ada dibagian bawah tumpukan tidak akan diperiksa selama berbulan-bulan, dan tagihan di dalamnya akan menjadi terlambat. Tentu saja, banyak orang tidak mengikuti pendekatan dari atas ke bawah ini dengan ketat. Misalnya, mengambil surat dari dasar tumpukan, untuk memproses surat tertua dulu. Atau mereka mungkin mengacak email sebelum mulai memproses dan letakkan huruf dengan prioritas lebih tinggi di atasnya. Dalam kasus ini, sistem email mereka tidak lagi tumpukan dalam arti ilmu komputer dari kata tersebut. Jika mereka mengambil surat dari bawah, itu antrian; dan jika mereka memprioritaskannya, itu adalah antrean prioritas. Kami akan melihat ini kemungkinan nanti.

Analogi tumpukan lainnya adalah tugas yang lakukan selama hari kerja biasa. Kamu sibuk pada proyek jangka panjang (A), tetapi diinterupsi oleh rekan kerja yang meminta bantuan sementara untuk proyek lain (B). Saat mengerjakan B, seseorang di akuntansi mampir untuk pertemuan tentang biaya perjalanan (C), dan selama pertemuan ini, anda mendapat panggilan darurat dari seseorang di bagian penjualan dan menghabiskan beberapa menit untuk memecahkan masalah produk besar (D). Setelah selesai dengan

panggilan D, Anda melanjutkan rapat C; ketika Anda selesai dengan C, Anda melanjutkan proyek B, dan ketika Anda selesai dengan B, Anda bisa (akhirnya!) kembali ke proyek A. Proyek dengan prioritas lebih rendah "ditumpuk" menunggu Anda untuk kembali kepada mereka.

Menempatkan item data di atas tumpukan disebut mendorongnya. Menghapusnya dari bagian atas tumpukan disebut popping itu. Ini adalah operasi tumpukan utama. Tumpukan adalah dikatakan sebagai mekanisme penyimpanan Last-In-First-Out (LIFO) karena item terakhir dimasukkan adalah yang pertama disingkirkan. Dalam tumpukan, elemen baru ditambahkan di satu ujung dan elemen dihapus dari ujung itu saja. Operasi insert dan delete sering disebut push and pop.



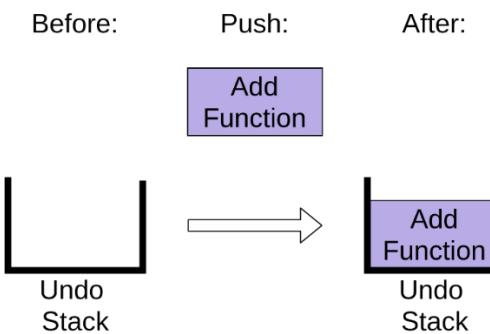
Gambar 12. 2 Operasi Stack

- a. kosong() – Mengembalikan apakah tumpukan kosong
- b. ukuran() – Mengembalikan ukuran tumpukan
- c. top() – Mengembalikan referensi ke elemen paling atas tumpukan
- d. push(a) – Menyisipkan elemen 'a' di bagian atas
- e. pop() – Menghapus elemen paling atas dari tumpukan

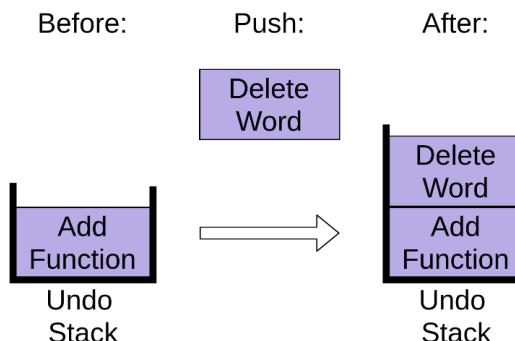
Ada berbagai cara dari mana tumpukan dapat diimplementasikan dalam Python. Artikel ini mencakup implementasi tumpukan menggunakan struktur data dan modul dari perpustakaan Python. Stack di Python dapat diimplementasikan dengan menggunakan cara-cara berikut:

- a. list
- b. Collections.deque
- c. LifoQueue

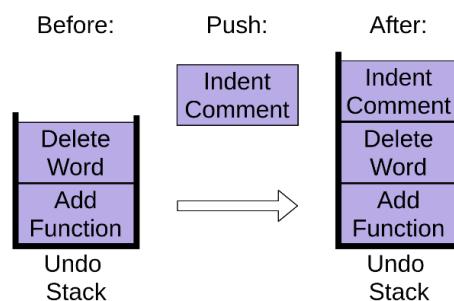
Mungkin paling mudah untuk memahami tumpukan jika, dengan studi kasus penggunaan yang dikenal misalnya : fitur Undo di editor Anda. Pada saat mengedit file Python sehingga dapat melihat beberapa operasi yang di lakukan. Pertama, menambahkan fungsi baru. Ini menambahkan item baru ke tumpukan undo:



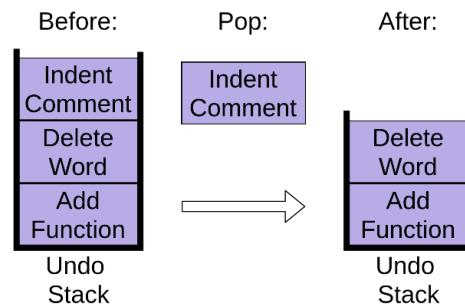
Dapat dilihat bahwa tumpukan sekarang memiliki operasi Tambahkan Fungsi di atasnya. Setelah menambahkan fungsi, dapat menghapus kata dari komentar. Ini juga akan ditambahkan ke tumpukan.



Perhatikan bagaimana item Hapus Word ditempatkan di atas tumpukan. Akhirnya inden komentar sehingga berbaris dengan benar:

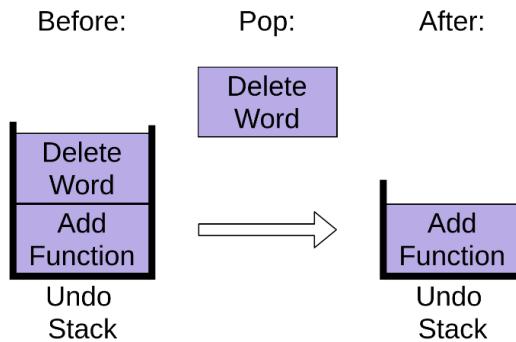


Dapat dilihat bahwa masing-masing perintah ini disimpan dalam tumpukan undo, dengan setiap perintah baru diletakkan di bagian atas. Saat bekerja dengan tumpukan, menambahkan item baru seperti ini disebut `.push`. Sekarang telah memutuskan untuk membatalkan ketiga perubahan tersebut, jadi jika Anda menekan perintah `undo`. Dibutuhkan item di bagian atas tumpukan, yang indentasi komentar, dan menghapusnya dari tumpukan:

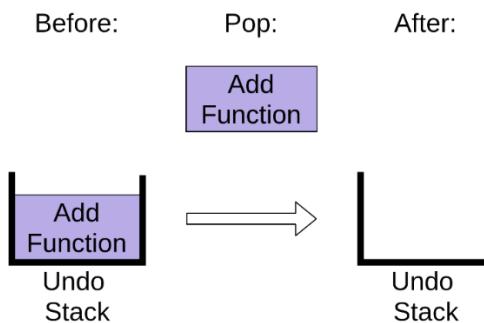


Editor akan membatalkan inden, dan tumpukan undo sekarang berisi dua item. Operasi ini adalah kebalikan dari dan biasa disebut `.pushpop`

Saat Anda menekan `undo` lagi, item berikutnya muncul dari tumpukan:



Ini menghapus item `Hapus Word`, hanya menyisakan satu operasi pada tumpukan. Akhirnya, jika Anda menekan `Undo` untuk ketiga kalinya, maka item terakhir akan muncul dari tumpukan:



Tumpukan undo sekarang kosong. Menekan Undo lagi setelah ini tidak akan berpengaruh karena tumpukan undo Anda kosong, setidaknya di sebagian besar editor. Anda akan melihat apa yang terjadi ketika Anda memanggil tumpukan kosong dalam deskripsi implementasi di bawah ini..pop()

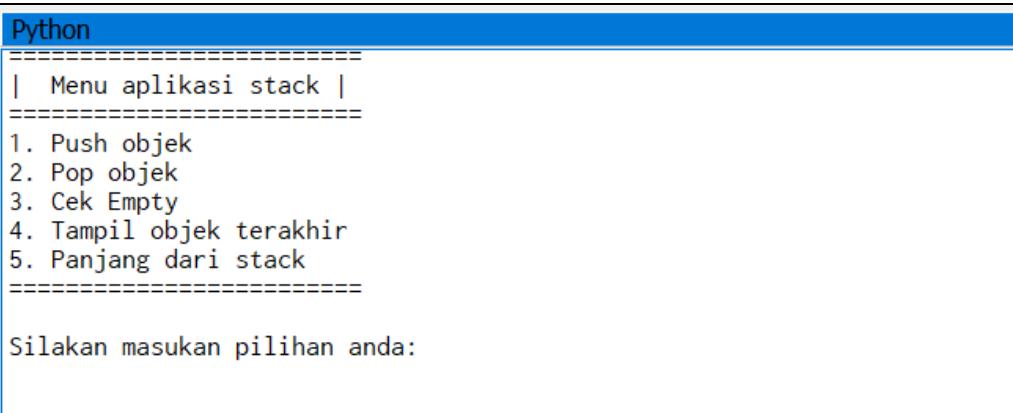
```
1 # Import library os untuk membersihkan layar console
2 import os
3
4 # Class stack
5 class Stack:
6     def __init__(self):
7         self.items = []
8
9     # Memeriksa apakah stack kosong
10    def isEmpty(self):
11        return self.items == []
12    # Menambah objek/data ke dalam stack
13    def push(self, item):
14        self.items.append(item)
15    # Mengeluarkan data dari stack
16    def pop(self):
17        return self.items.pop()
18    # Menampilkan objek terakhir dari stack
19    def peek(self):
20        return self.items[len(self.items)-1]
21    # Menghitung panjang stack
22    def size(self):
23        return len(self.items)
24
25    # Menu dari aplikasi
26    def mainmenu(self):
27        pilih = "y"
28        while (pilih == "y"):
29            os.system("clear")
30            print("=====")
```

```

31     print(" | Menu aplikasi stack |")
32     print("=====")
33     print("1. Push objek")
34     print("2. Pop objek")
35     print("3. Cek Empty")
36     print("4. Tampil objek terakhir")
37     print("5. Panjang dari stack")
38     print("=====")
39     pilihan=str(input("Silakan masukan pilihan anda: "))
40     if(pilihan=="1"):
41         os.system("clear")
42         obj = str(input("Masukan objek yang ingin anda tambahkan: "))
43         self.push(obj)
44         print("Object "+obj+" telah ditambahkan")
45         x = raw_input("")
46     elif(pilihan=="2"):
47         os.system("clear")
48         print("Objek "+self.pop()+" dihapus")
49         x = raw_input("")
50     elif(pilihan=="3"):
51         os.system("clear")
52         print(self.isEmpty())
53         x = raw_input("")
54     elif(pilihan=="4"):
55         os.system("clear")
56         print("Objek terakhir: "+self.peek())
57         x = raw_input("")
58     elif(pilihan=="5"):
59         os.system("clear")
60         print("Panjang dari stack adalah: "+str(self.size()))
61         x = raw_input("")
62     else:
63         pilih="n"
64
65 if __name__ == "__main__":
66     s=Stack()
67     s.mainmenu()
68

```

Hasil jika di run :



The screenshot shows a Python terminal window with the title bar "Python". Inside the terminal, the application displays its main menu:

```

Python
=====
| Menu aplikasi stack |
=====
1. Push objek
2. Pop objek
3. Cek Empty
4. Tampil objek terakhir
5. Panjang dari stack
=====

Silakan masukan pilihan anda:

```

2. Implementasi Menggunakan List

List struktur data bawaan Python dapat digunakan sebagai tumpukan. Alih-alih push(), append() digunakan untuk menambahkan elemen ke bagian atas tumpukan sementara pop() menghapus elemen dalam urutan LIFO. Sayangnya, list ini memiliki beberapa kekurangan. Masalah terbesar adalah bahwa hal itu dapat mengalami masalah kecepatan saat tumbuh. Item dalam list disimpan bersebelahan dalam memori, jika tumpukan tumbuh lebih besar dari blok memori yang saat ini memegangnya, maka Python perlu melakukan beberapa alokasi memori. Hal ini dapat menyebabkan beberapa tambahan () panggilan mengambil lebih lama dari yang lain.

```

1 # Python program to
2 # demonstrate stack implementation
3 # using list
4
5 stack = []
6
7 # append() function to push
8 # element in the stack
9 stack.append('a')
10 stack.append('b')
11 stack.append('c')
12
13 print('Initial stack')
14 print(stack)
15
16 # pop() function to pop
17 # element from stack in
18 # LIFO order
19 print('\nElements popped from stack:')
20 print(stack.pop())
21 print(stack.pop())
22 print(stack.pop())
23
24 print('\nStack after elements are popped:')
25 print(stack)
26
27 # uncommenting print(stack.pop())
28 # will cause an IndexError
29 # as the stack is now empty
30
31

```

Hasil jika di run :

```

Python
Initial stack
['a', 'b', 'c']

Elements popped from stack:
c
b
a

Stack after elements are popped:
[]

In [2]: |

```

3. Implementasi Collections.Deque

Python stack dapat diimplementasikan menggunakan kelas deque dari modul koleksi. Deque lebih disukai daripada list dalam kasus-kasus di mana kita membutuhkan tambahan dan operasi pop yang lebih cepat dari kedua ujung wadah, karena deque memberikan kompleksitas waktu $O(1)$ untuk operasi tambahan dan pop dibandingkan dengan list yang menyediakan $O(n)$ kompleksitas waktu.

Metode yang sama pada deque seperti yang terlihat dalam list digunakan, append() dan pop().

```

1 # Python program to
2 # demonstrate stack implementation
3 # using collections.deque
4
5 from collections import deque
6
7 stack = deque()
8
9 # append() function to push
10 # element in the stack
11 stack.append('a')
12 stack.append('b')
13 stack.append('c')
14
15 print('Initial stack: ')
16 print(stack)
17
18 # pop() function to pop
19 # element from stack in
20 # LIFO order
21 print('\nElements popped from stack: ')
22 print(stack.pop())
23 print(stack.pop())
24 print(stack.pop())
25
26 print('\nStack after elements are popped: ')
27 print(stack)
28
29 # uncommenting print(stack.pop())
30 # will cause an IndexError
31 # as the stack is now empty
32

```

```

Python
Initial stack:
deque(['a', 'b', 'c'])

Elements popped from stack:
c
b
a

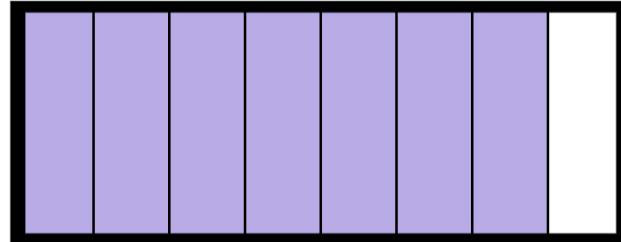
Stack after elements are popped:
deque([])

In [17]: |

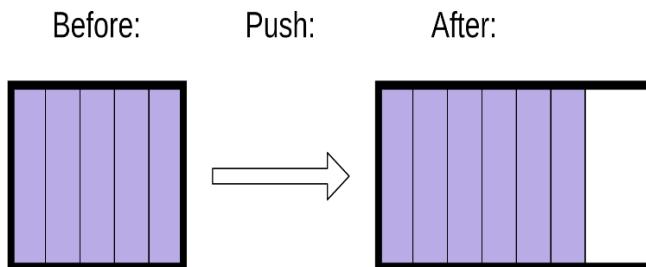
```

4. Dequelist

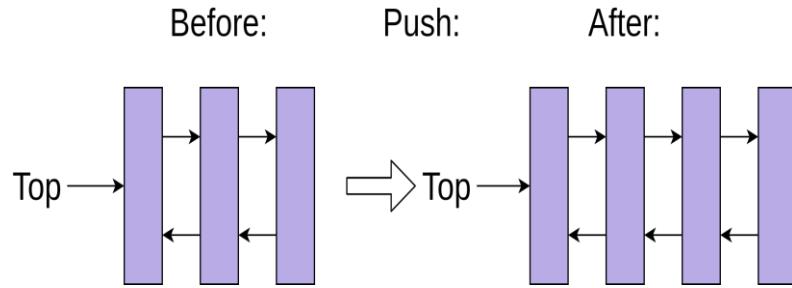
Seperti yang lihat dalam diskusi tentang di atas, itu dibangun di atas blok memori bersebelahan, yang berarti bahwa item dalam list disimpan tepat di sebelah satu sama lain:list.



Ini bekerja sangat baik untuk beberapa operasi, seperti mengindeks ke dalam mendapatkan cepat, karena Python tahu persis dimana mencari dalam memori untuk menemukannya. Tata letak memori ini juga memungkinkan irisan bekerja dengan baik pada list.list myList. Tata letak memori bersebelahan adalah alasan yang mungkin perlu mengambil lebih banyak waktu untuk beberapa objek daripada yang lain. Jika blok memori bersebelahan penuh, maka perlu mendapatkan blok lain, yang bisa memakan waktu lebih lama dari biasanya:list.append().append(). deque, di sisi lain, dibangun di atas list dua kali lipat terkait. Dalam struktur list tertaut, setiap entri disimpan dalam blok memorinya sendiri dan memiliki referensi ke entri berikutnya dalam list.



List yang ditautkan dua kali lipat sama, kecuali bahwa setiap entri memiliki referensi ke entri sebelumnya dan berikutnya dalam list. Hal ini memungkinkan Anda untuk dengan mudah menambahkan node ke kedua ujung list. Menambahkan entri baru ke dalam struktur list tertaut hanya memerlukan pengaturan referensi entri baru untuk menunjuk ke bagian atas tumpukan saat ini dan kemudian mengarahkan bagian atas tumpukan ke entri baru:



Penambahan waktu konstan dan penghapusan entri ke tumpukan ini dilengkapi dengan trade-off. Mendapatkan lebih lambat daripada untuk list, karena Python perlu berjalan melalui setiap node dari list untuk sampai ke elemen ketiga.`myDeque`. Melakukan pengindeksan acak atau mengiris tumpukan. Sebagian besar operasi pada tumpukan adalah salah satu atau `.pushpop`. Waktu dan operasi yang konstan membuat pilihan yang sangat baik untuk menerapkan tumpukan Python jika kode tidak menggunakan threading..`..append()..pop()deque`.

5. Implementasi Stack dalam Antrian

Modul antrian juga memiliki Antrian LIFO, yang pada dasarnya adalah Stack. Data dimasukkan ke dalam Antrian menggunakan fungsi `put()` dan `get()` mengambil data dari Antrian. Ada berbagai fungsi yang tersedia dalam modul ini:

- a. `maxsize` – Jumlah item yang diperbolehkan dalam antrian.
- b. `kosong()` – Kembalikan `True` jika antrian kosong, `False` sebaliknya.
- c. `full()` – Return `True` jika ada item *maksimal* dalam antrian. Jika antrian disialialisasi dengan `maxsize = 0` (default), maka `penuh()` tidak pernah mengembalikan `True`.
- d. `get()` – Hapus dan kembalikan item dari antrian. Jika antrian kosong, tunggu sampai item tersedia.
- e. `get_nowait()` – Kembalikan item jika ada yang segera tersedia, yang lain meningkatkan `QueueEmpty`.
- f. `put(item)` – Masukkan item ke dalam antrian. Jika antrian penuh, tunggu sampai slot gratis tersedia sebelum menambahkan item.

- g. `put_nowait(item)` – Masukkan item ke dalam antrian tanpa pemblokiran.
- h. `qsize()` – Kembalikan jumlah item dalam antrian. Jika tidak ada slot gratis yang segera tersedia, naikkan `QueueFull`.

```

1 # Python program to
2 # demonstrate stack implementation
3 # using queue module
4
5 from queue import LifoQueue
6
7 # Initializing a stack
8 stack = LifoQueue(maxsize=3)
9
10 # qsize() show the number of elements
11 # in the stack
12 print(stack.qsize())
13
14 # put() function to push
15 # element in the stack
16 stack.put('a')
17 stack.put('b')
18 stack.put('c')
19
20 print("Full: ", stack.full())
21 print("Size: ", stack.qsize())
22
23 # get() function to pop
24 # element from stack in
25 # LIFO order
26 print('\nElements popped from the stack')
27 print(stack.get())
28 print(stack.get())
29 print(stack.get())
30
31 print("\nEmpty: ", stack.empty())
32

```

```

Python
0
('Full: ', True)
('Size: ', 3)

Elements popped from the stack
c
b
a
('Empty: ', True)

In [18]: |

```

6. Implementasi Menggunakan Singly Linked List

- Singly Linked List memiliki dua metode addHead (item) dan removeHead() yang berjalan dalam waktu konstan. Kedua metode ini cocok untuk mengimplementasikan tumpukan.
- a. getSize() – Dapatkan jumlah item di tumpukan.
 - b. isEmpty() – Return True jika tumpukan kosong, Salah sebaliknya.
 - c. peek() – Kembalikan item teratas di tumpukan. Jika tumpukan kosong, ajukan pengecualian.
 - d. push (nilai) – Dorong nilai ke kepala tumpukan.
 - e. pop() – Hapus dan kembalikan nilai di kepala tumpukan. Jika tumpukan kosong, ajukan pengecualian.

7. Tumpukan Python dan Threading

Tumpukan Python dapat berguna dalam program multi-threaded juga, tetapi jika tidak tertarik untuk threading, maka dapat dengan aman melewati bagian ini dan melompat ke ringkasan. Jadi, sementara itu mungkin untuk membangun tumpukan Python thread-safe menggunakan, hal itu mengekspos kepada seseorang menyalah gunakannya di masa depan dan menyebabkan kondisi ras.deque. Jika threading, dapat menggunakan untuk tumpukan dan mungkin tidak ingin menggunakan untuk tumpukan, jadi bagaimana bisa membangun tumpukan Python untuk program berulir.

Ingin bagaimana belajar bahwa tumpukan beroperasi pada prinsip Last-In/First-Out. Itulah yang "LIFO" bagian dari singkatan.queueLifoQueue. Sedangkan antarmuka untuk dan serupa, menggunakan dan untuk menambah dan menghapus data dari tumpukan:listdequeLifoQueue.put().get()

Tidak seperti, dirancang untuk sepenuhnya thread-aman. Semua metodenya aman digunakan di lingkungan berulir. Ini juga menambahkan time-out opsional ke operasinya yang sering dapat menjadi fitur yang harus dimiliki dalam program berulir.dequeueLifoQueue. Namun, keamanan thread lengkap ini datang dengan biaya. Untuk mencapai keamanan thread ini, harus melakukan sedikit kerja ekstra pada setiap operasi, yang berarti bahwa itu akan memakan waktu sedikit lebih lama.LifoQueue

Seringkali, sedikit melambat ini tidak masalah dengan kecepatan program secara keseluruhan, tetapi jika telah mengukur kinerja dan menemukan bahwa operasi tumpukan adalah hambatan, maka dengan hati-hati beralih ke mungkin layak dilakukan deque. Menekankan lagi bahwa beralih dari ke karena lebih cepat tanpa pengukuran yang menunjukkan bahwa operasi tumpukan adalah hambatan adalah contoh optimasi prematur.

8. Implementasi Python Stacks

Ada beberapa opsi saat menerapkan tumpukan Python. Fokus menggunakan struktur data yang merupakan bagian dari perpustakaan Python, daripada menulis sendiri atau menggunakan paket pihak ketiga dan akan terlihat implementasi tumpukan Python berikut:

- a. list
- b. collections.deque
- c. queue.LifoQueue

Struktur built-in yang mungkin sering gunakan dalam program dapat digunakan sebagai tumpukan, dapat menggunakan .append() untuk menambahkan elemen baru ke bagian atas tumpukan, sambil menghapus elemen dalam urutan LIFO. Perintah akhir bahwa a akan menaikkan jika memanggil tumpukan kosong.listIndexError.pop() list. Masalah terbesar adalah bahwa hal itu dapat mengalami masalah kecepatan saat tumbuh. Item dalam a disimpan dengan tujuan menyediakan akses cepat ke elemen acak di. Pada tingkat tinggi, ini berarti bahwa item disimpan di samping satu sama lain dalam memori.listlistlist Jika tumpukan Anda tumbuh lebih besar dari blok memori yang saat ini memegangnya, maka Python perlu melakukan beberapa alokasi memori. Hal ini dapat menyebabkan beberapa panggilan memakan waktu lebih lama daripada yang lain..append()

Ada juga masalah yang kurang serius. Jika menggunakan untuk menambahkan elemen ke tumpukan pada posisi selain akhir, itu bisa memakan waktu lebih lama. Ini biasanya bukan sesuatu yang akan lakukan untuk tumpukan, namun..insert() Struktur data berikutnya akan membantu mengatasi masalah realokasi yang dilihat.list

Pada Implementasnya stacks jika menggunakan threading, maka harus menggunakan kecuali telah mengukur kinerja dan menemukan bahwa dorongan kecil dalam kecepatan untuk mendorong dan bermunculan akan membuat perbedaan yang cukup untuk menjamin risiko pemeliharaan. dequeLifoQueue list Mungkin akrab, tetapi harus dihindari karena berpotensi memiliki masalah realokasi memori. Antarmuka untuk dan identik, dan tidak memiliki masalah ini, yang membuat pilihan terbaik untuk tumpukan Python non-threaded dequeListdequedeque

Mari pahami, cara menggunakan PUSH di Stack. Lihat program yang di bawah ini

```
1 class Stack:
2     def __init__(self):
3         self.stack = []
4
5     def add(self, dataval):
6 # Use list append method to add element
7         if dataval not in self.stack:
8             self.stack.append(dataval)
9             return True
10        else:
11            return False
12
13 # Use list pop method to remove element
14     def remove(self):
15         if len(self.stack) <= 0:
16             return ("No element in the Stack")
17         else:
18             return self.stack.pop()
19
20 AStack = Stack()
21 AStack.add("senin")
22 AStack.add("selasa")
23 AStack.add("rabu")
24 AStack.add("kamis")
25 print(AStack.remove())
26 print(AStack.remove())
27
```

Ketika kode di atas dieksekusi, ia menghasilkan hasil berikut :

kamis

rabu

Seperti yang diketahui kumpulan data dapat dihapus hanya elemen data paling atas dari tumpukan, kita menerapkan program python yang melakukan itu. Fungsi hapus dalam program berikut mengembalikan elemen paling atas. kami memeriksa elemen atas dengan menghitung ukuran tumpukan terlebih dahulu dan kemudian menggunakan metode pop() bawaan untuk mengetahui elemen paling1 atas.

```

1 class Stack:
2     def __init__(self):
3         self.stack = []
4
5     def add(self, dataval):
6         # Use list append method to add element
7         if dataval not in self.stack:
8             self.stack.append(dataval)
9             return True
10        else:
11            return False
12
13 # Use list pop method to remove element
14 def remove(self):
15     if len(self.stack) <= 0:
16         return ("Nomer Eleman Tumpukan")
17     else:
18         return self.stack.pop()
19
20 AStack = Stack()
21 AStack.add("satu")
22 AStack.add("dua")
23 AStack.add("tiga")
24 AStack.add("empat")
25 print(AStack.remove())
26 print(AStack.remove())
27

```

Ketika kode di atas dieksekusi, ia menghasilkan hasil berikut :

empat

tiga

Kesimpulan materi tumpukan dan telah dilihat dari situasi di mana dapat digunakan dalam program kehidupan nyata. Dimana telah mengevaluasi tiga opsi berbeda untuk menerapkan tumpukan dan melihat itu adalah pilihan tepat untuk program non-threaded. Jika diterapkan tumpukan di lingkungan

threading, maka kemungkinan ide yang baik untuk menggunakan .dequeLifoQueue

Dalam kamus bahasa Inggris, kata stack berarti mengatur objek di atas yang lain. Ini adalah cara yang sama memori dialokasikan dalam struktur data ini. Ini menyimpan elemen data dengan cara yang sama seperti sekelompok piring disimpan satu di atas yang lain di dapur. Jadi tumpukan data strcuture memungkinkan operasi di salah satu ujung which dapat disebut bagian atas tumpukan. Kita dapat menambahkan elemen atau menghapus elemen hanya membentuk ini en dof tumpukan.

Dalam tumpukan elemen yang diinsret terakhir secara berurutan akan keluar lebih dulu karena hanya dapat menghapus dari bagian atas tumpukan. Fitur tersebut dikenal sebagai fitur Last in First Out (LIFO). Operasi menambahkan dan menghapus elemen dikenal sebagai PUSH dan POP. Dalam program berikut kami menerapkannya sebagai fungsi tambahkan dan dan hapus. Kami mendeklarasikan list kosong dan menggunakan metode append() dan pop() untuk menambahkan dan menghapus elemen data

C. LATIHAN SOAL

1. Jelaskan yang anda ketahui mengenai stack!
2. Jelaskan konsep LIFO dalam stack!
3. Jelaskan Konsep Record,Array,Top, Pada Stack !
4. Jelaskan 2 Perbedaan Top Dengan Max Stack !
5. Buatlah Contoh Stack dengan baha apemrograman 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). Pendekripsi Kantuk Secara Real Time Menggunakan Pustaka Opencv Dan Dlib Python. *Sainstech : Jurnal Penelitian Dan Pengkajian*

PERTEMUAN 13

QUEUE

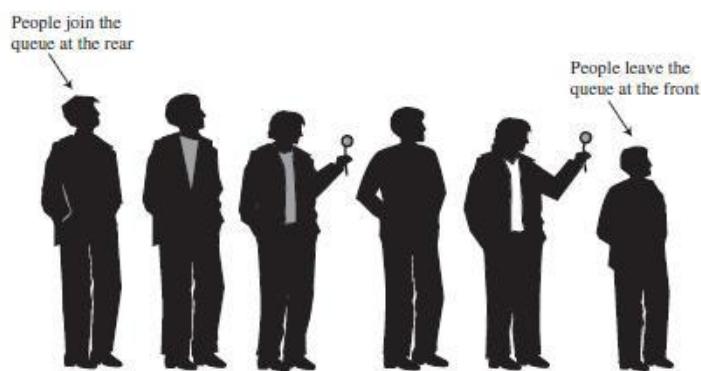
A. TUJUAN PEMBELAJARAN

Setelah mempelajari materi pertemuan diharapkan mahasiswa mengerti dan paham queue, proses FIFO, membuat dan menghapus item data antrian dan mampu membuat program queue dalam Python

B. URAIAN MATERI

1. Queue

Kata antrian dalam bahasa Inggris untuk baris untuk "mengantri" berarti mengantre. Dalam ilmu komputer, antrian adalah struktur data yang agak berbeda seperti tumpukan, kecuali bahwa dalam antrian, item pertama yang dimasukkan adalah yang pertama jadi dihapus (First-In-First-Out, FIFO), sementara dalam tumpukan, seperti yang telah terlihat, item terakhir dimasukkan adalah yang pertama akan dihapus (LIFO). Antrian berfungsi seperti baris di film: Orang pertama yang bergabung di belakang garis adalah orang pertama yang mencapai garis depan antrian dan beli tiket. Orang terakhir yang mengantre adalah orang terakhir yang membeli tiket (atau — jika pertunjukan terjual habis — gagal membeli tiket).



Gambar 13. 1 Antrian Orang

Antrian digunakan sebagai alat pemrogram sebagai tumpukan. Contoh di mana sebuah antrian membantu mencari grafik, digunakan untuk membuat model situasi dunia nyata seperti orang yang mengantri di bank, pesawat menunggu lepas landas, atau data paket menunggu untuk dikirim melalui Internet.

Ada berbagai antrian yang melakukan tugasnya di komputer (atau jaringan). Ada antrian printer tempat pekerjaan cetak menunggu printer akan tersedia. Antrian juga menyimpan data keystroke saat mengetik di papan ketik. Dengan cara ini, jika menggunakan pengolah kata tetapi komputer digunakan sebentar melakukan hal lain saat menekan tombol, tombol tidak akan hilang; itu menunggu di antri hingga pengolah kata memiliki waktu untuk membacanya. Menggunakan antrian menjamin penekanan tombol tetap berurutan sampai dapat diproses.

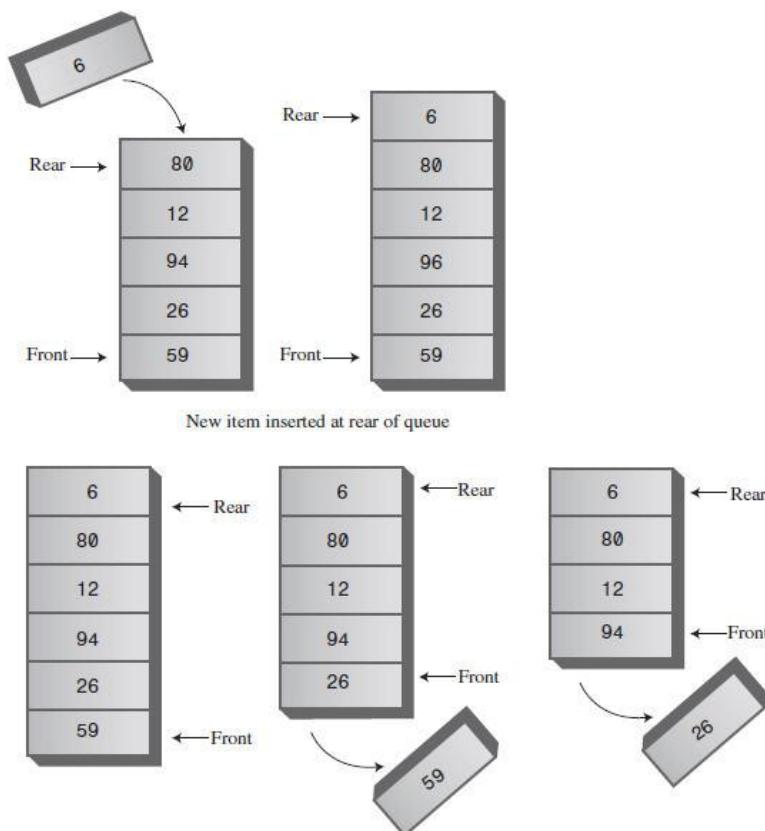
Applet ini mendemonstrasikan antrian berdasarkan array. Ini adalah pendekatan umum, meskipun daftar teraut juga biasa digunakan untuk mengimplementasikan antrian. Dua operasi antrian dasar adalah memasukkan item, yang ditempatkan di belakang antrian, dan menghapus item, yang diambil dari depan antrian. Ini adalah mirip dengan seseorang yang bergabung dengan barisan belakang penonton film dan, setelah tiba ditaris depan dan membeli tiket, melepaskan dirinya dari depan garis.

Istilah penyisipan dan penghapusan dalam tumpukan cukup standar; semua orang bilang dorong dan muncul. Standardisasi belum berkembang sejauh ini dengan antrian. Sisipkan juga disebut put atau add atau enqueue, sedangkan remove bisa disebut delete atau get atau dequeue. Bagian belakang queue, tempat item dimasukkan, juga disebut back atau tail atau end. Bagian depan, dimana barang dilepas, bisa juga disebut kepala. Kami akan menggunakan istilah sisipan, lepas, depan, dan belakang.

2. Insert Button Antrian

Dengan menekan berulang kali tombol Ins di applet antrian, dapat menyisipkan item baru. Setelah penekanan pertama, akan diminta memasukkan nilai kunci untuk item baru ke dalam bidang teks Angka; ini harus berupa angka dari 0 hingga 999. Selanjutnya penekanan akan memasukkan item dengan kunci ini di belakang antrian dan menaikkan panah belakang sehingga menunjuk ke item baru.

Tombol Hapus Demikian pula, dapat menghapus item di depan antrian menggunakan tombol Rem. Item dihapus, nilai item disimpan di bidang Angka (sesuai dengan metode remove () mengembalikan nilai), dan panah depan bertambah. Dalam applet, sel yang menahan item yang dihapus berwarna abu-abu untuk menunjukkan bahwa item tersebut hilang. Secara normal implementasi, itu akan tetap dalam memori tetapi tidak akan dapat diakses karena Front telah melewatinya. Operasi masukkan dan hapus ditunjukkan pada Gambar 12.2.

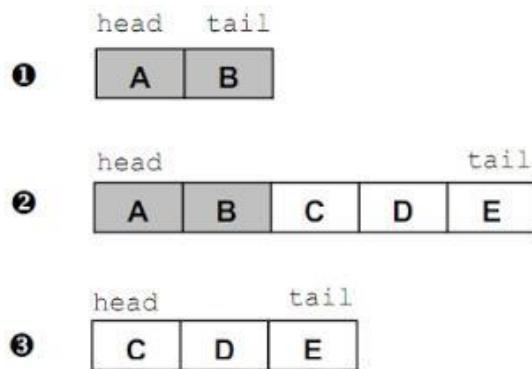


Gambar 13. 2 Operasi Metode Kelas Antrian

Dalam struktur data, antrian sedikit berbeda dengan stack. Jika dalam stack, data yang pertama masuk adalah yang paling akhir keluar, namun dalam queue data yang paling awal masuk adalah yang paling awal keluar. Konsep dari Queue adalah LIFO (Last In First Out).

Contoh dalam kehidupan, isalnya antrian dibioskop. pada saat mengantri, orang yang pertama kali mengantri akan mendapat pelayanan pertama dan yang pertama yang akan keluar dari antrian

Penggambaran dari Queue dalam struktur data adalah sebagai berikut :



Gambar 13. 3 Ilustrasi Queue dalam Struktur Data

Pada saat menempatkan elemen pada ujung (tail) dari queue disebut dengan *enqueue*. Pada saat memindahkan elemen dari kepala (head) ke dalam queue disebut dengan *dequeue*.. Karakteristik penting dari queue adalah:

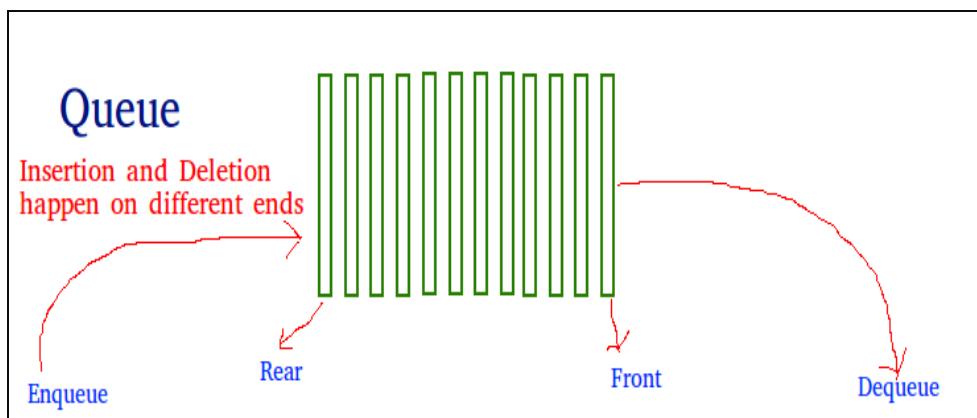
- Element antrian yaitu item-item data yang terdapat di elemen antrian
- Front (element terdepan di antrian)
- Rear (element terakhir di antrian)
- Jumlah elemen pada antrian (Count)
- Status antrian

Kondisi antrian yang menjadi perhatian adalah :

- Penuh : Bila elemen pada antrian mencapai kapasitas maksimum antrian.
Pada kondisi ini, tidak mungkin dilakukan penambahan ke antrian.
Penambahan elemen menyebabkan kondisi kesalahan Overflow.
- Kosong : Bila elemen pada antrian mencapai kapasitas maksimum antrian.
Pada kondisi ini, tidak mungkin dilakukan penambahan ke antrian.
Penambahan elemen menyebabkan kondisi kesalahan Overflow.

3. Menambahkan Elemen

Dalam contoh ini membuat kelas antrian di mana kita menerapkan metode First-in-First-Out. Penggunaan metode sisipkan bawaan untuk menambahkan elemen data. seperti tumpukan, antrian adalah struktur data linier yang menyimpan item dengan cara First In First Out (FIFO). Dengan antrian, item yang paling sedikit ditambahkan akan dihapus terlebih dahulu. Contoh antrian yang baik adalah antrian konsumen di mana konsumen yang pertama dilayani terlebih dahulu.



Gambar 13. 4 FIFO dalam Antrian

Operasi yang terkait dengan antrian adalah:

- a. Enqueue: Menambahkan item ke antrian. Jika antrian penuh, maka dikatakan sebagai kondisi Overflow – Kompleksitas Waktu: O (1)
- b. Dequeue: Menghapus item dari antrian. Item yang muncul dalam urutan yang sama di mana mereka didorong. Jika antrian kosong, maka dikatakan sebagai kondisi Underflow – Kompleksitas Waktu: O (1)
- c. Depan: Dapatkan item depan dari antrian – Kompleksitas Waktu: O(1)
- d. Belakang: Dapatkan item terakhir dari antrian – Kompleksitas Waktu: O(1)

4. Implementasi Menggunakan Antrian.

Queue adalah modul built-in Python yang digunakan untuk mengimplementasikan antrian. Antrian (maxsize) menginisialisasi variabel ke ukuran maksimal maxsize. Maxsize dari nol '0' berarti antrian yang tak terbatas. Antrian ini mengikuti aturan FIFO. Ada berbagai fungsi yang tersedia dalam modul ini:

- a. maxsize – Jumlah item yang diizinkan dalam antrian.
- b. kosong() – Kembalikan True jika antrian kosong, Salah sebaliknya.
- c. full() – Return True jika ada item maksimal dalam antrian. Jika antrian disialalisasi dengan maxsize = 0 (default), maka penuh () tidak pernah mengembalikan True.
- d. get() – Hapus dan kembalikan item dari antrian. Jika antrian kosong, tunggu sampai item tersedia.
- e. get_nowait() – Kembalikan item jika ada yang segera tersedia, yang lain meningkatkan QueueEmpty.
- f. put (item) – Masukkan item ke dalam antrian. Jika antrian penuh, tunggu sampai slot gratis tersedia sebelum menambahkan item.
- g. put_nowait (item) – Masukkan item ke dalam antrian tanpa pemblokiran. Jika tidak ada slot gratis yang segera tersedia, naikkan QueueFull.
- h. qsize() – Kembalikan jumlah item dalam antrian.

```

1 # demonstrate implementation of
2 # queue using queue module
3
4 from queue import Queue
5 # Initializing a queue
6 q = Queue(maxsize = 3)
7
8 # qsize() give the maxsize
9 # of the Queue
10 print(q.qsize())
11
12 # Adding of element to queue
13 q.put('a')
14 q.put('b')
15 q.put('c')
16
17 # Return Boolean for Full
18 # Queue
19 print("\nFull: ", q.full())
20
21 # Removing element from queue
22 print("\nElements dequeued from the queue")
23 print(q.get())
24 print(q.get())
25 print(q.get())
26
27 # Return Boolean for Empty
28 # Queue
29 print("\nEmpty: ", q.empty())
30
31 q.put(1)
32 print("\nEmpty: ", q.empty())
33 print("Full: ", q.full())
34
35 # This would result into Infinite
36 # Loop as the Queue is empty.
37 # print(q.get())
38

```

Hasil:

0

Full: True

Elements dequeued from the queue

a

b

c

Empty: True

Empty: False

Full: False

5. Cara Menggunakan Antrian di Python

Untuk mulai membangun antrian Python, diperlukan mengimpor modul Python terlebih dahulu:queue

Import queue

Semua versi yang lebih baru memiliki modul ini tersedia untuk digunakan. Hal ini memungkinkan untuk menerapkan antrian multithreadingPython:

- a. Untuk menambahkan elemen ke antrian, gunakan . Ini disebut operasi enqueue.put()
- b. Untuk menghapus elemen dari antrian, gunakan . Ini disebut operasi dequeue.get()
- c. Prinsip FIFO (First In, First Out) berarti elemen pertama yang dimasukkan juga akan menjadi yang pertama dihapus.

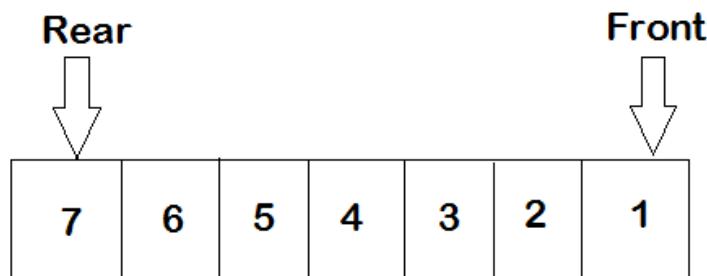
Dengan menggunakan argumen, dapat memilih untuk membatasi jumlah elemen dalam antrian. Untuk membiarkan ukurannya tidak terbatas, definisikan sebagai atau angka negatif.maxsize0

- a. Untuk menyederhanakan bekerja dengan antrian prioritas, ikuti pola dan gunakan untuk menentukan prioritas.number, elementnumber
- b. Untuk membuat antrian multiprocessing Python (sebagai lawan multithreading), gunakan fungsi untuk modul.multiprocessing.Queue()multiprocessin

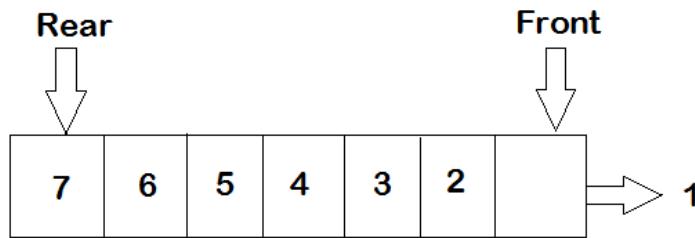
Antrian adalah wadah yang menyimpan data. Data yang dimasukkan pertama akan dihapus terlebih dahulu, dan karenanya antrian juga disebut "First in First Out" (FIFO). Antrian memiliki dua ujung depan dan belakang. Item dimasukkan dari belakang dan dikeluarkan dari sisi depan.

6. Cara Kerja Queue

Antrian dapat dengan mudah dibandingkan dengan contoh dunia nyata garis orang yang menunggu dalam antrian di loket tiket, orang yang berdiri lebih dulu akan mendapatkan tiket terlebih dahulu, diikuti oleh orang berikutnya dan sebagainya. Logika yang sama berlaku untuk struktur data antrian juga. Berikut adalah representasi diagram dari antrian:



- a. Bagian Belakang mewakili titik di mana item dimasukkan ke dalam antrian. Dalam contoh ini, 7 adalah nilai untuk itu.
- b. Front mewakili titik di mana item dari antrian akan dihapus. Jika menghapus item dari antrian, elemen pertama yang akan dapatkan adalah 1, seperti yang ditunjukkan pada gambar.
- c. Item 1 adalah yang pertama dimasukkan dalam antrian, dan saat menghapusnya adalah yang pertama keluar. Oleh karena itu antrian disebut FIRST IN FIRST OUT (FIFO)



Dalam antrian, item dihapus secara berurutan dan tidak dapat dihapus dari antaranya, tidak dapat menghapus item 5 secara acak dari antrian, untuk melakukan itu dimana harus menghapus semua item sebelum 5. Item dalam antrian akan dihapus dalam urutan yang dimasukkan. Jenis Antrian di Python ada dua jenis antrian di Python:

- Pertama di First out Queue: Untuk ini, elemen yang keluar pertama akan menjadi yang pertama keluar.Untuk bekerja dengan FIFO, harus menghubungi kelas Antrian() dari modul antrian.
- Terakhir di First out Queue: Di sini, elemen yang dimasukkan terakhir akan menjadi yang pertama keluar.Untuk bekerja dengan LIFO, dimana harus menghubungi kelas LifoQueue() dari modul antrian.

7. Menambahkan Item Antrian

Mari kerjakan contoh untuk menambahkan item dalam antrian. Untuk mulai bekerja dengan antrian, pertama-tama impor antrian modul, seperti yang ditunjukkan pada contoh di bawah ini. Untuk menambahkan item, dapat menggunakan metode put() seperti yang ditunjukkan pada contoh:

```
import queue
q1 = queue.Queue()
q1.put(10) #this will additem 10 to the queue.
```

Secara default, ukuran antrian tidak terbatas dan dapat menambahkan sejumlah item ke dalamnya. Jika ingin menentukan ukuran antrian yang sama dapat dilakukan sebagai berikut:

```
import queue
q1 = queue.Queue(5) #The max size is 5.
```

```
q1.put(1)  
q1.put(2)  
q1.put(3)  
q1.put(4)  
q1.put(5)  
print(q1.full()) # will return true.
```

Hasil:

True

Sekarang ukuran antrian adalah 5, dan tidak akan mengambil lebih dari 5 item, dan metode q1.full() akan kembali benar. Menambahkan item lagi tidak akan mengeksekusi kode lebih jauh.

8. Menghapus Item Antrian

Untuk menghapus item dari antrian, dimana dapat menggunakan metode yang disebut get(). Metode ini memungkinkan item dari antrian saat dipanggil. Contoh berikut menunjukkan cara menghapus item dari antrian.

```
import queue  
q1 = queue.Queue()  
q1.put(10)  
item1 = q1.get()  
print('The item removed from the queue is ', item1)
```

Hasil:

The item removed from the queue is 10

Ringkasan dalam proses diatas adalah:

- a. Antrian adalah wadah yang menyimpan data. Ada dua jenis Queue, FIFO, dan LIFO.
- b. Untuk FIFO (First in First out Queue), elemen yang masuk pertama akan menjadi yang pertama keluar.

- c. Untuk LIFO (Last in First out Queue), elemen yang dimasukkan terakhir akan menjadi yang pertama keluar.
- d. Item dalam antrian ditambahkan menggunakan metode put (item).
- e. Untuk menghapus item, metode get() digunakan

9. Penerapan Antrian di Python

Sangat mudah untuk bekerja dengan antrian di python. Berikut adalah langkah langkah yang harus diikuti untuk memanfaatkan antrian dalam kode :

- a. Langkah 1) Anda hanya perlu mengimpor modul antrian, seperti yang ditunjukkan di bawah ini:

```
import queue
```

Modul ini tersedia secara default dengan python, dan Anda tidak memerlukan instalasi tambahan untuk mulai bekerja dengan antrian. Ada 2 jenis antrian FIFO (pertama di first out) dan LIFO (terakhir di first out).

- b. Langkah 2) Untuk bekerja dengan antrian FIFO, hubungi kelas Antrian menggunakan modul antrian yang diimpor seperti yang ditunjukkan di bawah ini:

```
import queue
```

```
q1 = queue.Queue()
```

- c. Langkah 3) Untuk bekerja dengan antrian LIFO, hubungi kelas LifoQueue() seperti yang ditunjukkan di bawah ini:

```
import queue
```

```
q1 = queue.LifoQueue()
```

Metode yang tersedia di dalam kelas Queue dan LifoQueue

Berikut ini adalah metode penting yang tersedia di dalam kelas Queue dan LifoQueue:

- a. put(item): Ini akan menempatkan item di dalam antrian.
- b. dapatkan(): Ini akan mengembalikan item dari antrian.
- c. kosong(): Ini akan kembali benar jika antrian kosong dan salah jika item hadir.

- d. qsize(): mengembalikan ukuran antrian.
- e. penuh(): kembali benar jika antrian penuh, jika tidak salah.

Dalam kasus pertama di pertama keluar, elemen yang pergi pertama akan menjadi yang pertama untuk keluar. Ada berbagai cara untuk menerapkan antrian di Python. Artikel ini mencakup penerapan antrian menggunakan struktur data dan modul dari perpustakaan Python. Antrian di Python dapat diimplementasikan dengan cara berikut:

- a. daftar
- b. collections.deque
- c. queue

10. Implementasi Menggunakan List

List adalah struktur data built-in Python yang dapat digunakan sebagai antrian. Alih-alih enqueue() dan dequeue(), fungsi append() dan pop() digunakan. Namun, daftar cukup lambat untuk tujuan ini karena memasukkan atau menghapus elemen di awal membutuhkan pergeseran semua elemen lain dengan satu, membutuhkan waktu $O(n)$.

```
1 # Python program to
2 # demonstrate queue implementation
3 # using list
4 # Initializing a queue
5 queue = []
6 # Adding elements to the queue
7 queue.append('a')
8 queue.append('b')
9 queue.append('c')
10
11 print("Initial queue")
12 print(queue)
13
14 # Removing elements from the queue
15 print("\nElements dequeued from queue")
16 print(queue.pop(0))
17 print(queue.pop(0))
18 print(queue.pop(0))
19
20 print("\nQueue after removing elements")
21 print(queue)
22
```

Hasil:

Initial queue

['a', 'b', 'c']

Elements dequeued from queue

a

b

c

Queue after removing elements

[]

11. Implementasi Menggunakan Collections.Deque

Antrian di Python dapat diimplementasikan menggunakan kelas deque dari modul koleksi. Deque lebih disukai daripada daftar dalam kasus-kasus di mana kita membutuhkan tambahan dan operasi pop yang lebih cepat dari kedua ujung wadah, karena deque memberikan kompleksitas waktu O (1) untuk operasi tambahan dan pop dibandingkan dengan daftar yang menyediakan O (n) kompleksitas waktu. Alih-alih enqueue dan deque, fungsi append() dan popleft() digunakan.

```

1 # using collections.deque
2 from collections import deque
3
4 # Initializing a queue
5 q = deque()
6
7 # Adding elements to a queue
8 q.append('a')
9 q.append('b')
10 q.append('c')
11
12 print("Initial queue")
13 print(q)
14
15 # Removing elements from a queue
16 print("\nElements dequeued from the queue")
17 print(q.popleft())
18 print(q.popleft())
19 print(q.popleft())
20
21 print("\nQueue after removing elements")
22 print(q)
23
24 # Uncommenting q.popleft()
25 # will raise an IndexError
26 # as queue is now empty

```

Hasil:

Initial queue

```
deque(['a', 'b', 'c'])
```

Elements dequeued from the queue

a

b

c

Queue after removing elements

```
deque([])
```

Contoh Program Queue di python :

```
1 import os
2 import Queue
3
4 class myQueue:
5     def __init__(self):
6         self.items = Queue.Queue()
7
8     # Memeriksa apakah queue dalam keadaan kosong
9     def isEmpty(self):
10        return self.items.empty()
11    # Menambah data ke queue
12    def qPut(self, item):
13        self.items.put(item)
14    # Mengeluarkan data dari queue
15    def qGet(self):
16        if not self.items.empty():
17            return self.items.get()
18        else:
19            return "empty"
20    # Menghitung panjang queue
21    def size(self):
22        return self.items.qsize()
23
24    # Main menu aplikasi
25    def mainmenu(self):
26        pilih = "y"
27        while (pilih == "y"):
28            os.system("clear")
29            print("====")
30            print("| Menu aplikasi queue |")
31            print("====")
32            print("1. Put objek")
33            print("2. Get objek")
34            print("3. Cek Empty")
35            print("4. Panjang dari queue")
36            print("====")
37            pilihan=str(input("Silakan masukan pilihan anda: "))
38            if pilihan=="1":
```

```

37     pilihan=str(input(("Silakan masukan pilihan anda: ")))
38     if(pilihan=="1"):
39         os.system("clear")
40         obj = str(input("Masukan objek yang ingin anda tambahkan: "))
41         self.qPut(obj)
42         print("Object "+obj+" telah ditambahkan")
43         x = raw_input("")
44     elif(pilihan=="2"):
45         os.system("clear")
46         temp = self.qGet()
47         if temp != "empty":
48             print("Objek "+temp+" dihapus")
49         else:
50             print("Queue kosong")
51         x = raw_input("")
52     elif(pilihan=="3"):
53         os.system("clear")
54         print(self.isEmpty())
55         x = raw_input("")
56     elif(pilihan=="4"):
57         os.system("clear")
58         print("Panjang dari queue adalah: "+str(self.size()))
59         x = raw_input("")
60     else:
61         pilih="n"
62
63 if __name__ == "__main__":
64     q=myQueue()
65     q.mainmenu()
66

```

Hasil jika di run adalah :

=====

| Menu aplikasi queue |

=====

1. Put objek
2. Get objek
3. Cek Empty
4. Panjang dari queue

=====

Antrian dalam kehidupan kita sehari-hari saat kita menunggu layanan. Struktur data antrian aslo berarti sama di mana elemen data diatur dalam antrian. Keunikan antrian terletak pada cara item ditambahkan dan dihapus. Item diperbolehkan di ujung tetapi dihapus dari ujung yang lain. Jadi ini adalah metode first-in-first out. Antrian dapat diimplementasikan menggunakan daftar python di mana dapat menggunakan metode sisipkan () dan pop() untuk menambahkan dan menghapus elemen. Tidak ada penyisipan elemen data selalu ditambahkan di akhir antrian.

C. LATIHAN SOAL

1. Jelaskan yang anda ketahui mengenai antrian!
2. Jelaskan operasi push dalam sebuah antrian!
3. Jelaskan operasi POP dalam sebuah antrian!
4. Jelaskan Bagaimana pengecekan kondisi diawal antrian!
5. Buatlah contoh implementasi antrian dengan bahasa python!

D. REFERENSI

- Amanulhaq, A. A. (2021). Implementasi Algoritma Image Hashing Dan Hamming Distance Untuk Deteksi Kemiripan Gambar. *Repository Its*.
- Basant Agarwal, B. B. (2018). Hand-On Data Structures And Algorithms With Python. London: Packt Publishing.
- Emi Sita Eriana, A. Z. (2021). Penerapan Metode Personal Extreme Programming Dalam Perancangan Aplikasi Pemilihan Ketua Hmsi Dengan Weighted Product. *Jurnal Ilmu Komputer*, 27-32.
- Emi Sita Eriana, A. Z. (2021). Praktikum Algoritma Dan Pemrograman. Tangerang Selatan: Unpam Press.
- Jodi, U. R. (2020). Algoritma Dan Struktur Data.
- 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.

PERTEMUAN 14

ANTRIAN DENGAN ARRAY

A. TUJUAN PEMBELAJARAN

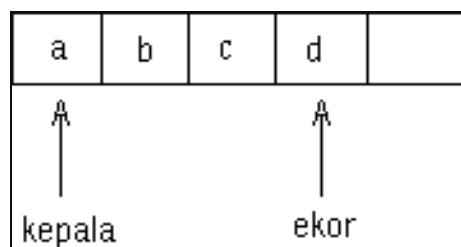
Setelah mempelajari materi pertemuan ini diharapkan mahasiswa mengerti dan paham antrian dengan array, Operasi antrian, menyisipkan dan menghapus data pada antrian dan mampu membuat program queue dalam Python

B. URAIAN MATERI

1. Antrian

Queue (antrian) adalah kumpulan data yang penambahan elemennya dilakukan pada suatu ujung (bagian belakang) dan penghapusannya dilakukan pada ujung yang lain (bagian depan). Prinsip ini biasa juga disebut dengan First In First Out (masuk pertama keluar pertama). Dengan kata lain urutan keluar suatu elemen data urutannya sama dengan urutan masuknya.

Implementasi Queue dengan Array Seperti dijelaskan sebelumnya, queue merupakan kumpulan data. Untuk mengimplementasikan kumpulan data biasanya kita menggunakan array. Array dapat digunakan untuk menyimpan data dan dengan dua variabel yang berisi nilai index kepala dan ekor queue. Dua index penunjuk ini berguna ketika akan dilakukan penambahan dan penghapusan. Pada gambar 1 dapat dilihat struktur queue dengan dua penunjuk.



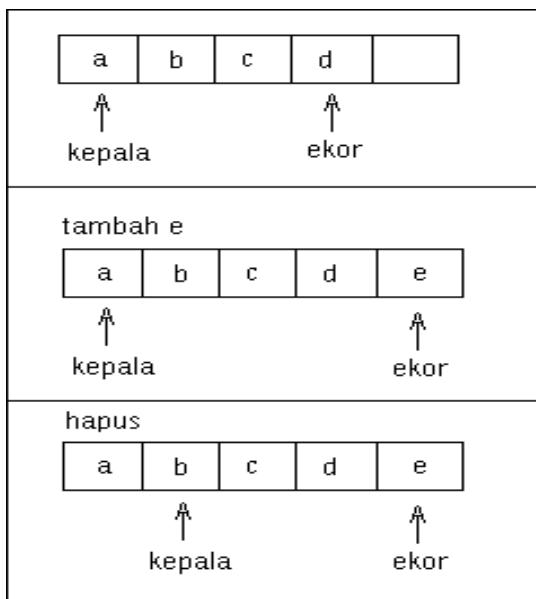
Gambar 14. 1 Ilustrasi Queue

Pada queue dikenal dua operasi dasar yaitu penambahan dan penghapusan. Penambahan akan mengubah nilai dari ekor. Pada saat data ditambahkan, nilai ekor akan menuju index berikutnya (jika tidak penuh) lalu

mengisi data pada tempat tersebut. Jika terjadi penghapusan maka data yang akan dihapus adalah data yang ditunjuk oleh index kepala. Hal ini memperjelas penggunaan FIFO dalam queue.

Jika mengimplementasikan queue dengan array penambahan data terbatas pada ukuran array yang digunakan. Hal lain yang harus diperhatikan adalah pada saat penghapusan data. Pada saat penghapusan data, harus terdapat pemeriksaan apakah queue kosong. Jika kosong maka penghapusan dibatalkan karena tidak mungkin menghapus data dari queue yang sudah kosong.

Cara pertama yang dapat dilakukan untuk mengimplementasikan queue dengan array adalah sebagai berikut. Ketika terjadi penambahan data maka index ekor akan ditambah dengan satu dan array dengan index tersebut akan diisi dengan data. Ketika terjadi penghapusan nilai index kepala akan ditambahkan dengan satu. Ilustrasi penambahan dan pengurangan dapat dilihat pada gambar 2.

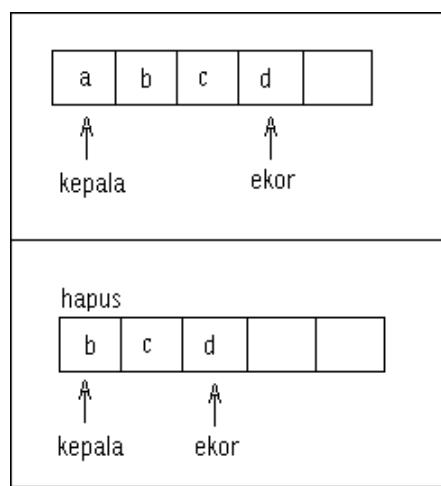


Gambar 14. 2 Ilustrasi Cara Pertama Queue

Pada Gambar 2 dapat dilihat bahwa data pada kepala queue tidak dihapus, hanya penunjuknya saja yang digeser ke index berikutnya. Kode program secara lengkap dapat dilihat pada Program 1. Cara pertama kurang efisien dalam penggunaan memori. Blok memori data yang sudah terhapus

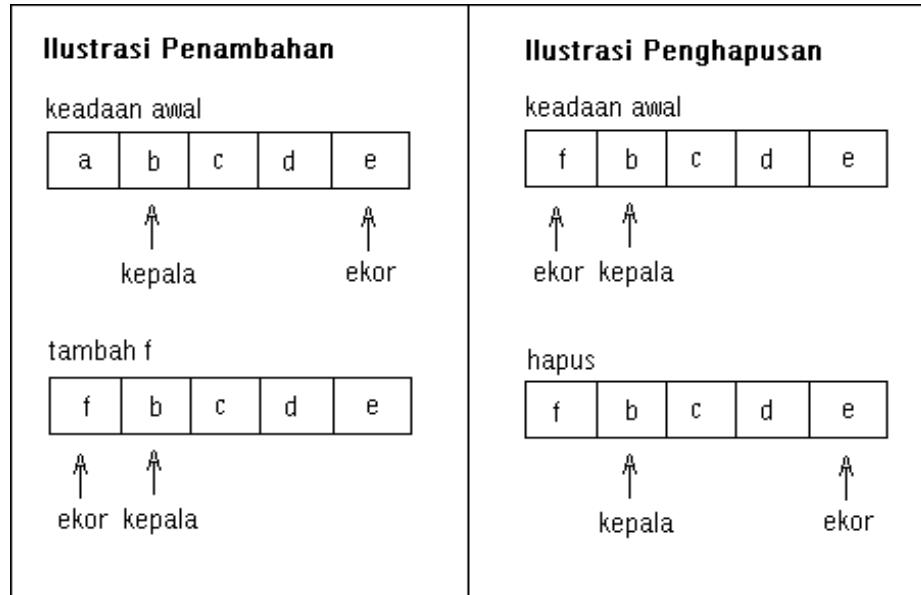
tidak dapat digunakan lagi. Sehingga blok memori yang dapat digunakan untuk menyimpan data menjadi semakin berkurang setiap terjadi penghapusan data.

Untuk memecahkan masalah tersebut dilakukan pergeseran setiap terjadi penghapusan data. Ketika data dihapus maka seluruh item array dari index kepala + 1 hingga ekor akan dipindahkan ke index sebelumnya. Dengan cara ini, nilai array dengan index kepala akan tertimpah oleh nilai sebelumnya (terhapus). Pada teknik ini tidak terjadi perubahan penunjuk kepala pada saat penghapusan data. Pada saat penambahan, yang terjadi sama dengan cara sebelumnya. Gambar 3 menunjukkan ilustrasi penghapusan cara kedua



Gambar 14. 3 Ilustrasi penghapusan cara kedua

Pada cara kedua tersebut dapat dilihat tidak ada pemborosan memori. Sehingga masalah pada cara pertama sudah terselesaikan. Cara array melingkar. Dengan cara penambahan maupun penghapusan data mirip dengan cara pertama namun dengan ketentuan berbeda pada penentuan kepala dan ekor. Pada saat penambahan di kedua cara sebelumnya jika nilai index ekor sudah mencapai nilai maximum maka disimpulkan bahwa queue telah penuh. Pada cara ini, nilai index ekor sudah mencapai nilai maximum nilai ekor akan melakukan pemeriksaan pada index pertama dari array. Jika bukan merupakan kepala maka index ekor merupakan nilai index pertama array. Pada saat penghapusan, yang terjadi adalah sebaliknya. Ketika data pada index array paling awal dihapus, akan dilakukan pemeriksaan apakah data pada nilai index ekor merupakan data terakhir atau bukan. Jika bukan maka penunjuk kepala akan menunjuk kepada index array terakhir. Gambar 4 menunjukkan ilustrasi penambahan dan penghapuan

**Gambar 14. 4 Ilustrasi Cara Ketiga Queue**

Dengan cara ini permasalahan cara pertama dan kedua dapat diselesaikan. Kode program secara lengkap dapat dilihat pada Program 3.

2. Antrian dengan Array

Antrian adalah antrian pelanggan yang membutuhkan pelayanan dari satu atau lebih fasilitas pelayanan. Definisi lain dari antrian adalah seperangkat klien, server, dan aturan yang mengatur aliran layanan antara kedatangan klien dan layanan. Pada komputer multi program, antrian adalah sekumpulan data, dimana penambahan elemen hanya dapat dilakukan di bagian belakang yang disebut bagian belakang, dan pengambilan elemen dilakukan melalui ujung depan yang disebut ujung depan.

Jumlah komponen array ditunjukkan oleh indeks yang disebut tipe indeks. Array adalah struktur data yang diproses berdasarkan indeksnya. Array terdiri dari array satu dimensi (linear array), dua dimensi (matriks) dan multi dimensi. Array dapat berupa tipe data sederhana (byte, kata, bilangan bulat, bilangan real, nilai Boolean, karakter, atau string) dan tipe data skalar.

berada dalam keadaan kosong atau tidak ada satupun elemen di dalamnya. Pada keadaan ini tidak dapat dilakukan penghapusan elemen karena akan menyebabkan kesalahan yang disebut underflow. Elemen baru dapat ditambahkan ketika antrian berada dalam keadaan tidak penuh. Jika

antrian dalam keadaan penuh maka elemen baru tidak dapat ditambahkan, karena jika dilakukan penambahan elemen baru maka akan terjadi kesalahan atau yang disebut overflow. Dengan demikian elemen tersebut harus menunggu sampai antrian berada dalam keadaan bisa menambahkan elemen baru. Dalam sebuah sistem antrian terdapat beberapa model antrian yang mungkin bisa dilakukan antara lain adalah model antrian dengan melakukan penggeseran elemen ke posisi elemen sebelumnya, model antrian front ditempati oleh elemen yang baru masuk, model antrian dimana perpindahan dilakukan oleh sembarang elemen, dan model antrian posisi elemen sembarang. Prosedur-prosedur yang dilakukan untuk menggambarkan beberapa model antrian tersebut adalah:

Prosedur awal yang dilakukan adalah menggambarkan keadaan awal antrian, yaitu dalam keadaan kosong sehingga penambahan elemen dapat dilakukan. Prosedur yang dilakukan untuk menggambarkan keadaan awal antrian yang dilakukan pertama kali sebelum memasukan elemen ke dalam antrian adalah memeriksa keadaan awal antrian, apakah berada dalam keadaan kosong atau penuh. Prosedur yang dilakukan untuk memeriksa apakah antrian dalam keadaan kosong,

Jika antrian dalam keadaan kosong, maka dapat dilakukan penambahan elemen ke dalam antrian. Prosedur yang dilakukan untuk menambahkan elemen ke dalam antrian. Pada program utama ada tiga tahap yang akan dilakukan terhadap antrian, ketiga tahap tersebut adalah memasukkan objek antrian, objek keluar antrian dan perintah keluar. Ketiga langkah tersebut terlebih dahulu dideklarasikan di awal program utama.

Pada tahap pertama yaitu untuk memasukkan objek antrian, dilakukan operasi insert. Operasi insert merupakan operasi untuk menambahkan elemen ke dalam antrian. Operasi ini hanya bisa dilakukan jika antrian tidak berada dalam keadaan penuh, jika operasi insert tetap dilakukan pada antrian yang sudah penuh maka akan terjadi error yang disebut overflow. Proses kedua yang dilakukan adalah proses menghapus elemen. Proses ini dilakukan dengan menggunakan operasi remove, syaratnya antrian tidak boleh kosong, karena ketika antrian dalam keadaan seperti stack, maka kita juga mengetahui bahwa ada dua operasi dasar dalam antrian yaitu penambahan elemen baru yang akan kita tempatkan di antrian Behind. Dan hapus elemen di depan antrian.

Selain itu, sering perlu memeriksa apakah antrian memiliki konten atau kosong. Operasi penambahan elemen baru selalu dapat diselesaikan karena tidak ada batasan jumlah elemen dalam antrian. Tetapi untuk menghapus suatu elemen, maka kita harus memeriksa apakah antriannya kosong. Tentu saja, tidak mungkin untuk menghapus elemen dari antrian yang sudah kosong.



Gambar 14. 5 Queue Linier Array

Struktur data antrian dapat diimplementasikan menggunakan array satu dimensi. Antrian yang diterapkan menggunakan array hanya menyimpan jumlah nilai data tetap. Penerapan struktur data antrian menggunakan array sangat sederhana. Cukup tentukan array satu dimensi dengan ukuran tertentu dan masukkan atau hapus nilai ke dalam array itu dengan menggunakan prinsip FIFO (First In First Out) dengan bantuan variabel 'depan' dan 'belakang'. Awalnya baik 'depan' dan 'belakang' diatur ke -1. Setiap kali ingin memasukkan nilai baru ke dalam antrian, menambah nilai 'belakang' satu dan kemudian menyisipkan pada posisi itu. Setiap kali ingin menghapus nilai dari antrian, kemudian hapus elemen yang berada di posisi 'depan' dan tingkatkan nilai 'depan' satu per satu.

3. Operasi Antrian Menggunakan Array

Struktur data antrian menggunakan array dapat diimplementasikan sebagai berikut. Sebelum menerapkan operasi yang sebenarnya, pertama ikuti langkah-langkah di bawah ini untuk membuat antrian kosong.

- a. Langkah 1 - Sertakan semua file header yang digunakan dalam program dan tentukan 'SIZE' konstan dengan nilai tertentu.
- b. Langkah 2 - Deklarasikan semua fungsi yang ditentukan pengguna yang digunakan dalam implementasi antrian.
- c. Langkah 3 - Buat array satu dimensi dengan SIZE yang ditentukan di atas(int queue[SIZE])
- d. Langkah 4 - Tentukan dua variabel integer 'depan' dan 'belakang' dan inisialisasi keduanya dengan '-1'. (int depan = -1, belakang = -1)
- e. Langkah 5 - Kemudian terapkan metode utama dengan menampilkan menu daftar operasi dan melakukan panggilan fungsi yang sesuai untuk melakukan operasi yang dipilih oleh pengguna pada antrian.

Penjelasan Operasi Antrian Menggunakan Array adalah sebagai berikut :

- a. EnQueue/ Menyisipkan Nilai ke Antrian

Dalam struktur data antrian, enQueue() adalah fungsi yang digunakan untuk memasukkan elemen baru ke dalam antrian. Dalam antrian, elemen baru selalu dimasukkan pada posisi belakang. Fungsi enQueue() mengambil satu nilai integer sebagai parameter dan menyisipkan nilai itu ke dalam antrian. Kita dapat menggunakan langkah-langkah berikut untuk menyisipkan elemen ke dalam antrian.

- 1) Langkah 1

Periksa apakah antrian penuh. (belakang == SIZE-1)

- 2) Langkah 2

Jika sudah PENUH,maka tampilkan "Queue is FULL!!! Penyisipan tidak mungkin!!!" dan mengakhiri fungsinya.

- 3) Langkah 3

Jika TIDAK PENUH,maka naikkan nilai belakang satu(belakang++) dan atur antrian [belakang] = nilai.

b. DeQueue(Menghapus nilai dari Antrian)

Dalam struktur data antrian, deQueue() adalah fungsi yang digunakan untuk menghapus elemen dari antrian. Dalam antrian, elemen selalu dihapus dari posisi depan. Fungsi deQueue() tidak mengambil nilai apa pun sebagai parameter. Kita dapat menggunakan langkah-langkah berikut untuk menghapus elemen dari antrian.

1) Langkah 1

Periksa apakah antrian kosong. (depan == belakang)

2) Langkah 2

Jika kosong,maka tampilkan "Antrian KOSONG, Penghapusan tidak mungkin, dan mengakhiri fungsinya.

3) Langkah 3

Jika tidak kosong,maka tingkatkan nilai depan dengan satu (depan ++). Kemudian tampilkan antrian [depan] sebagai elemen yang dihapus. Kemudian periksa apakah kedua depan dan belakang sama(depan == belakang),jika BENAR,kemudian atur baik depan dan belakang ke'-1'(depan = belakang = -1).

c. Display() - Menampilkan elemen Antrian

Langkah-langkah berikut untuk menampilkan elemen antrian.

1) Langkah 1

Periksa apakah antrian kosong. (depan == belakang)

2) Langkah 2

Jika KOSONG,maka tampilkan "Antrian KOSONG!!!" dan hentikan fungsinya.

3) Langkah 3

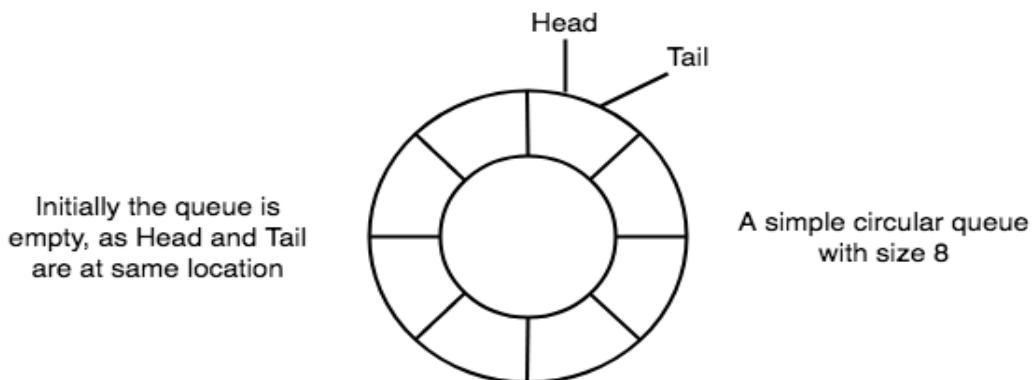
Jika TIDAK KOSONG,maka tentukan variabel integer'i'dan atur 'i = depan +1'.

4) Langkah 4

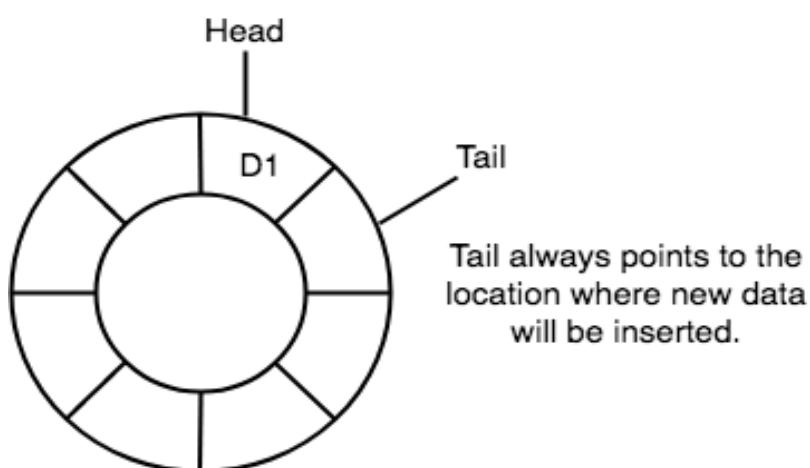
Tampilkan'antrian[i]nilai dan kenaikan 'i' nilai satu(i ++). Ulangi hal yang sama sampai nilai'i'mencapai ke belakang (i < = belakang)

Antrian Melingkar adalah struktur data antrian tetapi berbentuk melingkar, oleh karena itu setelah posisi terakhir, tempat berikutnya dalam antrian adalah posisi pertama. Disarankan untuk terlebih dahulu melalui tutorial Antrian Linier sebelum antrian Melingkar, karena kami akan memperluas implementasi yang sama. Tetapi dalam kasus antrian melingkar, karena ukuran antrian tetap, maka kami akan menetapkan daftar kami yang digunakan untuk implementasi antrian. Beberapa poin yang perlu diperhatikan di sini adalah:

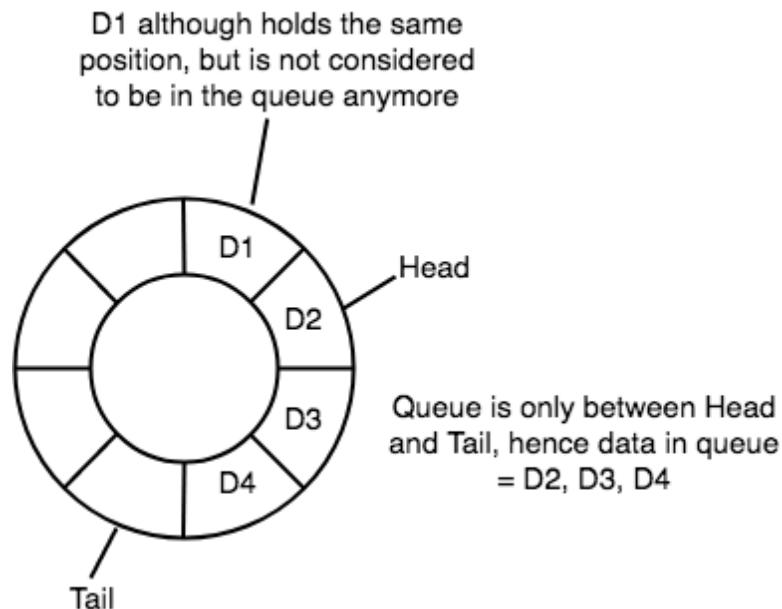
Dalam kasus antrian melingkar, pointer akan selalu menunjuk ke depan antrian, dan pointer akan selalu menunjuk ke akhir antrian. Awalnya, dan pointer akan menunjuk ke lokasi yang sama, ini berarti hwa antrian kosong. headtail



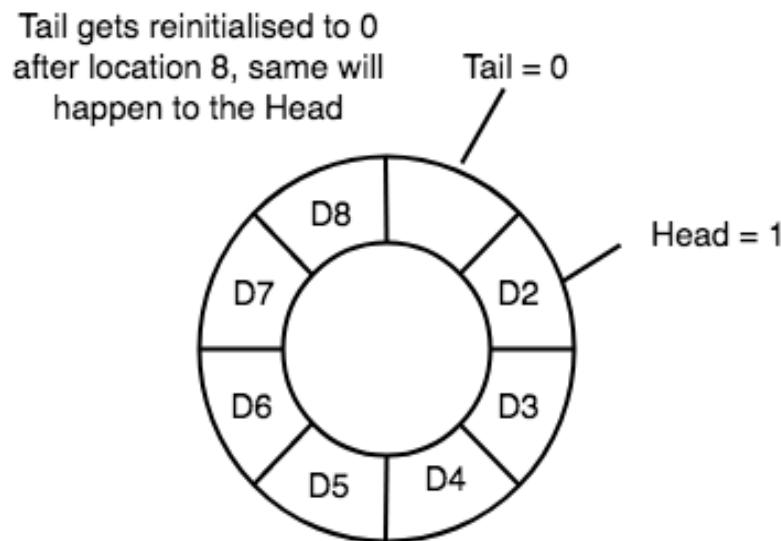
Data baru selalu ditambahkan ke lokasi yang ditunjukkan oleh penunjuk, dan setelah data ditambahkan, penunjuk akan ditambahkan untuk menunjuk ke lokasi berikutnya yang tersedia. tail



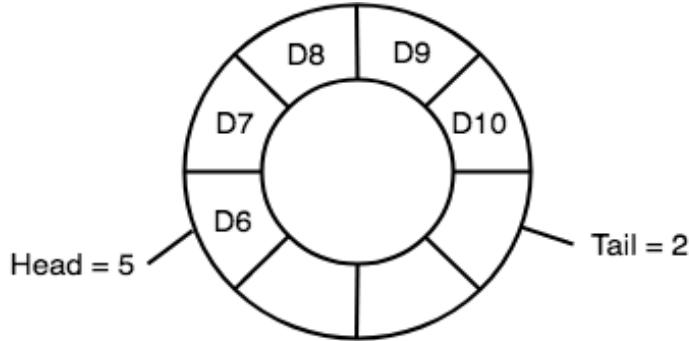
Dalam antrian melingkar, data sebenarnya tidak dihapus dari antrian. Hanya penunjuk yang ditambahkan dengan satu posisi saat dieksekusi. Karena data antrian hanya data antara dan, maka data yang tersisa di luar bukan bagian dari antrian lagi, maka dihapus.



Head dequeue head tail dan penunjuk akan mendapatkan reinitialised ke 0 setiap kali mereka mencapai akhir antrian.



headtail juga, dan pointer dapat menyilang/ berlawanan satu sama lain. Dengan kata lain, pointer bisa lebih besar dari yang terlihat. Ini akan terjadi ketika mengantre beberapa kali dan penunjuk akan dihidupkan kembali saat mencapai akhir antrian. Headtail headtail dequeuetail



In such a situation the value of the Head pointer will be greater than the Tail pointer

Poin lain yang sangat penting adalah menjaga nilai dan penunjuk dalam ukuran antrian maksimum. Pada diagram di atas antrian memiliki ukuran 8, oleh karena itu, nilai dan pointer akan selalu antara 0 dan 7. Hal ini dapat dikontrol baik dengan memeriksa setiap kali apakah atau telah mencapai dan kemudian menetapkan nilai 0 atau, kita memiliki cara yang lebih baik, yaitu untuk nilai jika membaginya dengan 8, yang tersisa tidak akan pernah lebih besar dari 8, itu akan selalu antara 0 dan 7, yang persis seperti yang kita inginkan.

```

1 # This is the CircularQueue class
2 class CircularQueue:
3     # constructor for the class
4     # taking input for the size of the Circular queue
5     # from user
6     def __init__(self, maxSize):
7         self.queue = list()
8         # user input value for maxSize
9         self.maxSize = maxSize
10        self.head = 0
11        self.tail = 0
12
13    # add element to the queue
14    def enqueue(self, data):
15        # if queue is full
16        if self.size() == (self.maxSize - 1):
17            return("Queue is full!")
18        else:
19            # add element to the queue
20            self.queue.append(data)
21            # increment the tail pointer
22            self.tail = (self.tail+1) % self.maxSize
23        return True
24
25    # remove element from the queue
26    def dequeue(self):
27        # if queue is empty
28        if self.size() == 0:
29            return("Queue is empty!")
30        else:
31            # fetch data
32            data = self.queue[self.head]

```

```
32     self.head = (self.head+1) % self.maxSize
33     return data
34 # find the size of the queue
35 def size(self):
36     if self.tail >= self.head:
37         qSize = self.tail - self.head
38     else:
39         qSize = self.maxSize - (self.head - self.tail)
40 # return the size of the queue
41     return qSize
42
43 # input 7 for the size or anything else
44 size = input("Enter the size of the Circular Queue : ")
45 q = CircularQueue(int(size))
46
47 # change the enqueue and dequeue statements as you want
48 print(q.enqueue(10))
49 print(q.enqueue(20))
50 print(q.enqueue(30))
51 print(q.enqueue(40))
52 print(q.enqueue(50))
53 print(q.enqueue('Studytonight'))
54 print(q.enqueue(70))
55 print(q.enqueue(80))
56 print(q.dequeue())
57 print(q.dequeue())
58 print(q.dequeue())
59 print(q.dequeue())
60 print(q.dequeue())
61 print(q.dequeue())
62 print(q.dequeue())
63 print(q.dequeue())
64 print(q.dequeue())
65
```

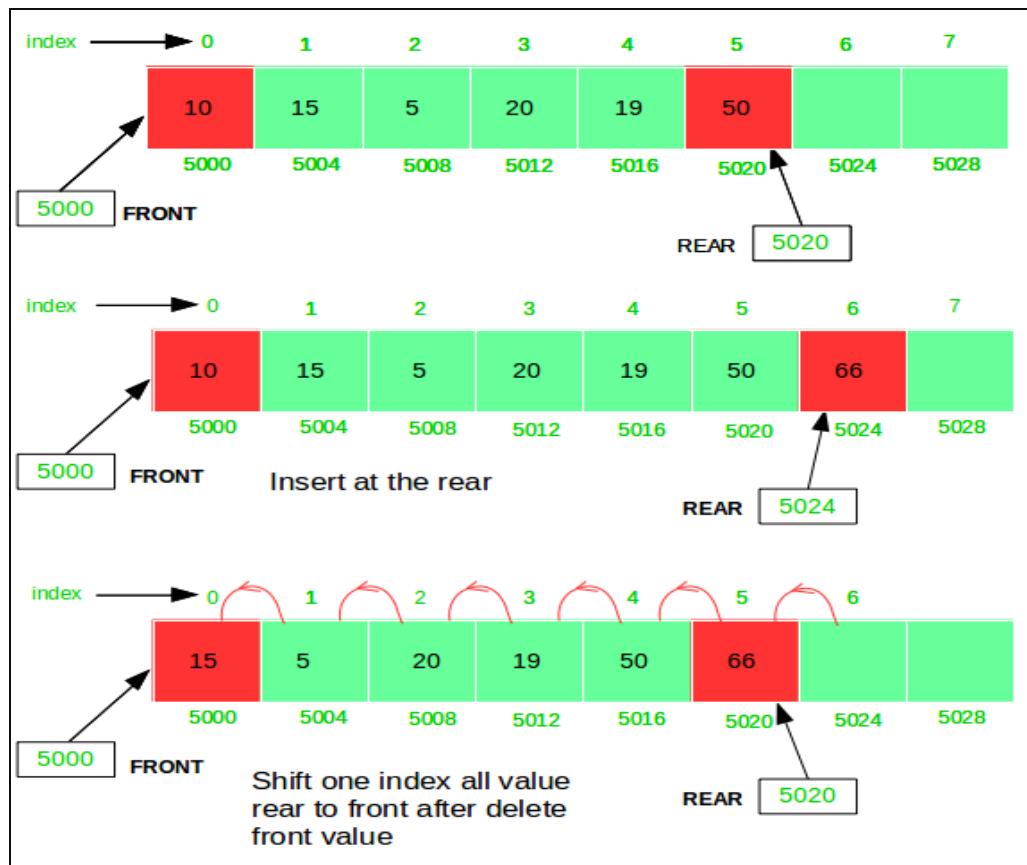
hasilnya jika di run sebagai berikut.

```
Enter the size of the Circular Queue : 7
True
True
True
True
True
True
Queue is full!
Queue is full!
10
20
30
40
50
Studytonight
Queue is empty!
Queue is empty!
Queue is empty!
```

4. Implementasi Array Antrian

Dalam antrian, penyisipan dan penghapusan terjadi di ujung yang berlawanan, sehingga implementasinya tidak sesederhana tumpukan. Untuk menerapkan antrian menggunakan array, buat array ukuran n dan ambil dua variabel depan dan belakang yang keduanya akan diinisialisasi menjadi 0 yang berarti antrian saat ini kosong. Elemen belakang adalah indeks upto yang elemen disimpan dalam array dan depan adalah indeks dari elemen pertama dari array. Sekarang, beberapa pelaksanaan operasi antrian adalah sebagai berikut:

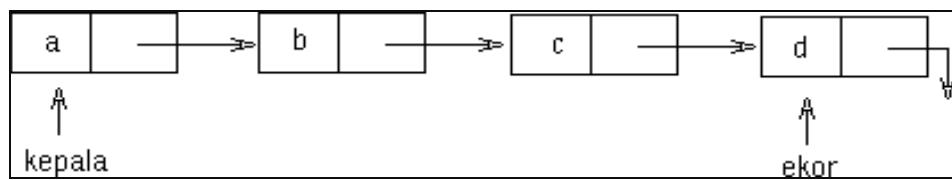
semua elemen dari indeks depan ke belakang.



5. Implementasi Queue dengan Pointer

Implementasi dengan array memiliki keterbatasan jumlah data yang dapat disimpan pada queue. Jika dilihat penjelasan sebelumnya struktur queue dapat diimplementasikan sebagai struktur linked list dengan perlakuan khusus. Sebelumnya telah dijelaskan untuk memanipulasi dibutuhkan dua penunjuk. Maka pada linked list juga terdapat kebutuhan yang sama. Penunjuk pertama

akan menunjuk kepada kepala list (merupakan kepala dari queue) dan penunjuk yang kedua menunjuk kepada item terakhir pada list (merupakan ekor dari queue). Linked list yang digunakan bisa linked list satu arah maupun dua arah. Pada pembahasan ini akan diimplementasikan dengan linked list satu arah. Struktur queue dapat dilihat pada Gambar 5.



Gambar 5. Struktur Queue dengan Linked List

Pada saat inisialisasi pointer kepala dan ekor akan menunjuk ke nil. Ini menunjukkan bahwa queue masih kosong. Pada saat penambahan, cara yang dilakukan sama seperti penambahan linked list satu arah untuk data terakhir hanya ditambah dengan penentuan pointer ekor. Pada saat penghapusan caranya sama pula dengan cara penghapusan linked list data pertama.

C. LATIHAN SOAL

1. Jelaskan yang anda ketahui mengenai antrian dan 3 contoh dalam sehari-hari!
2. Jelaskan yang anda ketahui mengenai Operasi Antrian Menggunakan Array!
3. Jelaskan langkah-langkah DeQueue dalam antrian array!
4. Jelaskan langkah-langkah Menampilkan elemen antrian!
5. Buatlah contoh implementasi antrian dengan array dengan bahasa python!

D. REFERENSI

Basant Agarwal, B. B. (2018). *Hand-On Data Structures And Algorithms With Python*. London: Packt Publishing.

Jodi, U. R. (2020). Algoritma Dan Struktur Data.

Mohamad Islam 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
- 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.

PERTEMUAN 15

BINARY TREE DAN SEARCH TREE

A. TUJUAN PEMBELAJARAN

Setelah mempelajari materi pertemuan ini diharapkan mahasiswa mengerti dan paham *binary tree* dan *search tree*, cara penyisipan dan mencari elemen data search tree dan mampu membuat program queue dalam Python

B. URAIAN MATERI

1. Binary Tree

Pohon pencarian biner adalah pohon di mana setiap node memiliki hingga dua anak. Selain itu, semua nilai di subpohon kiri dari sebuah simpul lebih kecil dari nilai pada akar pohon dan semua nilai pada subpohon kanan dari sebuah simpul lebih besar atau sama dengan nilai pada akar pohon. Akhirnya, subpohon kiri dan kanan juga harus berupa pohon pencarian biner.

Definisi ini memungkinkan untuk menulis kelas di mana nilai dapat dimasukkan ke dalam pohon sambil mempertahankan definisi. Pohon pencarian biner menarik minat akademis. Namun, mereka tidak digunakan banyak dalam praktek. Dalam kasus rata-rata, memasukkan ke dalam pohon pencarian biner membutuhkan $O(\log n)$ waktu. Untuk memasukkan n item ke dalam pohon pencarian biner akan memakan waktu $O(n \log n)$. Jadi, di kasus rata-rata kami memiliki algoritma untuk menyortir urutan item yang dipesan. Namun, dibutuhkan lebih banyak ruang daripada daftar dan algoritme quicksort dapat mengurutkan daftar dengan kompleksitas besar-Oh yang sama. Dalam kasus terburuk, pohon pencarian biner menderita masalah yang sama yang diderita quicksort. Ketika item sudah diurutkan, baik quicksort dan pohon pencarian biner berkinerja buruk. Kompleksitas memasukkan n item ke dalam pohon pencarian biner menjadi $O(n^2)$ dalam kasus terburuk. Pohon menjadi tongkat jika nilainya sudah diurutkan dan pada dasarnya menjadi daftar tertaut.

Ada beberapa properti bagus dari pohon pencarian biner yang dapat diakses secara acak daftar tidak memiliki. Memasukkan ke dalam pohon dapat dilakukan dalam waktu $O(\log n)$ dalam kasus rata-rata saat memasukkan ke

dalam daftar akan memakan waktu $O(n)$. Menghapus dari pohon pencarian biner juga dapat dilakukan dalam waktu $O(\log n)$ dalam kasus rata-rata. Mencari nilai dalam biner pohon pencarian juga dapat dilakukan dalam waktu $O(\log n)$ dalam kasus rata-rata. Jika kita memiliki banyak menyisipkan, menghapus, dan operasi pencarian untuk beberapa algoritma, struktur seperti pohon mungkin berguna. Namun, pohon pencarian biner tidak dapat menjamin kompleksitas $O(\log n)$. Ternyata bahwa ada implementasi struktur pohon pencarian yang dapat menjamin $O(\log n)$ kompleksitas atau lebih baik untuk menyisipkan, menghapus, dan mencari nilai. Beberapa contoh adalah Splay Trees, AVL-Trees, dan B-Trees yang semuanya akan dipelajari nanti dalam teks ini. Binary Tree ialah gambaran seoerti pohon dimana setiap node hanya boleh mempunyai maksimal 2 subtree dan harus terpisah keduanya. Jenis-jenis Binary Tree:

a. Full Binary Tree

Binary Tree setiap nodenya (kecuali leaf) dua child dan setiap subtree memiliki panjang path sama.

b. Complete Binary Tree

Memiliki kemiripan Full Binary Tree tetapi setiap subtree boleh mempunyai panjang path berbeda. Node kecuali leaf memiliki 0 atau 2 child.

c. Skewed Binary Tree

Binary Tree yang seluruh nodenya (kecuali leaf) memiliki satu child.

```

7 class Node:
8
9     def __init__(self,info): #constructor of class
10
11         self.info = info #information for node
12         self.left = None #left leaf
13         self.right = None #right leaf
14         self.level = None #level none defined
15
16     def __str__(self):
17
18         return str(self.info) #return as string
19
20 class searchtree:
21
22     def __init__(self): #constructor of class
23
24         self.root = None
25
26     def create(self,val): #create binary search tree nodes
27
28         if self.root == None:
29
30             self.root = Node(val)
31
32         else:
33
34             current = self.root

```

```

35
36         while 1:
37
38             if val < current.info:
39
40                 if current.left:
41                     current = current.left
42                 else:
43                     current.left = Node(val)
44                     break;
45
46             elif val > current.info:
47
48                 if current.right:
49                     current = current.right
50                 else:
51                     current.right = Node(val)
52                     break;
53
54             else:
55                 break
56
57     def bft(self): #Breadth-First Traversal
58
59         self.root.level = 0
60         queue = [self.root]
61         out = []
62         current_level = self.root.level
63
64         while len(queue) > 0:
65
66             current_node = queue.pop(0)
67
68             if current_node.level > current_level:

```

```

69                 current_level += 1
70                 out.append("\n")
71
72                 out.append(str(current_node.info) + " ")
73
74                 if current_node.left:
75
76                     current_node.left.level = current_level + 1
77                     queue.append(current_node.left)
78
79
80                 if current_node.right:
81
82                     current_node.right.level = current_level + 1
83                     queue.append(current_node.right)
84
85

```

```

86             print "".join(out)
87
88     def inorder(self,node):
89
90         if node is not None:
91
92             self.inorder(node.left)
93             print node.info
94             self.inorder(node.right)
95

```

```

96     def preorder(self,node):
97
98         if node is not None:
99
100             print node.info
101             self.preorder(node.left)
102             self.preorder(node.right)
103
104     def postorder(self,node):
105
106         if node is not None:
107
108             self.postorder(node.left)
109             self.postorder(node.right)
110             print node.info
111
112 tree = searchtree()
113 arr = [8,3,1,6,4,7,10,14,13]
114 for i in arr:
115     tree.create(i)
116 print 'Breadth-First Traversal'
117 tree.bft()
118 print 'Inorder Traversal'
119 tree.inorder(tree.root)
120 print 'Preorder Traversal'
121 tree.preorder(tree.root)
122 print 'Postorder Traversal'
123 tree.postorder(tree.root)

```

Binary Search Tree di Python adalah pohon yang dipesan atau diurutkan yang node internalnya berada dalam cara di mana mereka melekat satu sama lain dan dapat berkomunikasi satu sama lain dengan mulus. Konsep pohon pencarian biner dilengkapi dengan fakta bahwa node diatur dan diatur dengan cara di mana penambahan, pencarian cepat, dan penghapusan item data digunakan untuk manipulasi diikuti oleh pembuatan tabel pencarian dan set dinamis. Pasangan kunci dan nilai memainkan peran penting untuk subtree berada di kiri atau kanan node. Aliran sintaks untuk pohon pencarian biner di Python adalah sebagai berikut:

```

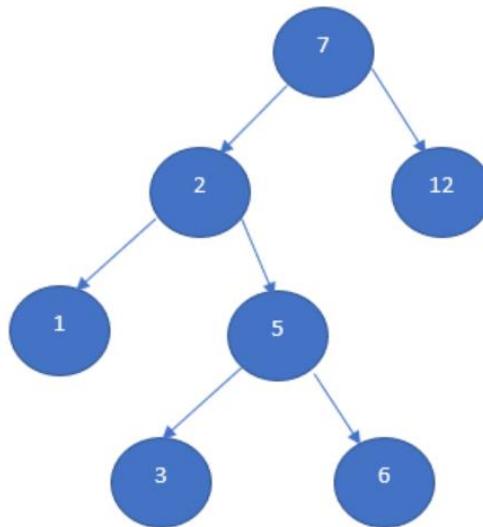
Class A_node:
Def _init_(self, key),
// Put and initialize the key and value pair
// Then a utility function as per requirement can be written
Def insert(root, key)
// define function
//Write the driver program & print with a logger to understand the flow.

```

2. Binary Tree Search di Python

Binary Search Tree bekerja dengan cara berikut mari kita ambil contoh ini untuk diilustrasikan. Mari kita asumsikan bahwa di bawah ini adalah pohon biner dan dengan cara di mana tidak ada pemesanan atau urutan dipertahankan maka bagaimana pohon dengan begitu banyak angka bervariasi membandingkan dan mengatur dirinya sendiri. Pada dasarnya, pohon biner adalah struktur data pohon biner berbasis node yang memiliki karakteristik berikut.

- a. Subtree kiri node berisi node dengan kunci lebih rendah dari kunci node.
- b. Kemudian, subtree kanan node berisi kunci yang lebih besar dari kunci node.
- c. Subtree kiri dan kanan harus memiliki pohon pencarian biner di mana node duplikat tidak semuanya diizinkan.
- d. Semua tiga node di atas harus memiliki kunci untuk operasi seperti penyisipan, pencarian penghapusan minimum dan maksimum.



Karena pohon biner dianggap sebagai struktur data, ia membantu dalam menyimpan data atau angka secara terorganisir.

- a. Urutan hadir sebanding dengan setengah dari nilai-nilai yang tersisa yang membuatnya sama sekali kurang untuk perhitungan atau manipulasi ketika datang untuk mempertahankan kompleksitas waktu agama.

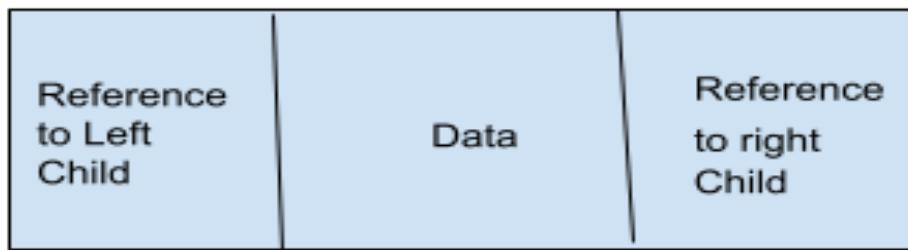
- b. Ini jauh lebih baik daripada waktu linier yang pada dasarnya digunakan untuk menyimpan data yaitu, item yang disimpan dalam bentuk kunci dan nilai yang membuatnya cukup lambat daripada operasi yang sesuai hadir pada tabel hash.
- c. Pohon pencarian biner adalah pohon biner berakar yang node internalnya terutama membuat studi perbandingan untuk menyimpan kunci setelah membuat perbandingan yang membedakan banyak set, multiset, dan item yang ada di pohon.
- d. Pada dasarnya, perbandingan dibuat, dan preorder total dipertahankan dengan subrutin perbandingan tiga arah yang merupakan bagian dari rutinitas yang sebenarnya.
- e. Kompleksitas waktu yang terkait dengan masing-masing traversals memiliki beberapa signifikansi dalam arti itu tergantung pada bagaimana unsur-unsur diputar dari kiri ke kanan atau sebaliknya seperti kasus terburuk, kasus terbaik, atau kasus rata-rata.
- f. Duplikat juga diperbolehkan tetapi sekali lagi pembatasan ada dengan skenario tertentu karena kadang-kadang mereka tidak hadir atau diizinkan.

Dalam serangkaian program tertentu penyisipan elemen dibuat untuk mengurnya dalam urutan dengan beberapa perbandingan yang dibuat sesuai persyaratan di mana seluruh perbandingan dan perhitungan dimulai dari akar sampai akhir. Ada beberapa kompleksitas waktu yang terkait dengan ini di mana pencarian dan penyisipan dibuat dengan perbandingan dan pencarian pada ketinggian dan tingkat bijaksana di mana itu dibuat dengan kompleksitas waktu terburuk $O(h)$.

Binary_Search_Tree di Python digunakan karena banyak alasan dan signifikansi di balik memilih python adalah untuk kemudahan penggunaan dan perilaku mulus yang disediakannya. Bahkan membantu dalam membuat banyak manipulasi di kiri ke kanan atau kanan ke kiri sesuai kebutuhan. Pohon pencarian biner memiliki banyak keuntungan dalam hal membuat pencarian berulang dengan cara yang teratur.

Pohon biner adalah struktur data pohon di mana setiap node dapat memiliki maksimal 2 anak. Ini berarti bahwa setiap node dalam pohon biner dapat memiliki satu, atau dua atau tidak ada anak. Setiap node dalam pohon

biner berisi data dan referensi untuk anak-anaknya. Kedua anak tersebut disebut sebagai anak kiri dan anak yang tepat sesuai dengan posisi mereka. Struktur node dalam pohon biner ditunjukkan pada gambar berikut.



Node dari Pohon Biner

Kita dapat menerapkan simpul pohon biner dalam python sebagai berikut.

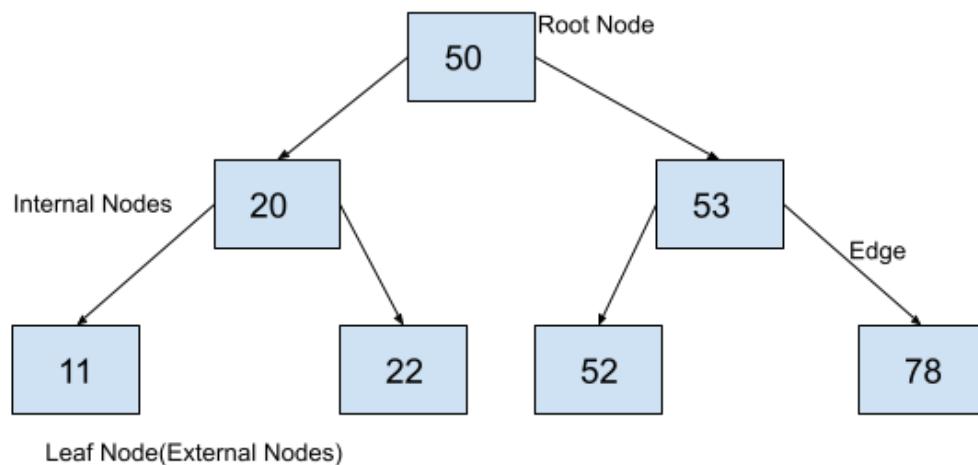
```
class BinaryTreeNode:  
    def __init__(self, data):  
        self.data = data  
        self.leftChild = None  
        self.rightChild=None
```

3. Pohon Pencarian Biner

Pohon pencarian biner adalah struktur data pohon biner dengan sifat-sifat berikut.

- a. Tidak ada elemen duplikat dalam pohon pencarian biner.
- b. Elemen pada anak kiri node selalu kurang dari elemen pada node saat ini.
- c. Subtree kiri node memiliki semua elemen kurang dari node saat ini.
- d. Elemen pada anak yang tepat dari node selalu lebih besar dari elemen pada node saat ini.
- e. Subtree kanan node memiliki semua elemen yang lebih besar dari node saat ini.

Berikut ini adalah contoh pohon pencarian biner yang memenuhi semua properti yang dibahas di atas.



4. Cara Menyisipkan Elemen Search Tree

Penggunaan sifat-sifat pohon pencarian biner untuk memasukkan elemen ke dalamnya. Jika kita ingin menyisipkan elemen pada node tertentu, tiga kondisi mungkin timbul.

- Node saat ini dapat menjadi node kosong yaitu Tidak ada. Dalam hal ini, kita akan membuat node baru dengan elemen yang akan dimasukkan dan akan menetapkan node baru ke node saat ini.
- Elemen yang akan dimasukkan bisa lebih besar dari elemen pada node saat ini. Dalam hal ini, kita akan memasukkan elemen baru di subtree kanan node saat ini sebagai subtree kanan dari setiap node berisi semua elemen yang lebih besar dari node saat ini.
- Elemen yang akan dimasukkan bisa kurang dari elemen pada node saat ini. Dalam hal ini, kita akan memasukkan elemen baru di subtree kiri node saat ini karena subtree kiri dari node apa pun berisi semua elemen yang lebih rendah dari node saat ini.

Untuk memasukkan elemen, kita akan mulai dari node root dan akan memasukkan elemen ke dalam pohon pencarian biner sesuai dengan aturan yang ditentukan di atas. Algoritma untuk memasukkan elemen dalam pohon pencarian biner diimplementasikan seperti dalam Python sebagai berikut.

```

1 class BinaryTreeNode:
2     def __init__(self, data):
3         self.data = data
4         self.leftChild = None
5         self.rightChild = None
6     def insert(root, newValue):
7 # jika pohon pencarian biner kosong, buat node baru dan deklarasikan sbg akar
8         if root is None:
9             root = BinaryTreeNode(newValue)
10            return root
11 # jika newValue kurang dari nilai data dalam root
12         if newValue < root.data:
13             root.leftChild = insert(root.leftChild, newValue)
14         else:
15 # jika newValue lebih besar dari nilai data dalam root
16             root.rightChild = insert(root.rightChild, newValue)
17         return root
18 root = insert(None, 50)
19 insert(root, 20)
20 insert(root, 53)
21 insert(root, 11)
22 insert(root, 22)
23 insert(root, 52)
24 insert(root, 78)
25 node1 = root
26 node2 = node1.leftChild
27 node3 = node1.rightChild
28 node4 = node2.leftChild
29 node5 = node2.rightChild
30 node6 = node3.leftChild

```

```

31 node4 = node2.leftChild
32 node5 = node2.rightChild
33 node6 = node3.leftChild
34 node7 = node3.rightChild
35 print("Root Node adalah:")
36 print(node1.data)
37
38 print("child kiri dari node adalah:")
39 print(node1.leftChild.data)
40
41 print("child kanan dari node adalah:")
42 print(node1.rightChild.data)
43
44 print("Node adalah:")
45 print(node2.data)
46
47 print("child kiri dari node adalah:")
48 print(node2.leftChild.data)
49
50 print("child kanan dari node adalah:")
51 print(node2.rightChild.data)
52
53 print("Node adalah:")
54 print(node3.data)
55
56 print("child kiri dari node adalah:")
57 print(node3.leftChild.data)
58
59 print("child kanan dari node adalah:")
60 print(node3.rightChild.data)
61
62 print("Node adalah:")
63 print(node4.data)
64

```

```
65 print("child kiri dari node adalah::")
66 print(node4.leftChild)
67
68 print("child kanan dari node adalah:")
69 print(node4.rightChild)
70
71 print("Node adalah:")
72 print(node5.data)
73
74 print("child kiri dari node adalah:")
75 print(node5.leftChild)
76
77 print("child kanan dari node adalah:")
78 print(node5.rightChild)
79
80 print("Node adalah:")
81 print(node6.data)
82
83 print("child kiri dari node adalah:")
84 print(node6.leftChild)
85
86 print("child kanan dari node adalah:")
87 print(node6.rightChild)
88
89 print("Node adalah:")
90 print(node7.data)
91
92 print("child kiri dari node adalah:")
93 print(node7.leftChild)
94
95 print("child kanan dari node adalah:")
96 print(node7.rightChild)
```

5. Mencari Elemen di Search Tree

Seperti yang diketahui bahwa pohon pencarian biner tidak dapat memiliki elemen duplikat, dapat mencari elemen apa pun di pohon pencarian biner menggunakan aturan berikut yang didasarkan pada sifat pohon pencarian biner. Kita akan mulai dari akar dan mengikuti properti ini

- a. Jika node saat ini kosong, kita akan mengatakan bahwa elemen tidak ada di pohon pencarian biner.
- b. Jika elemen dalam node saat ini lebih besar dari elemen yang akan dicari, kita akan mencari elemen di subtree kirinya karena subtree kiri dari node apa pun berisi semua elemen yang lebih rendah dari node saat ini.
- c. Jika elemen dalam node saat ini kurang dari elemen yang akan dicari, kami akan mencari elemen di subtree kanannya karena subtree kanan dari setiap node berisi semua elemen yang lebih besar dari node saat ini.
- d. Jika elemen pada node saat ini sama dengan elemen yang akan dicari, kami akan mengembalikan True.

Algoritma untuk mencari elemen dalam pohon pencarian biner berdasarkan sifat di atas diimplementasikan dalam program berikut.

```

1 class BinaryTreeNode:
2     def __init__(self, data):
3         self.data = data
4         self.leftChild = None
5         self.rightChild = None
6
7     def insert(root, newValue):
8         # jika pohon pencarian biner kosong, buat node baru dan deklarasikan sebagai akar
9         if root is None:
10             root = BinaryTreeNode(newValue)
11             return root
12         # jika newValue kurang dari nilai data dalam root, tambahkan ke subtree kiri dan lanjutkan secara rekursif
13         if newValue < root.data:
14             root.leftChild = insert(root.leftChild, newValue)
15         else:
16             # jika newValue lebih besar dari nilai data dalam root, tambahkan ke subtree kanan dan lanjutkan secara rekursif
17             root.rightChild = insert(root.rightChild, newValue)
18         return root
19
20     def search(root, value):
21         # node kosong
22         if root is None:
23             return False
24         # if element is equal to the element to be searched
25         elif root.data == value:
26             return True
27         # jika elemen sama dengan elemen yang akan dicari
28         elif root.data > value:
29             return search(root.leftChild, value)
30         #Elemen yang akan dicari lebih besar dari node saat ini
31         else:
32             return search(root.rightChild, value)
33
34 root = insert(None, 50)
35 insert(root, 20)
36 insert(root, 53)
37 insert(root, 11)
38 insert(root, 22)
39 insert(root, 52)
40 insert(root, 78)
41 print("53 ada di binary tree:", search(root, 53))
42 print("100 ada di binary tree:", search(root, 100))
43

```

Jika di run

('53 ada di binary tree:', True)

('100 ada di binary tree:', False)

C. LATIHAN SOAL

1. Jelaskan yang anda ketahui binary tree!!
2. Jelaskan jenis binary tree yang anda ketahui
3. Jelaskan bagaimana cara mencari elemen pada binary tree!
4. Jelaskan bagaimana cara menyisipkan elemen pada binary tree!
5. Buatlah contoh implementasi binary tree dengan bahasa pemrograman Python!

D. REFERENSI

- Robert Lafore (2003), *Data Structures and Algorithms in Java*, Sams Publishing.
- 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). Pendekripsi Kantuk Secara Real Time Menggunakan Pustaka Opencv Dan Dlib Python. Sainstech : Jurnal Penelitian Dan Pengkajian Sains Dan Teknologi.

PERTEMUAN 16

TREE TRAVERSAL

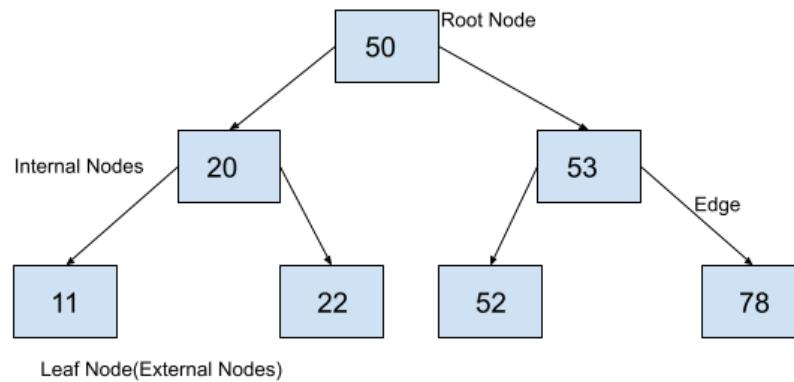
A. TUJUAN PEMBELAJARAN

Setelah mempelajari materi pertemuan ini diharapkan mahasiswa mengerti dan paham tree traversal, Algoritma Level Order Tree Traversal, Traversal Tree Inorder, pre-post Traversal dan mampu membuat tree traversal dalam Python

B. URAIAN MATERI

1. Definisi Tree Traversal

Algoritma traversal tree urutan tingkat adalah algoritma traversal tree yang luas pertama. Ini berarti bahwa saat melintasi tree, kita pertama kali melintasi semua elemen pada tingkat saat ini sebelum pindah ke tingkat berikutnya. Untuk memahami konsep ini, dengan pertimbangkan tree pencarian biner berikut.



Tingkat urutan traversal tree di atas akan sebagai berikut. Mulai dari akar dan mencetak 50. Setelah itu kita pindah ke tingkat berikutnya dan mencetak 20 dan 53. Setelah level ini, kami pindah ke level lain dan mencetak 11, 22, 52, dan 78. Jadi, urutan tingkat traversal di atas tree adalah 50, 20, 53, 11, 22, 52, 78.

Traversal adalah proses untuk mengunjungi semua node dari tree dan mungkin mencetak nilai-nilai. Hal tersebut beralasan bahwa semua node yang terhubung melalui link selalu mulai dari akar (kepala) node, tidak dapat secara

acak mengakses node di tree. Secara umum melintasi tree dengan mencari atau menemukan item atau kunci yang diberikan dengan mencetak semua nilai yang dikandungnya.

Pada materi ini telah membahas algoritma tree traversal urutan tingkat di Python. Tingkat urutan traversal tree dapat digunakan untuk menemukan binary tree in di Python. Algoritma ini telah digunakan untuk mengimplementasikan berbagai algoritma lain yang akan kita bahas nanti dalam seri ini. Berikutnya, kita akan menerapkan algoritma traversal tree lainnya seperti traversal tree in-order, traversal tree pre-order, dan algoritma traversal tree pasca-pesanan. Untuk mempelajari lebih lanjut tentang struktur data lainnya, dimana dapat membaca artikel ini di Daftar Tertaut di Python.

Traversal adalah proses melintasi node struktur data. Tetapi jika kita menggunakan struktur data seperti tumpukan, antrian, atau daftar terkait maka menjadi sulit untuk melintasi node karena struktur data ini linier, dan karenanya kompleksitas waktu traversal meningkat. Oleh karena itu, kita menggunakan struktur data tree (terutama tree biner) ketika kita harus melakukan proses traversal dan menemukan elemen dengan mudah. Dengan menggunakan struktur data tree, menjadi mudah untuk melintasi node dibandingkan dengan struktur data lainnya karena tree menyimpan elemen secara hierarkis, dan karenanya, kita dapat melintasi elemen dengan prioritas dan jalurnya sesuai dengan itu. Sekarang, ada 3 metode utama untuk traversal tree dalam python dengan rekursi menggunakan DFS yaitu:

- a. Inorder Traversal (kiri, akar, kanan)
- b. Preorder Traversal (akar, kiri, kanan)
- c. Postorder Traversal (kiri, kanan, akar)

Ingat bahwa kita akan menggunakan fungsi rekursif saat melintasi tree dan memanggil fungsi lagi dan lagi sampai kita akan melintasi semua node tree.

Selain itu, kita juga dapat menemukan traversal tree tanpa menggunakan metode rekursi. Satu-satunya perbedaan antara kedua metode ini adalah bahwa traversal tree dalam python tanpa menggunakan proses rekursi menggunakan struktur data tumpukan sementara traversal tree dalam python dengan rekursi menggunakan struktur data array.

2. Algoritma Level Order Tree Traversal

Keharusan memproses elemen pada setiap tingkat satu per satu dalam traversal tree urutan tingkat, kita dapat menggunakan pendekatan berikut. Kita akan mulai dari root node dan akan mencetak nilainya. Setelah itu, kita akan memindahkan kedua anak dari node root ke antrian. Antrian akan digunakan untuk mengandung elemen yang harus diproses selanjutnya. Setiap kali kita memproses elemen, kita akan menempatkan anak-anak dari elemen itu ke dalam antrian. Dengan cara ini semua elemen pada tingkat yang sama akan didorong ke dalam antrian dalam urutan terus menerus dan akan diproses dalam urutan yang sama. Algoritma untuk tingkat urutan traversal tree dapat dirumuskan sebagai berikut.

- a. Masukkan akar ke dalam Q.
- b. Keluarkan node dari Q.
- c. Jika node kosong yaitu Tidak ada, goto 8.
- d. Cetak elemen dalam node.
- e. Masukkan anak kiri dari node saat ini ke Q.
- f. Masukkan anak yang tepat dari node saat ini ke Q
- g. Periksa apakah Q kosong. Jika ya, berhentilah. Lainnya, goto 3.

Dalam program dibawah inih terlebih dahulu menerapkan tree pencarian biner yang diberikan pada gambar. Kemudian setelah menggunakan algoritma untuk tingkat urutan traversal tree untuk melintasi tree pencarian biner di Python. Seperti yang dapat diketahui untuk diproses. Juga, unsur-unsur depth in the tree telah dicetak dari kiri ke kanan dalam urutan kedalamannya di tree. Akar dicetak terlebih dahulu sementara node daun akhirnya dicetak

Dalam artikel ini, telah membahas algoritma traversal tree urutan tingkat di Python. Tingkat urutan traversal tree dapat digunakan untuk menemukan lebar tree biner di Python. Algoritma ini telah digunakan untuk mengimplementasikan berbagai algoritma lain yang akan kita bahas nanti dalam seri ini. Pada artikel berikutnya, kita akan menerapkan algoritma traversal tree lainnya seperti traversal tree in-order, traversal tree pre-order, dan algoritma traversal tree pasca-pesanan. Untuk mempelajari lebih lanjut tentang struktur data lainnya, dapat membaca artikel ini di Daftar Tertaut diPython.

Nantikan lebih banyak artikel tentang implementasi algoritma yang berbeda di Python.

```
1 from queue import Queue
2 class BinaryTreeNode:
3     def __init__(self, data):
4         self.data = data
5         self.leftChild = None
6         self.rightChild = None
7     def levelorder(root):
8         Q = Queue()
9         Q.put(root)
10        while (not Q.empty()):
11            node = Q.get()
12            if node == None:
13                continue
14            print(node.data)
15            Q.put(node.leftChild)
16            Q.put(node.rightChild)
17    def insert(root, newValue):
18        # if binary search tree is empty, create a new node and declare it as root
19        if root is None:
20            root = BinaryTreeNode(newValue)
21            return root
22        # if newValue is less than value of data in root, add it to left subtree and proceed recursively
23        if newValue < root.data:
24            root.leftChild = insert(root.leftChild, newValue)
25        else:
26            # if newValue is greater than value of data in root, add it to right subtree and proceed recursively
27            root.rightChild = insert(root.rightChild, newValue)
28    return root
29 root = insert(None, 50)
30 insert(root, 20)
31 insert(root, 53)
32 insert(root, 11)
33 insert(root, 22)
34 insert(root, 52)
35 insert(root, 78)
36 print("Level Order traversal of the binary tree is:")
37 levelorder(root)
```

Jika dirun maka hasilnya sebagai berikut,

50

20

53

11

22

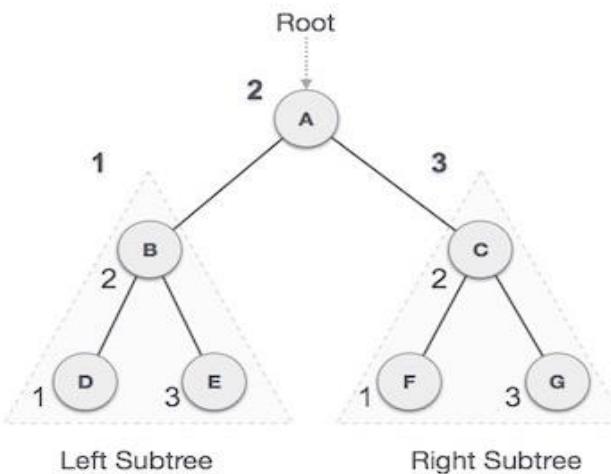
52

78

3. Traversal Tree Inorder

Dalam metode ini adalah node kiri yang dikunjungi terlebih dahulu dan kemudian basis, atau sub node dilalui dan akhirnya pada akhirnya pada akhirnya sub-tree kanan. Sub tree kiri dikunjungi terlebih dahulu sampai node tercapai. Ketika lokasi saat ini adalah akar dari sub node, sub tree yang tepat dilalui. Setelah menyelesaikan traversal tree sisi kiri, kami mengikuti proses yang sama sampai kami mencapai simpul akar tree dan mengulangi proses sampai semua node dikunjung. Harus selalu ingat bahwa setiap simpul mungkin merupakan subtree sendiri. Jika tree biner dilalui Inorder, output akan menghasilkan nilai-nilai kunci diurutkan dalam urutan menaik.

Jalur sesuai contoh struktur adalah: Node 1.1→Node 1→Node 1.2→Root→Node



Mulai dari A, dan mengikuti traversal Inorder bergerak ke kiri subtree B. B juga dilalui Inorder. Proses ini berlangsung sampai semua node yang dikunjungi. Output dari inorder traversal tree ini akan tampil seperti berikut.

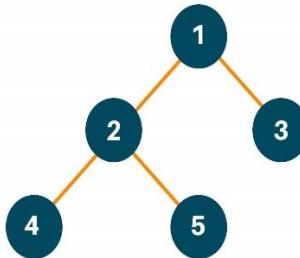
$$D \rightarrow B \rightarrow E \rightarrow A \rightarrow F \rightarrow C \rightarrow G$$

Dengan menggunakan metode traversal inorder, pertama-tama kami mengunjungi subtree kiri tree asli. Kemudian kita akan melintasi simpul akar tree dan terakhir subtree kanan dari tree asli. Algoritma untuk Inorder Tree Traversal menggunakan Python.

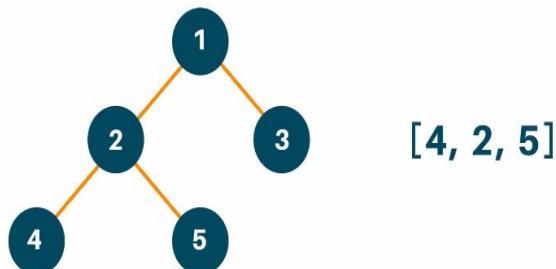
- Memanggil Inorder (subtree kiri)
- Kunjungi node root
- Memanggil Inorder (subtree kanan)

Misalnya:

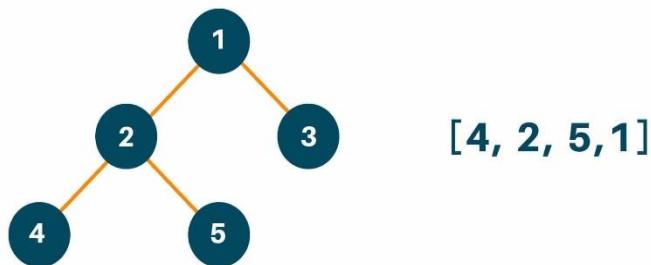
Mari kita pertimbangkan tree biner berikut:



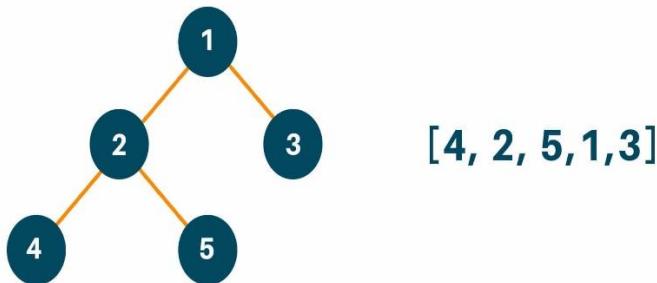
Sekarang sesuai dengan metode traversal tree inorder dalam python, pertama-tama kita melintasi subtree kiri tree asli. Ingat, bahkan saat melintasi subtree kiri, kita akan mengikuti proses yang sama yaitu kiri -> akar -> kanan jika node anak kiri tree asli memiliki node anak lebih jauh. Setelah melintasi subtree kiri, kami akan menambahkan hasilnya dalam array seperti yang ditunjukkan di bawah ini.



Setelah mengikuti langkah pertama, kita akan melintasi simpul akar tree asli seperti yang ditunjukkan di bawah ini.



Terakhir, kita akan melintasi subtree kanan setelah proses yang sama yaitu kiri -> akar -> kanan jika node anak kanan dari tree asli memiliki lebih dari satu node anak.



Kode Python dari Preorder Tree Traversal dengan rekursi

```

1 class TreeNode:
2
3     def __init__(self, val):
4         self.val = val
5         self.left = None
6         self.right = None
7
8     def preorderTraversal(self):
9         answer = []
10
11         preorderTraversalUtil(self, answer)
12         return answer
13
14     def preorderTraversalUtil(self, answer):
15
16         if self is None:
17             return
18
19         answer.append(self.val)
20
21         preorderTraversalUtil(self.left, answer)
22
23         preorderTraversalUtil(self.right, answer)
24
25     return
26
27 root = TreeNode(1)
28 root.left = TreeNode(2)
29 root.right = TreeNode(3)
30 root.left.left = TreeNode(4)
31 root.left.right = TreeNode(5)
32
33 print(preorderTraversal(root))
34
35

```

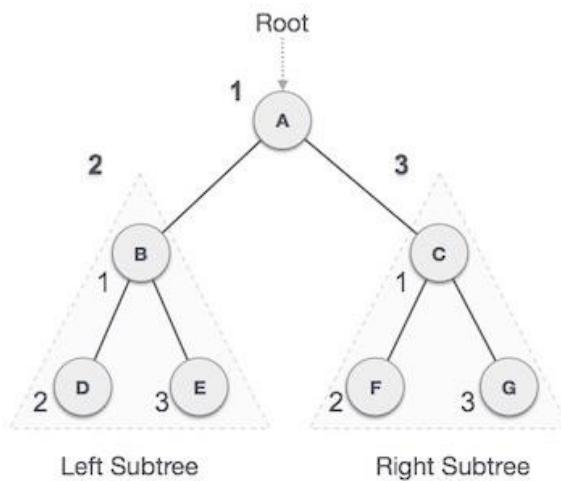
Hasilnya jika di run : [1, 2, 4, 5, 3]

4. Pre-order Traversal

Metode ini adalah simpul akar yang dikunjungi terlebih dahulu dan kemudian sub-tree kiri dilalui dan akhirnya pada akhirnya sub-tree kanan. Akar adalah lokasi pertama yang dikunjungi. Kemudian bergerak ke kiri cabang dan terus berjalan sampai bertemu ujung tree. Sesampaman di sana ia mulai melihat sisi kanan level sebelum mengangkat pada tingkat node atas.

Jalur sesuai contoh struktur adalah: Root→Node 1→Node 1.1→Node 1.2→Node 2.

Dalam metode traversal ini, akar simpul dikunjungi pertama, maka subtree kiri dan akhirnya subtree kanan.



Dimulai dari A, dan mengikuti pre-order traversal, pertama kita mengunjungi A itu sendiri dan kemudian pindah ke subtree kiri B. B juga dilalui pre-order. Proses ini berlangsung sampai semua node yang dikunjungi. Output dari pre-order traversal tree ini akan

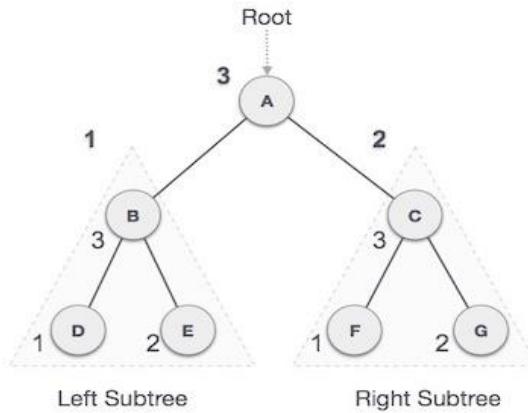
$$A \rightarrow B \rightarrow D \rightarrow E \rightarrow C \rightarrow F \rightarrow G$$

5. Post-order Traversal

Pada metode ini adalah sisi kiri dikunjungi terlebih dahulu dan kemudian sub-tree sisi kanan dilalui dan akhirnya di node. Lokasi pertama adalah node daun paling kiri. Kemudian bergerak ke node kanan sebelum pindah ke tingkat atas node dan mengikuti traversal yang sama sampai mencapai node dasar tree.

Jalur sesuai contoh struktur adalah: Node 1.1→Node 1.2→ Node 1→Node 2→ Root.

Dalam metode traversal ini, akar simpul yang terakhir dikunjungi, maka nama itu. Pertama kita melintasi subtree kiri, maka subtree kanan dan akhirnya simpul akar.



Dimulai dari A, dan mengikuti pre-order traversal, pertama kita kunjungi subtree kiri B. B juga dilalui pasca-order. Proses ini berlangsung sampai semua node yang dikunjungi. Output dari post-order traversal tree ini akan :

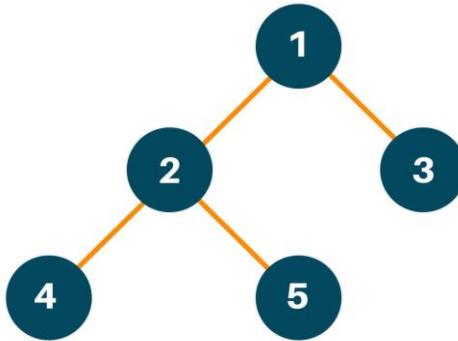
$$D \rightarrow E \rightarrow B \rightarrow F \rightarrow G \rightarrow C \rightarrow A$$

Python tree traversal didefinisikan sebagai algoritma asan yang diimplementasikan dalam bahasa pemrograman Python dan memungkinkan pengembang untuk mengembangkan aplikasi yang mengharuskan setiap elemen untuk dikunjungi sebagai bagian dari eksekusi. Melintasi berarti mengunjungi dan tree adalah implementasi struktur data abstrak dalam berbagai bahasa pemrograman. Implementasi tree bersifat non-linear dibandingkan dengan struktur data lain seperti array atau daftar di mana elemen hadir secara linear. Komponen tree adalah akar dan node anak beberapa di antaranya berakhir pada simpul tertentu dan disebut sebagai daun dan yang lainnya menciptakan lebih banyak sub-tree. Ada persyaratan struktur data untuk itu akan disebut sebagai tree yang akan kita bahas nanti dalam artikel ini.

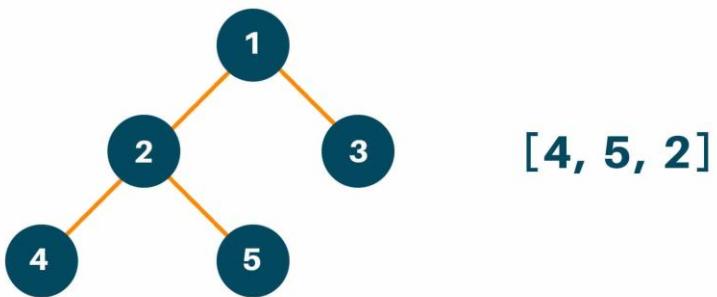
Dengan menggunakan metode traversal tree postorder, pertama kali mengunjungi subtree kiri tree asli diikuti oleh subtree kanan dan terakhir simpul akar tree asli. Algoritma Postorder Tree Traversal menggunakan Python

- a. Memanggil subtree kiri
- b. Memanggil subtree kanan
- c. Kunjungi simpul akar

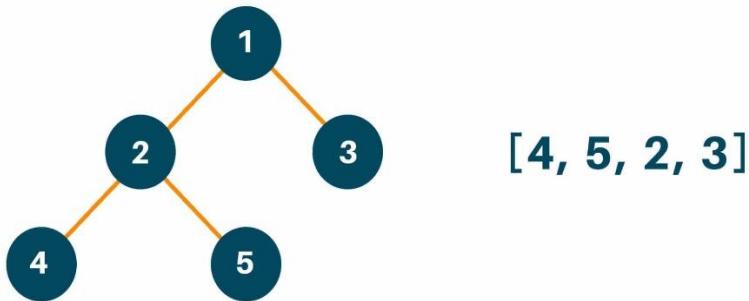
Misalnya pertimbangkan contoh tree berikut:



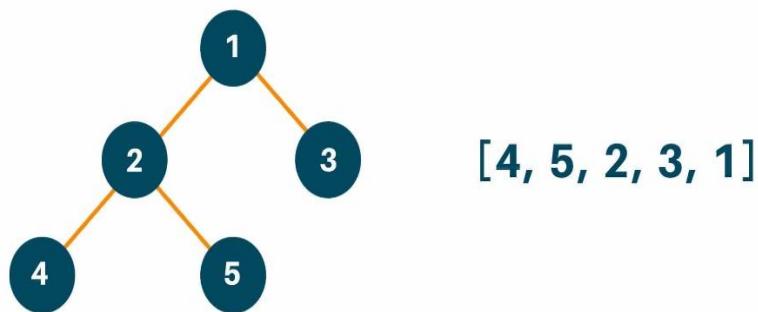
Dengan menggunakan metode traversal postorder, pertama-tama kita akan melintasi subtree kiri tree asli. Ingat bahwa kita akan mengikuti proses yang sama melintasi subtree kiri yaitu kiri -> kanan -> root jika subtree kiri memiliki lebih dari satu node anak dan kemudian menempatkan hasilnya dalam array jawaban seperti yang ditunjukkan pada gambar.



Nantinya kita akan melintasi subtree kanan tree asli sama seperti yang kita lakukan dengan subtree kiri dan menambahkan jawabannya dalam array hasil seperti yang ditunjukkan di bawah ini.



Dan terakhir, kita akan melintasi simpul akar tree asli dan menyelesaikan metode traversal kita seperti yang ditunjukkan pada gambar di bawah ini.



Contoh sourcecode sebagai berikut.

16.6 Traversal Tree di Python

Sekarang terlebih dahulu memahami struktur tree sehingga ketika berbicara tentang traversal itu akan memastikan bahwa kita memahami rinciannya melalui daya tarik visual dari struktur tree dan mendapatkan pemahaman yang lebih baik tentang berbagai cara traversal tree. Dalam pendahuluan kami memahami bahwa tree adalah struktur di mana ada satu node akar dan node itu memberi jalan bagi node anak dan setiap node anak dapat mengakhiri di sana atau membentuk lebih banyak sub tree. Sekarang traversal berarti bahwa masing-masing node di tree harus dikunjungi dan traversal mungkin timbul jika seseorang perlu menemukan misalnya, maksimum atau minimum dalam struktur data berbasis tree.

Dalam struktur di atas akar adalah tempat di mana pembentukan tree dimulai dan Node 1 mengarah ke sub tree lebih lanjut tetapi node 2 berhenti di sana sendiri. Sekarang ada banyak kondisi struktur data yang disebut sebagai

tree. Beberapa dari mereka adalah, harus ada satu dan hanya satu akar, berikutnya adalah bahwa satu node anak tidak dapat memiliki koneksi ke lebih dari satu node induk, yang lain menjadi satu node tidak dapat memiliki dirinya sebagai node anak.

```
1 class TreeNode:
2
3     def __init__(self, val):
4         self.val = val
5         self.left = None
6         self.right = None
7
8     def postorderTraversal(root):
9         answer = []
10
11     postorderTraversalUtil(root, answer)
12     return answer
13
14     def postorderTraversalUtil(root, answer):
15
16         if root is None:
17             return
18
19         postorderTraversalUtil(root.left, answer)
20
21         postorderTraversalUtil(root.right, answer)
22
23         answer.append(root.val)
24
25     return
26
27 root = TreeNode(1)
28 root.left = TreeNode(2)
29 root.right = TreeNode(3)
30 root.left.left = TreeNode(4)
31 root.left.right = TreeNode(5)
32
33 print(postorderTraversal(root))
34
```

Hasilnya [4, 5, 2, 3, 1]

6. Kompleksitas Waktu

Dalam traversal tree di python menggunakan rekursi, kompleksitas waktu adalah $O(n)$ di mana ada n node di tree. Sedangkan kompleksitas ruang juga $O(n)$ untuk node n hadir dalam array jawaban aplikasi.

- a. Metode traversal inorder digunakan untuk traversal tree untuk mendapatkan urutan node yang tidak menurun.
- b. Metode pra-konsangkas digunakan untuk mendapatkan ekspresi awalan dari tree ekspresi. Juga, preorder traversal membantu membuat salinan tree.
- c. Metode traversal postorder digunakan untuk mendapatkan ekspresi postfix dari tree ekspresi. Juga, metode traversal postorder membantu menghapus tree.

Oleh karena itu, belajar dan memahami aplikasi dan penggunaan traversal tree di dunia nyata membuktikan bahwa itu adalah topik penting untuk dipelajari untuk setiap programmer. Oleh karena itu, dalam artikel di atas, kami mempelajari traversal tree dalam python menggunakan rekursi dan metode melintasi dengan sifat rekursif. Juga, kami mempelajari kode python untuk traversal tree dan output yang sesuai.

C. LATIHAN SOAL

1. Jelaskan yang anda ketahui mengenai tree Tranversal!
2. Jelaskan yang anda ketahui mengenai Algoritma Level Order Tree Traversal!
3. Jelaskan yang anda ketahui mengenai Traversal Tree Inorder!
4. Jelaskan yang anda ketahui mengenai Pre-order Traversal!
5. Buatlah contoh implementasi tree Tranversal dengan bahasa python!

D. REFERENSI

Distance Untuk Deteksi Kemiripan Gambar. *Repository Its.*

Basant Agarwal, B. B. (2018). *Hand-On Data Structures And Algorithms With Python*. London: Packt Publishing.

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

Jodi, U. R. (2020). Algoritma Dan Struktur Data.

- 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.
- 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). Pendekripsi Kantuk Secara Real Time Menggunakan Pustaka Opencv Dan Dlib Python. *Sainstech : Jurnal Penelitian Dan Pengkajian Sains Dan Teknologi*.
- Zein, A. (2019). Pendekripsi Penyakit Malaria Menggunakan Medical Images Analisis Dengan Deep Learning Python. *Jurnal Penelitian Dan Pengkajian Sains Dan Teknologi*.

PERTEMUAN 17

GRAPH

A. TUJUAN PEMBELAJARAN

Setelah mempelajari materi pertemuan diharapkan mahasiswa mengerti dan paham sejarah dan termininologi graph, manfaat graph dan analisis jaringan graph dan mampu membuat ghaph dalam Python.

B. URAIAN MATERI

1. Sejarah Graph

Asal usul teori dapat ditelusuri kembali ke masalah jembatan Konigsberg (sekitar tahun 1730-an). Masalahnya menanyakan apakah tujuh jembatan di kota Konigsberg dapat dilalui di bawah kendala berikut.

- a. Tidak menggandakan kembali
- b. Anda berakhir di tempat yang sama dengan yang anda mulai

Ini sama dengan menanyakan apakah multigraf dari 4 node dan 7 tepi memiliki siklus Eulerian (Siklus Eulerian adalah jalur Eulerian yang dimulai dan berakhir pada Vertex yang sama. Dan jalur Eulerian adalah jalan dalam Graph yang melintasi setiap tepi tepat sekali. Terminologi lebih lanjut diberikan di bawah ini). Masalah ini menyebabkan konsep Eulerian Graph. Dalam kasus masalah jembatan Konigsberg, jawabannya tidak dan pertama kali dijawab oleh Euler.

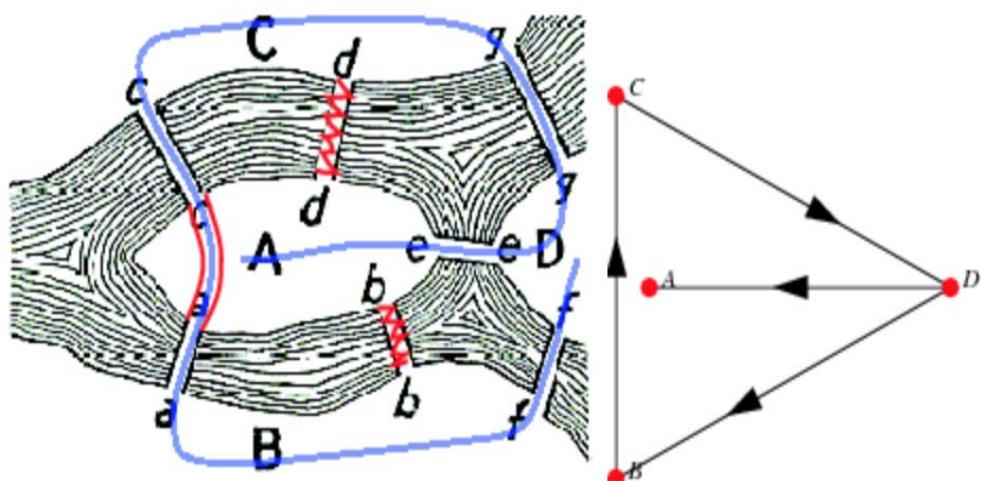
Pada tahun 1840, A.F Mobius memberikan gagasan graph lengkap dan graph bipartit dan Kuratowski membuktikan bahwa planar melalui masalah rekreasi. Konsep pohon, (graph yang terhubung tanpa siklus) diimplementasikan oleh Gustav Kirchhoff pada tahun 1845, dan ia menggunakan ide-ide teoritis graph dalam perhitungan arus dalam jaringan listrik atau sirkuit.

Pada tahun 1852, Thomas Gutherie menemukan empat masalah warna yang terkenal. Kemudian pada tahun 1856, Thomas. P. Kirkman dan William R.Hamilton mempelajari siklus pada polyhydra dan menemukan konsep yang

disebut graph Hamiltonian dengan mempelajari perjalanan yang mengunjungi situs-situs tertentu tepat sekali. Pada tahun 1913, H.Dudeney menyebutkan masalah teka-teki. Meskipun empat masalah warna ditemukan itu diselesaikan hanya setelah satu abad oleh Kenneth Appel dan Wolfgang Haken. Kali ini dianggap sebagai kelahiran Teori Graph.

Caley mempelajari bentuk analisis tertentu dari kalkulus diferensial untuk mempelajari pohon-pohon. Ini memiliki banyak implikasi dalam kimia teoritis. Hal ini menyebabkan penemuan teori graph enumeratif. Apapun bagaimana istilah "Graph" diperkenalkan oleh Sylvester pada tahun 1878 di mana ia menggambar analogi antara "invarian Quantic" dan kovarians aljabar dan diagram molekuler.

Pada tahun 1941, Ramsey bekerja pada warna yang mengarah pada identifikasi cabang lain dari teori graph yang disebut teori graph ekstrim. Pada tahun 1969, empat masalah warna diselesaikan dengan menggunakan komputer oleh Heinrich. Studi tentang konektivitas graph asymptotic memunculkan teori graph acak. Sejarah Teori Graph dan Topologi juga terkait erat. Mereka berbagi banyak konsep dan teorema umum.



Gambar 17. 1 Sejarah Graph

2. Terminologi Graph

Graph adalah struktur data yang sangat berguna dalam memecahkan banyak tantangan matematika yang penting. Misalnya topologi jaringan komputer atau menganalisis struktur molekul senyawa kimia. Mereka juga digunakan dalam lalu lintas kota atau perencanaan rute dan bahkan dalam bahasa manusia dan tata bahasa mereka. Semua aplikasi ini memiliki tantangan umum untuk melintasi graph menggunakan tepinya dan memastikan bahwa semua node graph dikunjungi. Ada dua metode umum yang ditetapkan untuk melakukan traversal ini yang dijelaskan di bawah ini.

Teori Graph adalah bidang studi yang luas yang didasarkan pada gagasan sederhana tentang titik-titik individu - yang dikenal sebagai simpul - dihubungkan oleh garis yang dikenal sebagai tepi, yang masing-masing mungkin memiliki nilai numerik terkait yang disebut berat dan mungkin juga arah.

Kumpulan simpul, tepi, bobot, dan arah sederhana ini dikenal sebagai graph (jangan dikelirukan dengan graph data yang lebih akrab atau fungsi matematika) dan dapat digunakan untuk mewakili banyak situasi dunia nyata atau konsep abstrak. Setelah memiliki graph yang disimpan dalam beberapa jenis struktur data ada banyak algoritma yang dapat diterapkan untuk memecahkan masalah, terutama yang optimasi. Graph seperti pohon bahkan, dalam pengertian matematis, pohon adalah sejenis graph. Di komputer pemrograman, bagaimanapun, graph digunakan dengan cara yang berbeda daripada pohon.

Struktur data yang diperiksa sebelumnya dalam buku ini memiliki arsitektur ditentukan oleh algoritma yang digunakan pada mereka. Misalnya, pohon biner dibentuk seperti itu karena bentuk itu memudahkan untuk mencari data dan menyisipkan yang baru data. Tapi di pohon mewakili cara cepat untuk mendapatkan dari simpul ke simpul.

Graph, di sisi lain, sering memiliki bentuk yang ditentukan oleh masalah fisik atau abstrak. Misalnya, node dalam graph dapat mewakili kota, sedangkan tepi dapat mewakili rute penerbangan maskapai antar kota. Lainnya lagi contoh abstrak adalah graph yang mewakili individu tugas yang diperlukan untuk menyelesaikan suatu proyek. Dalam graph, node dapat mewakili tugas, sementara tepi terarah (dengan panah di salah satu ujungnya) menunjukkan

tugas mana yang harus diselesaikan sebelum yang lain. Dalam kedua kasus, bentuk graph muncul dari dunia nyata tertentu situasi.

Sebelum melangkah lebih jauh, kita harus menyebutkan bahwa, ketika membahas graph, node secara tradisional disebut simpul (tunggal adalah simpul). Ini mungkin karena nomenklaturnya untuk graph lebih tua dari itu untuk pohon, yang muncul dalam matematika berabad-abad lalu. Pohon lebih erat terkait dengan ilmu komputer. Namun, kedua istilah digunakan lebih atau kurang secara bergantian.

Dalam graph, lingkaran mewakili simpang susun jalan bebas hambatan, dan garis lurus yang menghubungkan lingkaran mewakili segmen jalan bebas hambatan. Lingkaran adalah simpul, dan garis adalah tepi. Simpul biasanya diberi label dengan cara tertentu seringkali, seperti yang ditunjukkan di sini, dengan huruf alfabet. Setiap sisi dibatasi oleh dua simpul pada ujungnya. Graph tidak berusaha untuk mencerminkan posisi geografis yang ditampilkan pada peta; saya hanya menunjukkan hubungan simpul dan tepi yaitu, tepi mana yang terhubung ke simpul mana. Termologi grafh disimpulkan dibawah ini.

- a. Simpul dan disebut tepi uvend vertices(u,v)
- b. Jika dua tepi memiliki hal yang sama mereka end verticesParallel
- c. Tepi dari bentuk adalah (v,v) loop
- d. Graph adalah jika tidak memiliki tepi paralel dan loopsimple
- e. Graph dikatakan jika tidak memiliki tepi. Artinya kosongEmptyE
- f. Graph adalah jika tidak memiliki simpul. Artinya dan kosongNull GraphVE
- g. Graph dengan hanya 1 Vertex adalah graphTrivial
- h. Tepi adalah jika mereka memiliki simpul umum. Vertices adalah jika mereka memiliki tepi yang samaAdjacentAdjacent
- i. Tingkat simpul, ditulis sebagai, adalah jumlah tepi dengan sebagai simpul akhir. Dengan konvensi, kita menghitung loop dua kali dan tepi paralel berkontribusi secara terpisah. $vd(v)$
- j. Vertices terisolasi adalah simpul dengan derajat 1. Vertices diisolasi(1)
- k. Graph Selesai jika set tepinya berisi setiap tepi yang mungkin antara semua simpul

- I. A dalam Graph adalah urutan bentuk yang terbatas dan bergantian. WalkG = (V,E) ViEiViEi terdiri dari simpul dan tepi graph G
- m. Berjalan adalah jika simpul awal dan terakhir berbeda. Berjalan adalah jika simpul awal dan terakhir adalah sama. OpenClosed
- n. A Walk adalah jika tepi apapun dilalui paling cepat Trail
- o. Trail adalah jika setiap vertex dilalui paling cepat (Kecuali untuk berjalan-jalan tertutup) Path
- p. Jalur Tertutup adalah Analog dengan sirkuit listrik Circuit

Pada bagian ini akan terlihat beberapa konsep yang berguna untuk Analisis Data (tanpa urutan tertentu). Harap dicatat bahwa ada lebih banyak konsep yang membutuhkan kedalaman yang berada di luar cakupan artikel ini. Jadi mari kita masuk ke dalamnya.

- a. Panjang Jalur Rata-rata

Rata-rata panjang jalur terpendek untuk semua pasangan node yang mungkin. Memberikan ukuran 'sesak' graph dan dapat digunakan untuk memahami seberapa cepat / mudah sesuatu mengalir dalam Jaringan ini.

- b. *BFS* dan *DFS*

Luasnya pencarian pertama dan pencarian pertama Depth adalah dua algoritma berbeda yang digunakan untuk mencari Node dalam Graph. Mereka biasanya digunakan untuk mencari tahu apakah kita dapat mencapai Node dari Node tertentu. Ini juga dikenal sebagai Graph Traversal.

- c. Tujuan dari BFS adalah untuk melintasi Graph sedekat mungkin dengan node root, sedangkan algoritma DFS bertujuan untuk bergerak sejauh mungkin dari node root.

- d. Sentralitas

Salah satu alat konseptual yang paling banyak digunakan dan penting untuk menganalisis jaringan. Centrality bertujuan untuk menemukan node yang paling penting dalam jaringan. Mungkin ada gagasan yang berbeda tentang "penting" dan karenanya ada banyak langkah sentralitas. Langkah-langkah sentralitas sendiri memiliki bentuk klasifikasi (atau jenis tindakan sentralitas).

Ada langkah-langkah yang ditandai dengan aliran di sepanjang tepi dan yang ditandai dengan Struktur Berjalan.

Beberapa yang paling umum digunakan adalah:

a. Degree Centrality

Definisi Sentralitas pertama dan konseptual yang paling sederhana. Ini adalah jumlah tepi yang terhubung ke node. Dalam kasus graph yang diarahkan, kita dapat memiliki ukuran sentralitas 2 derajat. Inflow dan Outflow Centrality

b. Closeness Centrality

Dari node adalah panjang rata-rata jalur terpendek dari node ke semua node lainnya

c. Antaraness Centrality

Berapa kali node hadir dalam jalur terpendek antara 2 node lainnya

Langkah-langkah sentralitas ini memiliki varian dan definisi dapat diimplementasikan menggunakan berbagai algoritma. Secara keseluruhan, ini berarti sejumlah besar definisi dan algoritma.

a. Kepadatan Jaringan

Ukuran berapa banyak tepi graph memiliki. Definisi yang sebenarnya akan bervariasi tergantung pada jenis Graph dan konteks di mana pertanyaan diajukan. Untuk Graph lengkap yang tidak diarahkan, Kepadatan adalah 1, sedangkan 0 untuk Graph kosong. Kepadatan graph bisa lebih besar dari 1 dalam beberapa situasi (melibatkan loop).

b. Pengacakan Graph

Sementara definisi beberapa metrik Graph mungkin mudah dihitung, tidak mudah untuk memahami kepentingan relatif mereka. Kami menggunakan Pengacakan Jaringan / Graph dalam kasus seperti itu. Kami menghitung metrik untuk Graph di tangan dan untuk Graph serupa lainnya yang dihasilkan secara acak. Kesamaan ini misalnya bisa menjadi jumlah kepadatan dan node yang sama. Biasanya kita menghasilkan 1000 graph acak serupa dan menghitung metrik Graph untuk masing-masing dan kemudian membandingkannya dengan metrik yang sama untuk Graph di

tangan untuk sampai pada beberapa gagasan patokan. Dalam Ilmu Data ketika mencoba untuk membuat klaim tentang Graph itu membantu jika kontras dengan beberapa Graph yang dihasilkan secara acak.

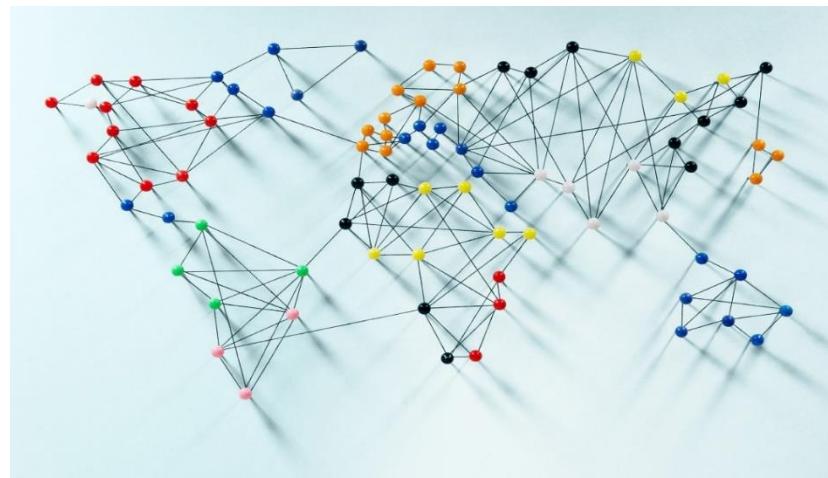
3. Manfaat Graph

Berikut adalah beberapa poin yang membantu memotivasi untuk menggunakan graph dalam masalah ilmu data sehari-hari

- a. Graph menyediakan cara yang lebih baik untuk berurusan dengan konsep abstrak seperti hubungan dan interaksi. Mereka juga menawarkan cara berpikir visual yang intuitif tentang konsep-konsep ini. Graph juga membentuk dasar alami untuk menganalisis hubungan dalam konteks sosial.
- b. Database Graph telah menjadi alat komputasi umum dan alternatif untuk database SQL dan NoSQL.
- c. Graph digunakan untuk memodelkan alur kerja analitik dalam bentuk DAGs (Graph asiklik terarah)
- d. Beberapa Neural Network Frameworks juga menggunakan DAGs untuk memodelkan berbagai operasi di lapisan yang berbeda.
- e. Konsep Teori Graph digunakan untuk mempelajari dan memodelkan Jejaring Sosial, pola Penipuan, pola konsumsi daya, Viralitas dan Pengaruh di Media Sosial. Social Network Analysis (SNA) mungkin adalah aplikasi teori graph untuk ilmu data yang paling terkenal.
- f. Hal ini digunakan dalam algoritma Clustering - Khususnya K-Means
- g. Dinamika Sistem juga menggunakan beberapa konsep Teori Graph – Khususnya loop
- h. Path Optimization adalah bagian dari masalah Optimasi yang juga menggunakan konsep Graph.
- i. Dari perspektif Ilmu Komputer – Graph menawarkan efisiensi komputasi. Kompleksitas Big O untuk beberapa algoritma lebih baik untuk data yang diatur dalam bentuk Graph (dibandingkan dengan data tabular)

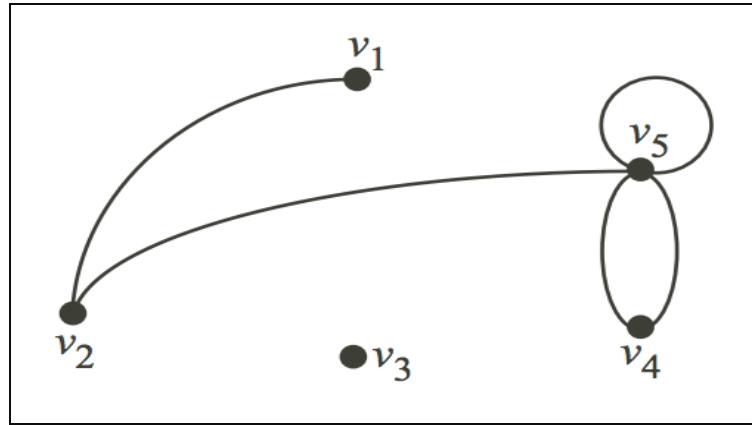
4. Graph dan Analisis Jaringan

"Gambar berbicara seribu kata" adalah salah satu frasa yang paling umum digunakan. Tapi graph berbicara jauh lebih dari itu. Representasi visual data, dalam bentuk graph, membantu kita mendapatkan wawasan yang dapat ditindaklanjuti dan membuat keputusan berbasis data yang lebih baik berdasarkan mereka. Tetapi untuk benar-benar memahami graph apa dan mengapa mereka digunakan, kita perlu memahami konsep yang dikenal sebagai Teori Graph. Memahami konsep ini membuat programmer yang lebih baik



Gambar 17. 2 Ilustrasi Graph

Tetapi jika telah mencoba memahami konsep ini sebelumnya, dimana akan menemukan banyak konsep formula dan teori. Itulah sebabnya menjelaskan konsep dan kemudian memberikan ilustrasi sehingga dapat mengikuti dan secara intuitif memahami bagaimana kinerja fungsinya. Memberikan penjelasan yang tepat tentang konsep ini adalah pilihan yang jauh lebih disukai daripada definisi ringkas. Pada artikel ini akan melihat graph apa, aplikasi dan sejarah tentang graph. Mari lihat grafik sederhana untuk memahami konsepnya. Lihat gambar di bawah ini .

**Gambar 17. 3** Grafh V

Pertimbangkan bahwa grafik ini mewakili tempat-tempat di kota yang umumnya dikunjungi orang, dan jalan yang diikuti oleh pengunjung kota itu. Mari kita anggap V sebagai tempat dan E sebagai jalan untuk melakukan perjalanan dari satu tempat ke tempat lain.

$$V = \{v_1, v_2, v_3, v_4, v_5\}$$

$$E = \{(v_1, v_2), (v_2, v_5), (v_5, v_5), (v_4, v_5), (v_4, v_4)\}$$

Tepi (u, v) sama dengan tepi (v, u) – Mereka adalah pasangan yang tidak teratur.

Grafik adalah struktur matematika yang digunakan untuk mempelajari hubungan berpasangan antara objek dan entitas. Ini adalah cabang matematika diskrit dan telah menemukan beberapa aplikasi dalam Ilmu Komputer, Kimia, Linguistik, Penelitian Operasi, Sosiologi dll.

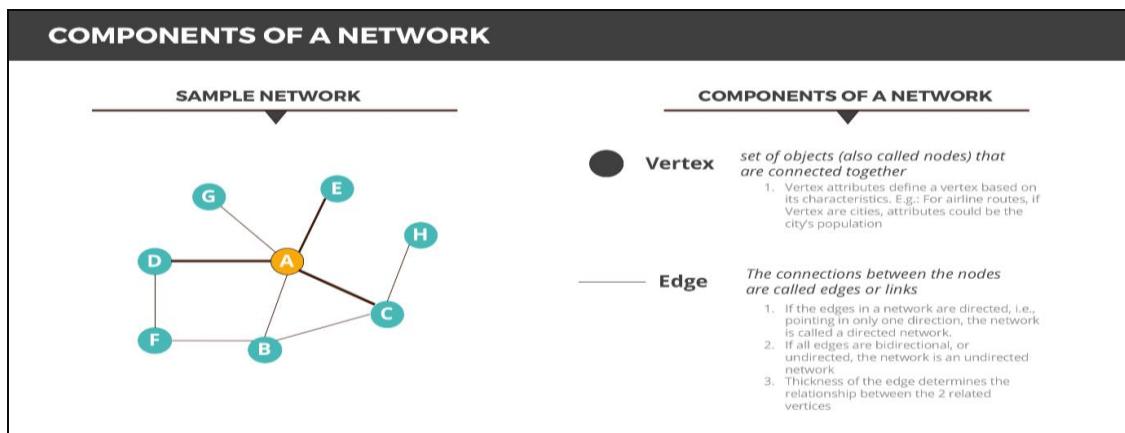
Bidang Ilmu Data dan Analytics juga telah menggunakan Grafik untuk memodelkan berbagai struktur dan masalah. Sebagai Ilmuwan Data, Anda harus dapat memecahkan masalah dengan cara yang efisien dan Grafik menyediakan mekanisme untuk melakukan itu dalam kasus-kasus di mana data diatur dengan cara tertentu. Secara resmi graph sebagai berikut.

- Grafik adalah sepasang set. V adalah satu set simpul. E adalah satu set tepi. E terdiri dari pasangan elemen dari V (unordered pair) $G = (V, E)$

- b. DiGraph juga merupakan sepasang set. V adalah satu set simpul. A adalah satu set busur. A terdiri dari pasangan elemen dari V (pasangan yang dipesan) $D = (V, A)$

Dalam kasus digraf, ada perbedaan antara '(u, v)' dan '(v, u)'. Biasanya tepi disebut busur dalam kasus seperti itu untuk menunjukkan gagasan arah. Ada paket yang ada di R dan Python untuk menganalisis data menggunakan konsep teori Grafik. Pada artikel ini kita akan melihat secara singkat beberapa konsep dan menganalisis dataset menggunakan paket Networkx Python.

```
from IPython.display import Image
Image('images/network.PNG')
```



Gambar 17. 4 Ilustrasi Komponen Network

EXAMPLES OF NETWORKS AND THEIR COMPONENTS				
NETWORK	VERTICES	VERTEX ATTRIBUTES	EDGES	EDGE ATTRIBUTES
Airlines Network	Airports	Footfall, Terminals, Staff, City population, International/Domestic, Freight, Hangar capacity	Airplanes / Routes	Frequency, # Passengers, Plane Type, Fuel Usage, Distance covered, Empty seats
Banking Network	Account Holders	Name, demographics, KYC Document, Products, Account status, balance and other details	Transactions	Type, Amount, Authentication (pass/OTP), Time, Location, Device
Social Network	Users	Name, demographics, # connections, likes, circles belong to, subscriptions	Interactions	Medium (like/comment/direct message), time, duration, type of content, topic
Physician Network	Doctors	Demographics, speciality, experience, affiliation (type and size), Weekly patient intake	Patients	Demographics, Diagnosis history, visit frequency, purpose, referred to, insurance
Supply Chain Network	Warehouses	Location, size, capacity, storage type, connectivity, manual/automated	Trucks	Load capacity, # wheels, year of make, geographical permit, miles travelled, Maintenance cost, driver experience

Dari contoh-contoh di atas jelas bahwa aplikasi Graph dalam Analisis Data sangat banyak dan luas. Mari kita lihat beberapa kasus penggunaan:

a. Analisis Pemasaran

Graph dapat digunakan untuk mencari tahu orang-orang paling berpengaruh di Jejaring Sosial. Pengiklan dan Pemasar dapat memperkirakan ledakan terbesar untuk uang pemasaran dengan merutekan pesan mereka melalui orang-orang paling berpengaruh di Jejaring Sosial.

b. Transaksi Perbankan

Graph dapat digunakan untuk menemukan pola yang tidak biasa membantu dalam mengurangi transaksi penipuan. Ada contoh di mana aktivitas teroris telah terdeteksi dengan menganalisis aliran uang di seluruh jaringan Perbankan yang saling berhubungan.

c. Rantai Pasokan

Graph membantu dalam mengidentifikasi rute optimal untuk truk pengiriman Anda dan dalam mengidentifikasi lokasi untuk gudang dan pusat pengiriman

d. Farmasi

Perusahaan farmasi dapat mengoptimalkan rute salesman menggunakan teori Graph. Ini membantu dalam memotong biaya dan mengurangi waktu perjalanan bagi salesman.

e. Telekomunikasi

Perusahaan telekomunikasi biasanya menggunakan Graph (diagram Voronoi) untuk memahami jumlah dan lokasi menara Cell untuk memastikan cakupan maksimum.

5. Graph dalam Python

Penggunaan paket di Python dapat dipasang di lingkungan Root Anaconda (jika Anda menggunakan distribusi Anaconda Python), dapat juga bisa.`networkx``pip install`. Mari lihat beberapa hal umum yang bisa dilakukan dengan paket Networkx. Dibawah ini mengimpor dan membuat Graph dan cara untuk memvisualisasikannya. Pembuatan Graph dibawah ini sebagai contoh.

```
import networkx as nx

# Creating a Graph
G = nx.Graph() # Right now G is empty

# Add a node
G.add_node(1)
G.add_nodes_from([2,3]) # You can also add a list of nodes by passing a list argument

# Add edges
G.add_edge(1,2)

e = (2,3)
G.add_edge(*e) # * unpacks the tuple
G.add_edges_from([(1,2), (1,3)]) # Just like nodes we can add edges from a list
```

Atribut Node dan Edge dapat ditambahkan bersama dengan pembuatan Node dan Edges dengan melewati tuple yang berisi node dan dict atribut.

Selain membangun graph node-by-node atau edge-by-edge, mereka juga dapat dihasilkan dengan menerapkan operasi graph klasik, seperti:

```
subgraph(G, nbunch)      - induced subgraph view of G on nodes in nbunch
union(G1,G2)            - graph union
disjoint_union(G1,G2)   - graph union assuming all nodes are different
cartesian_product(G1,G2) - return Cartesian product graph
compose(G1,G2)          - combine graphs identifying nodes common to both
complement(G)           - graph complement
create_empty_copy(G)    - return an empty copy of the same graph class
```

```
convert_to_undirected(G) - return an undirected representation of G
```

```
convert_to_directed(G) - return a directed representation of G
```

Kelas terpisah ada untuk berbagai jenis Graph. Misalnya kelas memungkinkan untuk membuat Graph Terarah. Graph spesifik yang berisi jalur dapat dibuat langsung menggunakan satu metode.

```
Image('images/graphclasses.PNG', width = 400)
```

Mengakses tepi dan node

Node dan Edges dapat diakses bersama menggunakan dan metode. Node dan tepi individu dapat diakses menggunakan notasi braket /
subskrip. `G.nodes()` `G.edges()`

```
G.nodes()
```

```
NodeView((1, 2, 3))
```

```
G.edges()
```

```
EdgeView([(1, 2), (1, 3), (2, 3)])
```

```
G[1] # same as G.adj[1]
```

```
AtlasView({2: {}, 3: {}})
```

```
G[1][2]
```

```
{}
```

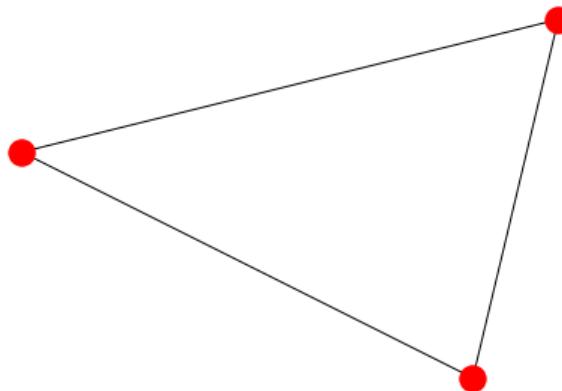
```
G.edges[1, 2]
```

```
{}
```

6. Visualisasi Graph

Networkx menyediakan fungsi dasar untuk memvisualisasikan graph, tetapi tujuan utamanya adalah untuk memungkinkan analisis graph daripada melakukan visualisasi graph. Visualisasi graph sulit dan kita harus menggunakan alat khusus yang didedikasikan untuk tugas ini. Antarmuka Python dalam bentuk (tautan ke dokumentasi di bawah). MatplotlibGraphVizPyGraphViz seperti dibawah ini.

```
%matplotlib inline  
import matplotlib.pyplot as plt  
nx.draw(G)
```



Gambar 17. 5 Graph Segitiga

Terlebih dahulu menginstal Graphviz, Dalam opsi instalasi harus menyediakan jalur ke Graphviz dan folder.pip install pygraphviz --install-option="--withinclude"

```
import pygraphviz as pgv  
  
d={'1': {'2': None}, '2': {'1': None, '3': None}, '3': {'1': None}}  
  
A = pgv.AGraph(data=d)  
  
print(A) # This is the 'string' or simple representation of the Graph
```

Output:

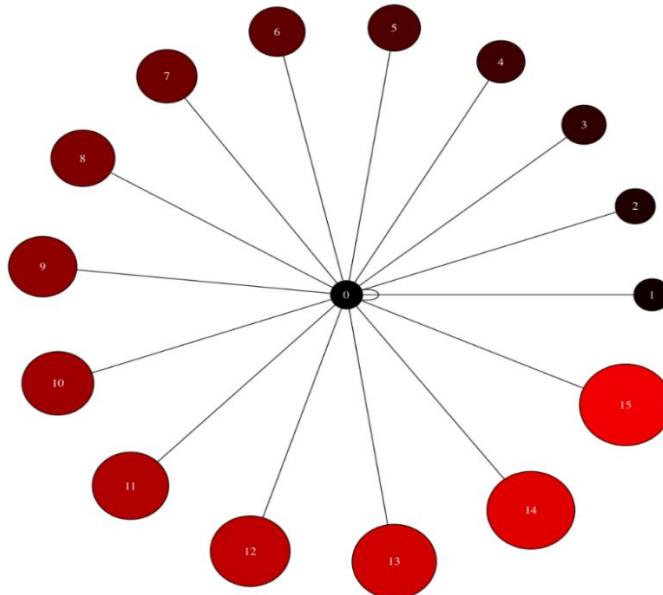
```
strict graph "" {  
    1 -- 2;  
    2 -- 3;  
    3 -- 1;  
}
```

PyGraphviz memberikan kontrol besar atas atribut individu dari tepi dan node.

Kita bisa mendapatkan visualisasi yang sangat indah menggunakannya.

```
# Let us create another Graph where we can individually control the colour of each  
node  
  
B = pgv.AGraph()  
  
  
# Setting node attributes that are common for all nodes  
  
B.node_attr['style']='filled'  
  
B.node_attr['shape']='circle'  
  
B.node_attr['fixedsize']=true'  
  
B.node_attr['fontcolor']='#FFFFFF'  
  
  
# Creating and setting node attributes that vary for each node (using a for loop)  
  
for i in range(16):  
  
    B.add_edge(0,i)  
  
    n=B.get_node(i)  
  
    n.attr['fillcolor']="#%2x0000"%(i*16)  
  
    n.attr['height']="%"s"%(i/16.0+0.5)  
  
    n.attr['width']="%"s"%(i/16.0+0.5)  
  
B.draw('star.png',prog="circo") # This creates a .png file in the local directory.  
Displayed below.
```

```
Image('images/star.png', width=650) # The Graph visualization we created above.
```



Gambar 17. 6 Visualisasi Graph Titik

Biasanya, visualisasi dianggap sebagai tugas terpisah dari analisis Graph. Graph yang pernah dianalisis diekspor sebagai Dotfile. Dotfile ini kemudian divisualisasikan secara terpisah untuk menggambarkan titik tertentu yang kami coba buat.

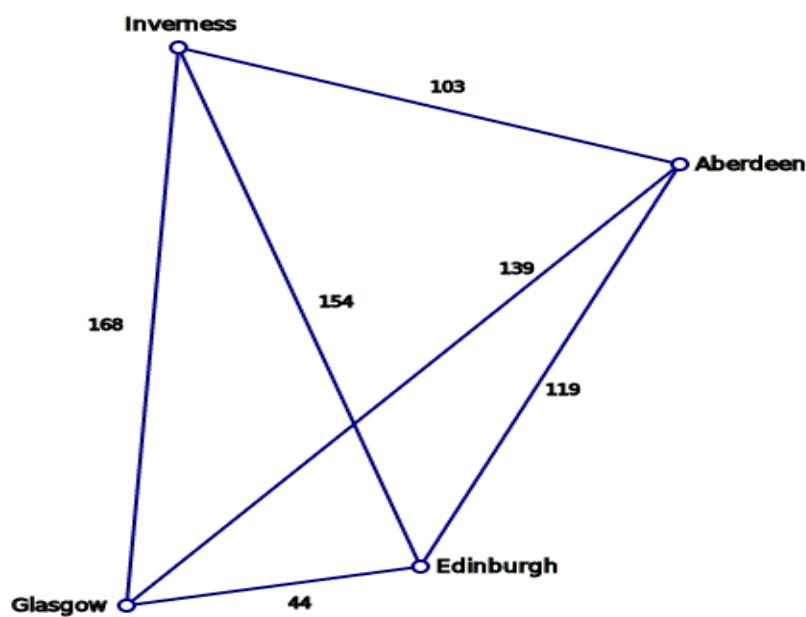
- a. Ada jalan terpendek berdasarkan jarak
- b. Ada jalur terpendek berdasarkan waktu penerbangan

Apa yang bisa dilakukan adalah menghitung algoritma jalur terpendek dengan menimbang jalur dengan jarak atau airtime. Harap dicatat bahwa ini adalah solusi perkiraan. Masalah sebenarnya yang harus dipecahkan adalah menghitung anjuk jalur terpendek dalam ketersediaan penerbangan ketika mencapai bandara transfer + waktu tunggu untuk transfer. Pendekatan yang lebih lengkap adalah bagaimana manusia biasanya merencanakan perjalanan mereka. Untuk tujuan artikel hanya akan berasumsi bahwa penerbangan sudah tersedia ketika mencapai bandara dan menghitung jalur terpendek

menggunakan airtime sebagai berat. Pengetahuan tentang teori dan paket Python akan menambahkan toolset berharga ke gudang data scientist. Untuk dataset yang digunakan di atas, serangkaian pertanyaan lain dapat diajukan seperti:

- a. Temukan jalur terpendek antara dua bandara yang diberi Biaya, Airtime dan Ketersediaan?
- b. Jika Anda Maskapai penerbangan dan Anda memiliki armada pesawat terbang. Anda memiliki gagasan tentang permintaan yang tersedia untuk penerbangan Anda. Mengingat bahwa Anda memiliki izin untuk mengoperasikan 2 pesawat lagi (atau menambahkan 2 pesawat ke armada Anda) rute mana yang akan Anda operasikan untuk memaksimalkan profitabilitas?
- c. Dapatkah Anda mengatur ulang penerbangan dan jadwal untuk mengoptimalkan parameter tertentu (seperti Ketepatan Waktu atau Profitabilitas dll)

Graph yang ditunjukkan di bawah ini mewakili empat kota Skotlandia dan garis dan angka mewakili koneksi dan jarak (dalam mil) di antara mereka.



Gambar 17. 7 Graph Kota Scotlandia

Mungkin masalah yang paling terkenal dalam teori graph adalah Masalah Salesman Perjalanan. Dalam masalah ini, seorang salesman hipotetis perlu mengunjungi sejumlah kota seefisien mungkin. Ada enam kemungkinan kombinasi - yang harus dia pilih untuk meminimalkan jarak atau waktu? Itu masalah untuk posting di masa depan tetapi untuk saat ini kita perlu membuat struktur data untuk menahan graph, dan saya akan memulai dengan menulis kelas untuk melakukan hal itu.

Sebagian besar data dapat dengan mudah dipasang ke dalam semacam struktur baris / kolom, di Python ini mungkin daftar daftar, kamus tuples, beberapa kombinasi lain dari ini, atau mungkin struktur data kustom Anda sendiri. Graph sedikit lebih berantakan tetapi ada dua cara utama untuk mewakili mereka.

Yang pertama disebut daftar adjacency, yang akan saya gunakan di sini. Ini terdiri dari daftar simpul (misalnya kota), yang masing-masing berisi daftar tepi (misalnya daftar kota lain yang terhubung bersama dengan jarak). Graph kota-kota Skotlandia yang akan saya gunakan dalam proyek ini akan terlihat seperti ini ketika diwakili dalam daftar adjacency.

Tabel 17. 1 Daftar Adjacency Kota Scotlandia

Simpul	Tepi					
Inverness	Aberdeen	103	Glasgow	168	Edinburgh	154
Aberdeen	Inverness	103	Glasgow	139	Edinburgh	119
Glasgow	Inverness	168	Aberdeen	139	Edinburgh	44
Edinburgh	Inverness	154	Aberdeen	119	Glasgow	44

Ada beberapa poin yang patut diperhatikan di sini. Pertama setiap tepi ada dua kali, misalnya ada tepi dari Inverness ke Aberdeen dengan berat (jarak) 103, dan juga satu dari Aberdeen ke Inverness dengan berat yang

sama. Tepinya tidak diarahkan tetapi dengan menghilangkan salah satu tepi ini kita dapat mewakili tepi yang diarahkan, misalnya jalan satu arah.

Kedua, dalam contoh ini semua kota terhubung dengan yang lain. Jika kita menambahkan semua kota, kota dan desa di Skotlandia, tidak satupun dari mereka akan terhubung dengan yang lain; Bahkan sebagian besar hanya akan memiliki jalan langsung ke beberapa orang lain. Menggunakan daftar adjacency efisien karena hanya perlu menyimpan tepi yang sebenarnya. Ini membawa ke jenis utama kedua dari struktur data yang digunakan untuk mewakili grafik, matriks adjacency. Ini adalah grid dua dimensi yang terlihat seperti ini:

Tabel 17. 2 Matriks Adjacency Kota Scotlandia

	Inverness	Aberdeen	Glasgow	Edinburgh
Inverness	-	103	168	154
Aberdeen	103	-	139	119
Glasgow	168	139	-	44
Edinburgh	154	119	44	-

Dalam hal ini ada sedikit memori yang terbuang tetapi jika struktur data seperti itu digunakan untuk mewakili semua kota, kota dan desa di Skotlandia, sebagian besar memori akan terbuang sia-sia. Oleh karena itu, daftar adjacency hanya cocok jika sebagian besar atau semua simpul terhubung.

C. LATIHAN SOAL

1. Jelaskan terminologi Graph menurut ahli yang anda ketahui!
2. Jelaskan alasan mengapa menggunakan graph dalam menyelesaikan permasalahan disekitar kita!
3. Jelaskan keterkaitan Graph dengan analisis Jaringan pada ilmu komputer!
4. Buatlah visualisasi Graph dengan sekitar lingkungan anda!

5. Buatlah contoh implementasi graph dengan bahasa pemrograman Python!

D. REFERENSI

- Emi Sita Eriana, A. Z. (2021). *Praktikum Algoritma Dan Pemrograman*. Tangerang Selatan: Unpam Press.
- Eriana, E. S. (2020). Pemilihan Ketua Himpunan Universitas Pamulang Dengan Metode Simple Additive Weighting (Saw). *Jik(Jurnal Ilmu Komputer)*.
- Jodi, U. R. (2020). Algoritma Dan Struktur Data.
- 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 Databases.
- 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.
- 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). Pendekripsi Kantuk Secara Real Time Menggunakan Pustaka OpenCV Dan Dlib Python. *Sainstech : Jurnal Penelitian Dan Pengkajian Sains Dan Teknologi*.

PERTEMUAN 18

ALGORITMA DIJKSTRA

A. TUJUAN PEMBELAJARAN

Setelah mempelajari materi pertemuan ini diharapkan mahasiswa mengerti dan paham algoritma Djikstra, greedy approach, cara kerja algoritma djiktra dan mampu membuat program dengan algoritma djiktra dengan Python

B. URAIAN MATERI

1. Algoritma Dijkstra

Algoritma Dijkstra ditemukan di tahun 1959 oleh Edsger.Wybe Dijkstra. Algoritma memecahkan masalah pencarian jalur terpendek dari setiap simpul dengan nilai non-negatif pada graf. Dijkstra merupakan algoritma yang termasuk serakah, yaitu algoritma yang sering digunakan untuk menyelesaikan masalah yang berhubungan dengan optimasi. Dalam mencari jalur terpendek, algoritma dijkstra bekerja dengan mencari bobot terkecil dari graf berbobot, jarak terpendek akan diperoleh dari dua atau lebih titik pada graf, dan nilai total yang diperoleh adalah nilai minimum. Misalkan G adalah graf berarah berlabel, simpul $V(G) = \{v_1, v_2, \dots, v_n\}$ dan jalur terpendek yang dicari adalah dari v_1 ke v_n . Algoritma Dijkstra dimulai pada titik v_1 . Dalam iterasinya, algoritma akan memulai dari titik 1 untuk mencari titik dengan jumlah bobot paling sedikit. Pisahkan titik-titik yang dipilih dan tidak lagi mempertimbangkan titik-titik ini pada iterasi berikutnya. Langkah untuk menentukan jalur terpendek dalam algoritma Dijkstra adalah:

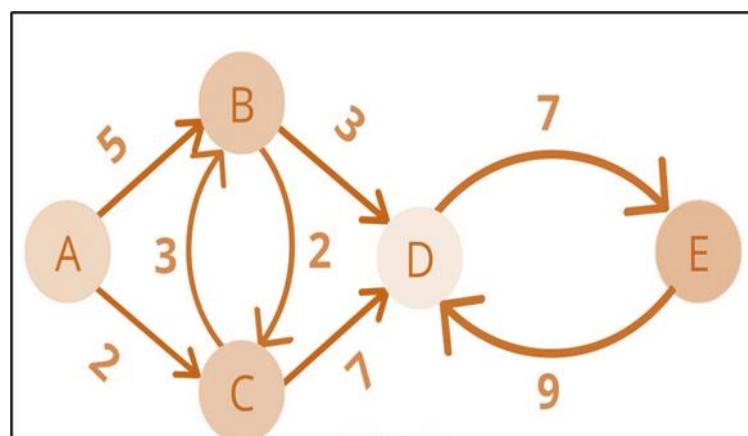
- a. Pertama pilih node sumber sebagai node awal, dan inisialisasi ke 1'.
- b. bentuk tabel yang terdiri dari node, status, bobot, dan pendahulunya. Lengkapi kolom bobot yang diperoleh dari jarak dari node sumber ke semua node yang terhubung langsung dengan node sumber.
- c. Jika node sumber ditemukan, itu ditetapkan sebagai node yang dipilih.
- d. Tetapkan label permanen ke node yang dipilih dan perbarui node yang terhubung langsung.

- e. Menentukan node sementara yang terhubung dengan node yang dipilih sebelumnya dan memiliki bobot terkecil dilihat dari tabel, dan menentukannya sebagai node terpilih berikutnya.
- f. Apakah node yang dipilih adalah node target? Jika ya, node atau set pendahulu yang dipilih adalah rangkaian yang menunjukkan jalur terpendek.

Secara matematis, graf didefinisikan sebagai berikut : Graf G didefinisikan sebagai pasangan himpunan (V,E) yang dalam hal ini : V = himpunan tidak kosong dari simpul - simpul (vertices atau node): $\{v_1, v_2, \dots, v_n\}$ E = himpunan sisi (edges atau arcs) yang menghubungkan sepasang simpul: $\{e_1, e_2, \dots, e_n\}$ atau dapat ditulis singkat notasi $G = (V,E)$.

2. Algoritma Dijkstra dalam Python

Setiap kali perlu untuk mewakili dan menyimpan koneksi atau link antara elemen, di gunakan struktur data yang dikenal sebagai graph. Dalam graph, memiliki node (simpul) dan tepi. Node adalah objek (nilai), dan tepi adalah garis yang menghubungkan node. Dalam python, mewakili node graph sebagai kunci dan koneksinya sebagai nilai. Diperlukan untuk menemukan jarak terpendek antara node ini, dan umumnya menggunakan Algoritma Dijkstra dalam python. Sebuah graph pada umumnya terlihat seperti ini



Masalah jalur terpendek sumber tunggal adalah tentang menemukan jalur antara simpul tertentu (disebut sumber) ke semua simpul lain (disebut tujuan) dalam graph sehingga jarak total di antara mereka adalah minimum. Algoritma Dijkstra adalah algoritma untuk menemukan jalur terpendek antara dua node

dari graph yang diberikan. Saat melintasi jalur terpendek antara dua node, tidak perlu setiap node akan dikunjungi.

Algoritma Dijkstra dapat digunakan untuk memecahkan masalah SSSP untuk graph tertimbang. Dalam contoh di atas, jalur terpendek antara simpul V5 dan V3 berbobot numerik 8 (V5 -> V4 -> V3). Algoritma Dijkstra menemukan penggunaan dalam berbagai aplikasi kehidupan nyata:

- a. Layanan Pemetaan Digital
- b. Aplikasi Jejaring Sosial
- c. Jaringan Telepon
- d. Ip Routing untuk menemukan jalur terpendek

Untuk mengimplementasikan algoritma Dijkstra dalam python, membuat metode yang mengambil dua parameter - graph yang sedang diobservasi dan node awal yang akan menjadi titik sumber untuk algoritma dijkstra. Algoritma Dijkstra didasarkan pada langkah-langkah berikut:

- a. Menerima graph tertimbang dan node awal.
- b. Mulailah dengan node awal. Periksa node yang berdekatan.
- c. Temukan node dengan nilai tepi minimum.
- d. Ulangi proses ini sampai simpul tujuan dikunjungi.
- e. Pada akhir fungsi, kita mengembalikan berat jalur terpendek untuk setiap node dan jalur juga.

Kompleksitas waktu untuk algoritma Dijkstra adalah $O(V^2)$ di mana "V" adalah jumlah simpul graph. Hal ini disebabkan oleh fakta bahwa kita telah menggunakan dua loop bersarang bersama-sama dan masing-masing iterates atas semua node. Kompleksitas ruang adalah $O(E)$ di mana "E" adalah jumlah tepi graph karena dapat menambahkan dengan tepi.path.

Sumber simpul dalam graph, temukan jalur terpendek dari sumber ke semua simpul dalam graph yang diberikan. Algoritma dijkstra sangat mirip dengan algoritma prim untuk pohon spanning minimum. Seperti prim's mst, menghasilkan spt (pohon jalur terpendek) dengan sumber yang diberikan sebagai akar, dimana mempertahankan dua set, satu set berisi simpul yang

termasuk dalam pohon jalur terpendek, satu set lainnya termasuk simpul yang belum termasuk dalam pohon jalur terpendek.

Disetiap langkah algoritma, ditemukan simpul yang ada di set lain (set belum termasuk) dan memiliki jarak minimum dari sumbernya. Di bawah ini adalah langkah-langkah terperinci yang digunakan dalam algoritma dijkstra untuk menemukan jalur terpendek dari satu sumber vertex ke semua simpul lain dalam graph yang diberikan. Algoritma sebagai berikut.

- a. Buat sptSet set (set pohon jalur terpendek) yang melacak simpul yang termasuk dalam pohon jalur terpendek, yaitu, yang jarak minimumnya dari sumber dihitung dan diselesaikan. Awalnya, set ini kosong.
- b. Tetapkan nilai jarak ke semua simpul dalam graph input. Menginisialisasi semua nilai jarak sebagai INFINITE. Tetapkan nilai jarak sebagai 0 untuk sumber vertex sehingga dipilih terlebih dahulu.
- c. Sementara sptSet tidak termasuk semua simpul:
 - 1) Pilih vertex u yang tidak ada di sptSet dan memiliki nilai jarak minimum.
 - 2) Sertakan Anda ke sptSet.
 - 3) Perbarui nilai jarak dari semua simpul yang berdekatan dari Anda. Untuk memperbarui nilai jarak, iterasi melalui semua simpul yang berdekatan. Untuk setiap vertex v yang berdekatan, jika jumlah nilai jarak Anda (dari sumber) dan berat tepi u-v, kurang dari nilai jarak v, maka perbarui nilai jarak v.

Algoritma ini digunakan untuk menemukan rute terpendek atau jalur antara dua node dalam graph tertentu. Menggunakan:

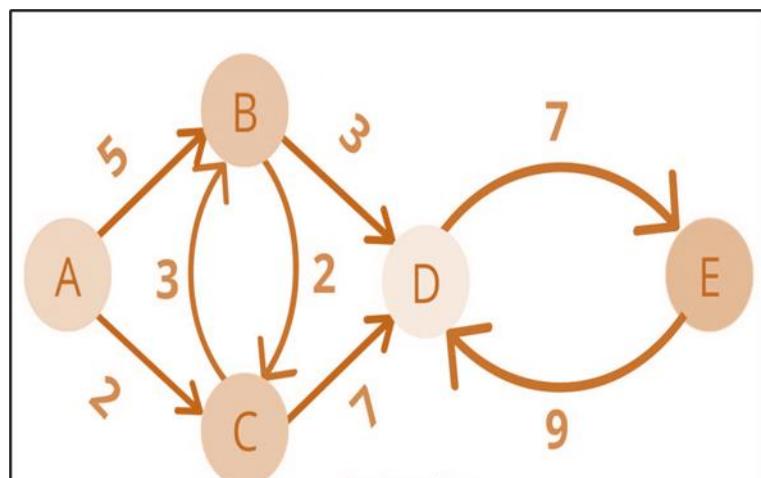
- a. Penggunaan utama dari algoritma ini adalah bahwa graph memperbaiki node sumber dan menemukan jalur terpendek ke semua node lain yang ada dalam graph yang menghasilkan pohon jalur terpendek.
- b. Hal ini juga dapat digunakan untuk menemukan jarak antara node sumber ke node tujuan dengan menghentikan algoritma setelah rute terpendek diidentifikasi.

Jadi, Algoritma Dijkstra digunakan untuk menemukan jarak terpendek antara node sumber dan node target. Pendekatan yang mengikuti Algoritma Dijkstra dikenal sebagai Greedy Approach. Meskipun titik diskusi hari ini adalah memahami logika dan implementasi Algoritma Dijkstra dalam python, jika tidak terbiasa dengan istilah-istilah seperti Greedy Approach and Graphs.

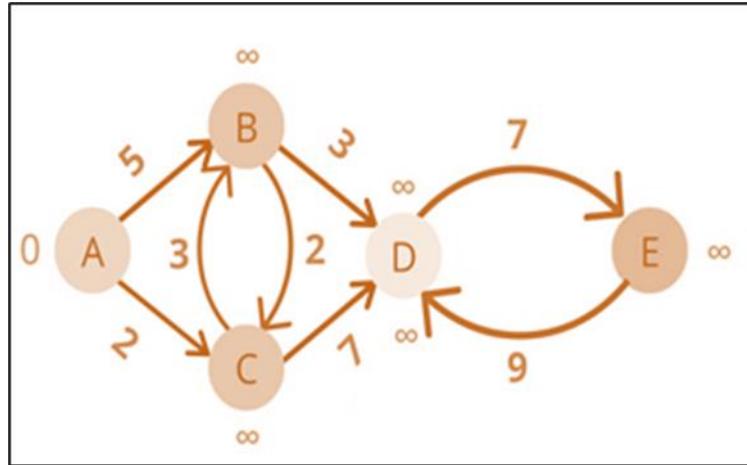
3. Greedy Approach

Dalam istilah Laymen, pendekatan Greedy adalah strategi di mana memilih pilihan terbaik yang tersedia, dengan asumsi bahwa itu akan membawa kami ke solusi terbaik. Pikirkan tentang hal ini dengan cara ini, kami memilih solusi terbaik pada saat itu tanpa banyak memikirkan konsekuensi di masa depan. Deskripsi Algoritma Dijkstra

Langkah 1: Buat graph sementara yang menyimpan nilai graph asli dan beri nama sebagai graph yang tidak terlihat. Juga, menginisialisasi daftar yang disebut jalur untuk menyimpan jalur terpendek antara sumber dan target.



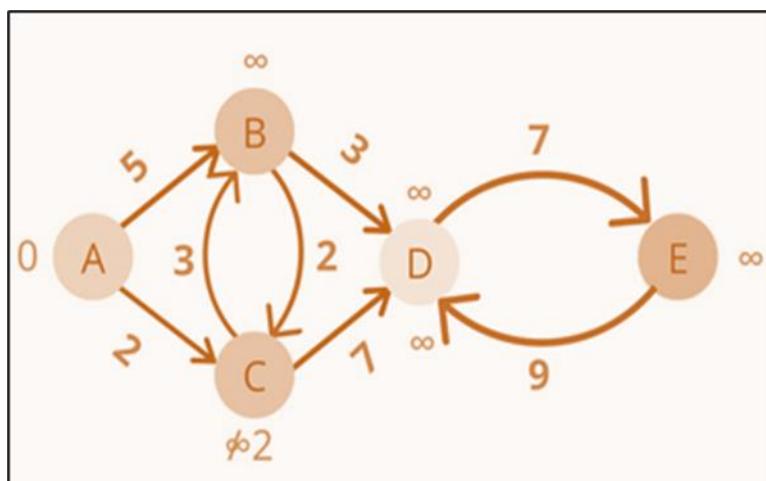
Langkah 2: Diperlukan untuk menghitung Jarak Minimum dari node sumber ke setiap node. Awalnya, tandai total_distance untuk setiap node sebagai infinity (∞) dan tandai node sumber total_distance sebagai 0, karena jarak dari node sumber ke node sumber adalah 0. Juga, tandai node sumber ini sebagai current_node.



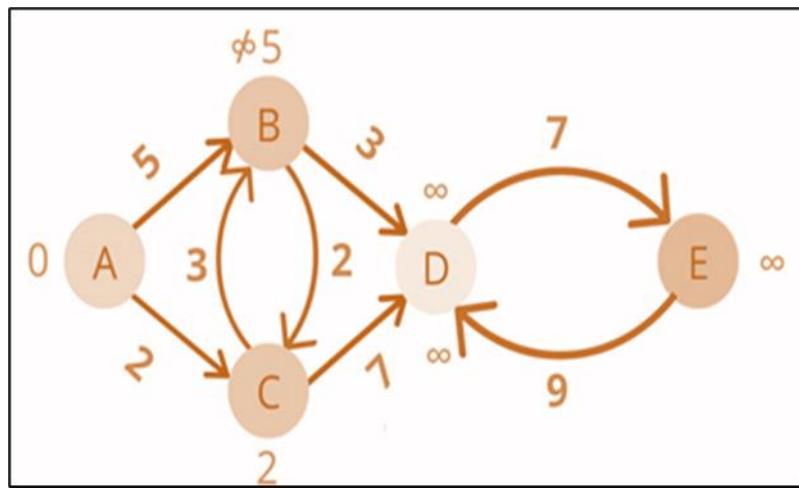
Node sumber / current_node adalah 'A', tandai sebagai 0.

Langkah 3: Dari current_node, pilih node tetangga (node yang terhubung langsung) dalam urutan acak apa pun. Periksa apakah nilai node saat ini (awalnya akan (∞)) lebih tinggi dari (nilai nilai current_node + dari tepi yang menghubungkan node tetangga ini dengan current_node).

Jika iya, maka ganti pentingnya node tetangga ini dengan nilai current_node + nilai tepi yang menghubungkan node tetangga ini dengan current_node. Ulangi proses ini untuk semua node tetangga dari node saat ini

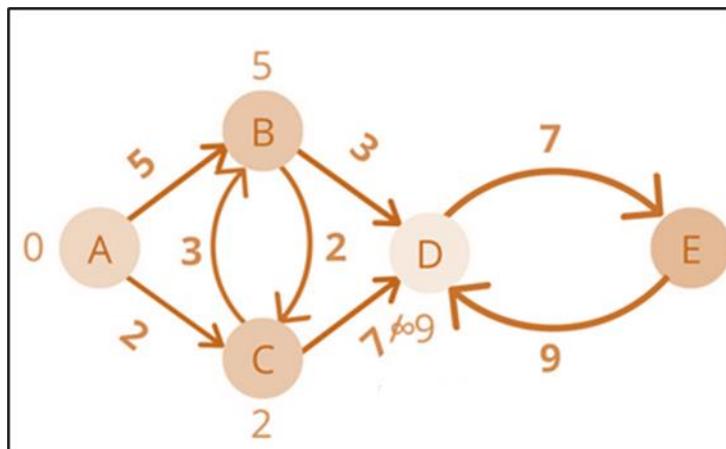


Pada C, sebagai $2 < \infty$, perbarui nilai C dari ∞ menjadi 2.



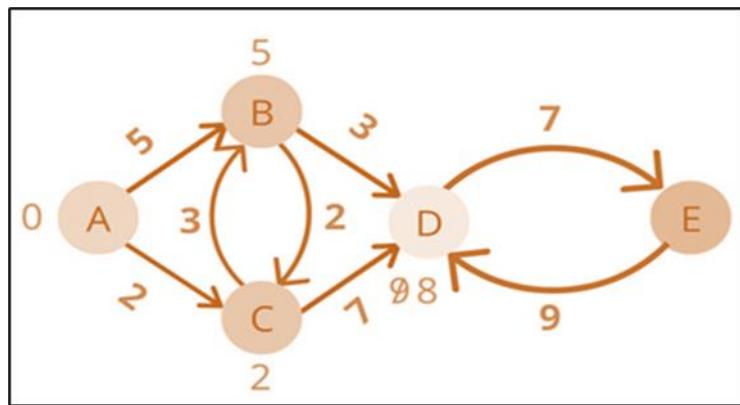
Di B, sebagai $5 < \infty$, perbarui nilai B dari ∞ menjadi 5.

Langkah 4: Setelah memperbarui semua node tetangga dari nilai node saat ini, saatnya untuk menghapus node saat ini dari unvisited_nodes. Dari semua node yang merupakan tetangga dari node saat ini, tetangga memilih tetangga dengan minimum_distance dan mengaturnya sebagai current_node.

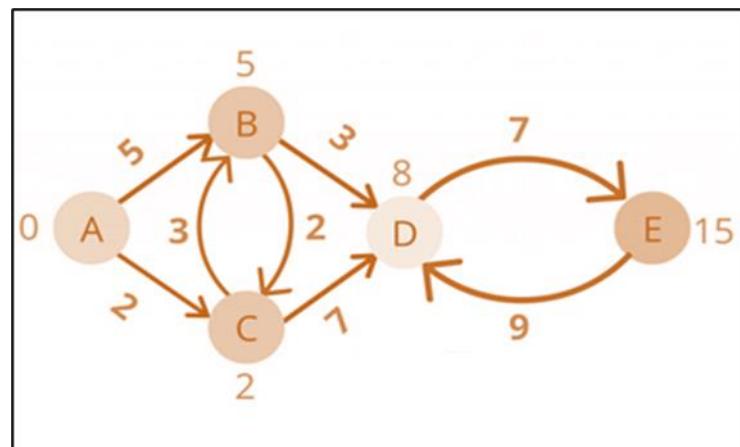


Di D (jalurnya adalah A->C->D), 9 ($7 + 2$) kurang dari ∞ , perbarui nilai dari ∞ ke 9.

Langkah 5: Ulangi langkah 3 dan 4 sampai dan kecuali semua node di node unvisited_visited telah dikunjungi. Setelah semua node telah dikunjungi, akan mendapatkan jarak terpendek dari node sumber ke node target.



Ketika kembali tiba di D (kali ini jalurnya adalah A->B ->D), nilai D menjadi 8.



Akhirnya, di E, $15 < \infty$, Jadi jarak akan diperbarui dari ∞ menjadi 15.

Output yang di dapatkan di sini adalah 15.

4. Implementasi Algoritma Dijkstra di Python

- Pertama, buat graph.

```
def initial_graph() :
    return {
        'A': {'B':1, 'C':4, 'D':2},
        'B': {'A':9, 'E':5},
        'C': {'A':4, 'F':15},
        'D': {'A':10, 'F':7},
        'E': {'B':3, 'J':7},
        'F': {'C':11, 'D':14, 'K':3, 'G':9},
        'G': {'F':12, 'I':4},
        'H': {'J':13},
        'I': {'G':6, 'J':7},
        'J': {'H':2, 'I':4},
        'K': {'F':6}
    }
```

Sekarang, menginisialisasi node sumber.

- Tetapkan variabel yang disebut jalur untuk menemukan jarak terpendek antara semua node.
- Tetapkan variabel yang disebut adj_node untuk menjelajahi node yang berdekatan atau berdekatan.
- Tetapkan variabel yang disebut antrian untuk menambahkan node yang tidak divisuangi dan untuk menghapus node yang dikunjungi.
- Tetapkan variabel yang disebut graph untuk mengimplementasikan graph yang dibuat.

```
initial = 'A' #2
path = {} #3
adj_node = {} #4
queue = [] #5
graph = initial_graph() #6
```

Buat loop yang disebut node sehingga setiap node dalam graph dikunjungi. Juga, menginisialisasi jalan ke nol.

```

for node in graph:
    path[node] = float("inf")
    adj_node[node] = None
    queue.append(node)

path[initial] = 0

```

Sekarang, buat loop sementara di dalam antrian untuk menghapus node yang dikunjungi dan juga untuk menemukan jarak minimum antara node.

```

while queue:

    key_min = queue[0]
    min_val = path[key_min]
    for n in range(1, len(queue)):
        if path[queue[n]] < min_val:
            key_min = queue[n]
            min_val = path[key_min]
    cur = key_min
    queue.remove(cur)

    for i in graph[cur]:
        alternate = graph[cur][i] + path[cur]
        if path[i] > alternate:
            path[i] = alternate
            adj_node[i] = cur

```

Akhirnya, tetapkan variabel x untuk node tujuan untuk menemukan jarak minimum antara node sumber dan node tujuan.

```

x = 'H'
print('The path between A to H')
print(x, end = '<-')
while True:
    x = adj_node[x]
    if x is None:
        print("")
        break
    print(x, end='<-')

```

Kode Python untuk mengetahui jalur terpendek menggunakan Algoritma Dijkstra. Jadi Jika digabung untuk source pemograman python diatas sebagai berikut.

```

def initial_graph() :
    return {
        'A': {'B':1, 'C':4, 'D':2},
        'B': {'A':9, 'E':5},
        'C': {'A':4, 'F':15},
        'D': {'A':10, 'F':7},
        'E': {'B':3, 'J':7},
        'F': {'C':11, 'D':14, 'K':3, 'G':9},
        'G': {'F':12, 'I':4},
        'H': {'J':13},
        'I': {'G':6, 'J':7},
        'J': {'H':2, 'I':4},
        'K': {'F':6}
    }
print(initial_graph())

initial = 'A'
path = {}
adj_node = {}
queue = []
graph = initial_graph()

for node in graph:
    path[node] = float("inf")
    adj_node[node] = None
    queue.append(node)

path[initial] = 0

while queue:
    # find min distance which wasn't marked as current
    key_min = queue[0]
    min_val = path[key_min]
    for n in range(1, len(queue)):
        if path[queue[n]] < min_val:
            key_min = queue[n]
            min_val = path[key_min]
    cur = key_min
    queue.remove(cur)
    print(cur)

    for i in graph[cur]:
        alternate = graph[cur][i] + path[cur]
        if path[i] > alternate:
            path[i] = alternate
            adj_node[i] = cur

x = 'H'
print ('The path between A to H')
print (x, end='--')
while True:
    x = adj_node[x]
    if x is None:
        print("")
        break
    print(x, end='--')

```

Contoh pemrograman Python dalam algoritma djiktra sebagai berikut

```

1 def dijkstra(nodes, distances):
2     # Ini semua node yang belum dikunjungi
3     unvisited = {node: None for node in nodes}
4     # It will store the shortest distance from one node to another
5     visited = {}
6     current = 'B'
7     # Ini akan menyimpan pendahulu dari node
8     currentDistance = 0
9     unvisited[current] = currentDistance
10    # Menjalankan loop sementara semua node telah dikunjungi
11    while True:
12        # iterasi melalui semua node yang tidak divisited
13        for neighbour, distance in distances[current].items():
14            # Iterasi melalui node yang terhubung dari current_node (untuk
15            # misalnya, a terhubung dengan b dan c memiliki nilai 10 dan 3
16            # masing-masing) dan berat tepi
17            if neighbour not in unvisited: continue
18            newDistance = currentDistance + distance
19            if unvisited[neighbour] is None or unvisited[neighbour] > newDistance:
20                unvisited[neighbour] = newDistance
21            # Sampai sekarang jarak terpendek antara node sumber dan node target
22            # telah ditemukan. Mengatur node saat ini sebagai node target
23
24        visited[current] = currentDistance
25        del unvisited[current]
26        if not unvisited: break
27        candidates = [node for node in unvisited.items() if node[1]]
28        current, currentDistance = sorted(candidates, key = lambda x: x[1])[0]
29    return visited
30
31 nodes = ('A', 'B', 'C', 'D', 'E', 'F', 'G')
32 distances = {
33     'B': {'A': 5, 'D': 1, 'G': 2},
34     'A': {'B': 5, 'D': 3, 'E': 12, 'F': 5},
35     'D': {'B': 1, 'G': 1, 'E': 1, 'A': 3},
36     'G': {'B': 2, 'D': 1, 'C': 2},
37     'C': {'G': 2, 'E': 1, 'F': 16},
38     'E': {'A': 12, 'D': 1, 'C': 1, 'F': 2},
39     'F': {'A': 5, 'E': 2, 'C': 16}
40
41 print(dijkstra(nodes, distances))

```

Jika dirun contoh diatas akan menghasilkan sebagai berikut.

{'A': 4, 'C': 3, 'B': 0, 'E': 2, 'D': 1, 'G': 2, 'F': 4}

Disimpulkan bahwa algoritma Dijkstra dalam python datang dengan sangat mudah ketika kita ingin menemukan jarak terpendek antara sumber dan target. Ini dapat bekerja untuk graph terarah dan tidak diarahkan. Keterbatasan algoritma ini adalah bahwa hal itu mungkin atau mungkin tidak memberikan hasil yang benar untuk angka negatif. Di Google Maps, untuk menemukan rute terpendek antara satu sumber ke sumber lain, menggunakan Algoritma Dijkstra. Aplikasi lain adalah dalam jaringan, di mana dapat membantu dalam mengirim paket dari sumber ke tujuan.

C. LATIHAN SOAL

1. Jelaskan yang anda ketahui mengenai algoritma djiktra
2. Gambarkan secara virtual bagaimana algoritma djiktra berkerja!
3. Jelaskan manfaat memilih algoritma djiktra dalam suatu studi kasus!
4. Jelaskan bagaimana cara kerja dalam pencarian binary!
5. Buatlah contoh algoritma djiktra dengan bahasa python!

D. REFERENSI

- Amanulhaq, A. A. (2021). Implementasi Algoritma Image Hashing Dan Hamming Distance Untuk Deteksi Kemiripan Gambar. *Repository Its*.
- Basant Agarwal, B. B. (2018). Hand-On Data Structures And Algorithms With Python. London: Packt Publishing.
- Emi Sita Eriana, A. Z. (2021). Praktikum Algoritma Dan Pemrograman. Tangerang Selatan: Unpam Press.
- Jodi, U. R. (2020). Algoritma Dan Struktur Data.
- 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.
- 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). Pendekripsi Kantuk Secara Real Time Menggunakan Pustaka OpenCV Dan Dlib Python. Sainstech : Jurnal Penelitian Dan Pengkajian Sains Dan Teknologi.

DAFTAR PUSTAKA

- Amanulhaq, A. A. (2021). Implementasi Algoritma Image Hashing Dan Hamming Distance Untuk Deteksi Kemiripan Gambar. *Repository Its*.
- Emi Sita Eriana, A. Z. (2021). Penerapan Metode Personal Extreme Programming Dalam Perancangan Aplikasi Pemilihan Ketua Hmsi Dengan Weighted Product. *Jurnal Ilmu Komputer*, 27-32.
- Emi Sita Eriana, A. Z. (2021). *Praktikum Algoritma Dan Pemrograman*. Tangerang Selatan: Unpam Press.
- Eriana, E. S. (2020). Pemilihan Ketua Himtif Universitas Pamulang Dengan Metode Simple Additive Weighting (Saw). *Jik(Jurnal Ilmu Komputer)*.
- Eriana, E. S. (2021). Analisis Penerapan Metode Waterfall Dan Topsis Dalam Perancangan Sistem Pemeringkatan Kualitas Dosen Mengajar. *Joaiia : Journal Of Artificial Intelligence And Innovative Applications*.
- Eriana, E. S. (2021). Model–V Pada Perancangan Sistem Informasi Kepegawaian Berbasis Web. *Jurnal Esit (E-Bisnis, Sistem Informasi, Teknologi Informasi)*.
- Jodi, U. R. (2020). *Algoritma Dan Struktur Data*.
- 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). Pendekripsi Kantuk Secara Real Time Menggunakan Pustaka OpenCV Dan Dlib Python. *Sainstech : Jurnal Penelitian Dan Pengkajian Sains Dan Teknologi*.
- Zein, A. (2019). Pendekripsi Penyakit Malaria Menggunakan Medical Images Analisis Dengan Deep Learning Python. *Jurnal Penelitian Dan Pengkajian Sains Dan Teknologi*.
- Zein, A. (2020). Pendekripsi Virus Corona Dalam Gambar X-Ray Menggunakan Algoritma Artificial Intelligence Dengan Deep Learning Python. *Jurnal Esit (E-Bisnis, Sistem Informasi, Teknologi Informasi)*.

**RENCANA PEMBELAJARAN SEMESTER
(RPS)**

Program Studi	: Sistem Informasi S-1	Mata Kuliah/Kode	: Algoritma dan Struktur Data/KB1101
Prasyarat	: -	Sks	: 3 Sks
Semester	: 3	Kurikulum	: KKNI
Deskripsi Kuliah	Mata Kuliah	: Mata kuliah ini membahas mengenai Algoritma dan Struktur Data mata kuliah ini merupakan mata kuliah wajib Program Studi S-1 sistem Informasi yang membahas tentang Pengantar Algoritma dan struktur data, tipe data, array, linked List, teknik searching, teknik sorting, stack, queue, binary tree, graph dan algoritma djiktra	Capaian Pembelajaran : Setelah pembelajaran ini, mahasiswa mampu mampu membuat program dengan menggunakan bahasa python dengan benar
Penyusun	Drs. Afrizal Zein, M.Kom. (Ketua) Emi Sita Eriana, S.Kom., M.Kom. (Anggota)		

PERTEMUAN KE-	KEMAMPUAN AKHIR YANG DIHARAPKAN	BAHAN KAJIAN (MATERI AJAR)	METODE PEMBELAJARAN	PENGALAMAN BELAJAR MAHASISWA	KRITERIA PENILAIAN	BOBOT NILAI
(1)	(2)	(3)	(4)	(5)	(6)	(7)
1	Mahasiswa mampu memahami sejarah, konsep Algoritma dan Struktur Data dan mengimplementasi kedalam bahasa pemrograman	Pengantar Algoritma dan Struktur Data	Ceramah, diskusi	Mengungkapkan pendapat dan mengajukan Pertanyaan tentang algoritma dan struktur data	Mahasiswa aktif bertanya materi yang diberikan dan mengungkapkan pendapat mengenai algoritma dan struktur data	5%
2	Mahasiswa mengenal pemrograman python, installasi perangkat lunak dan mampu membuat program dengan Python	Pengenalan Pemrograman Python	Ceramah, diskusi	Mendeklarasikan varabel sesuai dengan tipe data yangdigunakan dalam Python	Mahasiswa mampu merespon, bertanya dan berdiskusi aktif	5%
3	Mahasiswa mengetahui tentang Tipe data, ADT dan implementasi ADT	Tipe Data	Ceramah, diskusi dan penyelesaian soal	Mengkonversikan bilangan desimal ke biner,	Mahasiswa mampu merespon, bertanya dan berdiskusi aktif	5%

	pada Python			melakukan operasi penjumlahan dan pengurangan bilangan biner		
4	Mahasiswa mengerti penggunaan If-Else, Elif, If bersarang dan mampu membuat if dalam pemrograman Python	If -Else	Ceramah, Tanya jawab dan penyelesaian soal	Latihan 1	Mahasiswa mampu merespon secara aktif materi yang diberikan dengan cara bertanya dan berdiskusi.	5%
5	Mahasiswa mengerti dan mampu menggunakan dan membuat program Array dan Linked List dengan Python	Array dan Linked List	Ceramah, Tanya jawab dan penyelesaian soal	Latihan 2	Mahasiswa mampu merespon secara aktif materi yang diberikan dengan cara bertanya dan berdiskusi.	6%
6	Mahasiswa mengerti dan mampu menggunakan fungsi, rekursif dan membuat program dengan Rekursif dengan Python	Fungsi Rekursif	Ceramah, diskusi	Mengungkapkan pendapat dan mengajukan pertanyaan mengenai fungsi rekursif	Mahasiswa mampu merespon secara aktif materi yang diberikan dengan cara bertanya dan berdiskusi.	6%

7	Mahasiswa mengerti dan paham teknik search, teknik binary, pencarian Biner dan mampu membuat pencarian binary dengan Pemrograman python	Teknik Search	Ceramah, diskusi	Mengungkapkan pendapat dan mengajukan pertanyaan tentang teknik search	Mahasiswa mampu merespon secara aktif materi yang diberikan dengan cara bertanya dan berdiskusi.	6%
8	Mahasiswa mengerti dan paham teknik sorting dan bubble sort, Cara kerja dan membuat program bubble sort dengan Python	Teknik Bubble Sort	Ceramah, Tanya jawab dan penyelesaian soal	Latihan 3	Mahasiswa mampu merespon, bertanya dan membuat program bubble sort	6%
9	Mahasiswa mengerti dan paham teknik merge sort, reprezentasi, kompleksitas waktu seleksi logika dan cara kerja merge sort di python dan mampu membuat program	Teknik Merge Sort	Ceramah, Tanya jawab dan penyelesaian soal	Latihan 4	Mahasiswa mampu merespon, bertanya dan membuat program merge sort sort	6%

	Merge sort					
UTS						
10	Mahasiswa mengerti dan paham teknik selection sort, kompleksitas waktu, kelebihan dan kekurangan selectin sort dan mampu membuat program selection sort	Teknik Selection Sort	Ceramah, Tanya jawab dan penyelesaian soal	Latihan 5	Mahasiswa mampu merespon, bertanya dan embuat program selection sort	5%
11	Mahasiswa mengerti dan paham teknik merge sort, logika dan teknik divide dan conquer pada quick sort dan mampu membuat program quick sort dengan python	Teknik Quick Sort	Ceramah, diskusi	Latihan 6	Mahasiswa mampu merespon, bertanya dan embuat program quick sort	5%
12	Mahasiswa mengerti dan paham stack, proses LIFO, dan ampu mambuat program stack dalam	Stack	Ceramah, diskusi	Mengungkapkan pendapatdan mengajukan pertanyaan	Mahasiswa mampu merespon secara aktif materi yang diberikan dengan cara bertanya	5%

	Python			tentang stack	dan berdiskusi.	
13	Mahasiswa mengerti dan paham Queue, proses FIFO, membuat dan menghapus item data antrian dan mampu membuat program queue dalam Python	Queue	Ceramah, Tanya jawab dan penyelesaian soal	Mengungkapkan pendapat dan mengajukan pertanyaan tentang queue	Mahasiswa mampu merespon secara aktif materi yang diberikan dengan cara bertanya dan berdiskusi.	5%
14	Mahasiswa mengerti dan paham antrian dengan array, Operasi antrian, menyisipkan dan menghapus data pada antrian dan mampu membuat program queue dalam Python	Antrian dengan Array	Ceramah, Tanya jawab dan penyelesaian soal	Latihan 7	Mahasiswa mampu merespon, bertanya dan membuat program antrian dengan array	6%
15	Mahasiswa mengerti dan paham <i>binary tree</i> dan <i>search tree</i> , cara penyisipan dan mencari elemen data di <i>search tree</i>	Binary Tree dan Search Tree	Ceramah, diskusi	Mengungkapkan pendapat dan mengajukan pertanyaan <i>Binary Tree</i> dan <i>Search</i>	Mahasiswa mampu merespon secara aktif materi yang diberikan dengan cara bertanya dan berdiskusi.	6%

	dan mampu membuat program queue dalam Python			Tree		
16	Mahasiswa mengerti dan paham tree traversal, Algoritma Level Order Tree Traversal, Traversal Tree Inorder, pre-post Traversal dan mampu membuat tree traversal dalam Python	Tree Traversal	Ceramah, Tanya jawab dan penyelesaian soal	Mengungkapkan pendapat dan mengajukan pertanyaan mengenai tree traversal	Mahasiswa mampu merespon secara aktif materi yang diberikan dengan cara bertanya dan berdiskusi.	6%
17	Mahasiswa mengerti dan paham sejarah dan termininologi graph, manfaat graph dan analisis jaringan graph dan mampu membuat ghaph dalam Python.	Graph	Ceramah, Tanya jawab dan penyelesaian soal	Mengungkapkan pendapat dan mengajukan pertanyaan mengenai graph	Mahasiswa mampu merespon secara aktif materi yang diberikan dengan cara bertanya dan berdiskusi	6%
18	Mahasiswa mengerti dan paham algoritma Djiktra, greedy approach, cara kerja algoritma djiktra dan	Algoritma Djikstra	Ceramah, Tanya jawab dan penyelesaian soal latihan	Latihan 9	Mahasiswa mampu merespon, bertanya dan membuat program dengan Algoritma djiktra	6%

	mampu membuat program dengan algoritma djiktra dengan Python					
UAS						

Referensi:

- Amanulhaq, A. A. (2021). Implementasi Algoritma Image Hashing Dan Hamming Distance Untuk Deteksi Kemiripan Gambar. *Repository Its.*
- Emi Sita Eriana, A. Z. (2021). Penerapan Metode Personal Extreme Programming Dalam Perancangan Aplikasi Pemilihan Ketua Hmsi Dengan Weighted Product. *Jurnal Ilmu Komputer*, 27-32.
- Emi Sita Eriana, A. Z. (2021). *Praktikum Algoritma Dan Pemrograman*. Tangerang Selatan: Unpam Press.
- Eriana, E. S. (2020). Pemilihan Ketua Himtif Universitas Pamulang Dengan Metode Simple Additive Weighting (Saw). *Jik(Jurnal Ilmu Komputer)*.
- Jodi, U. R. (2020). *Algoritma Dan Struktur Data*.
- 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). Pendekripsi Kantuk Secara Real Time Menggunakan Pustaka Opencv Dan Dlib Python. *Sainstech : Jurnal Penelitian Dan Pengkajian Sains Dan Teknologi*.
- Zein, A. (2019). Pendekripsi Penyakit Malaria Menggunakan Medical Images Analisis Dengan Deep Learning Python. *Jurnal Penelitian Dan Pengkajian Sains Dan Teknologi*.
- Zein, A. (2020). Pendekripsi Virus Corona Dalam Gambar X-Ray Menggunakan Algoritma Artificial Intelligence Dengan Deep Learning Python. *Jurnal Esit (E-Bisnis, Sistem Informasi, Teknologi Informasi)*.

Ketua Program Studi

S1 Sistem Informasi

Dede Supriyadi, S.Kom., M.Kom.
NIDN. 0403078402

Tangerang Selatan, 14 Januari 2022

Ketua Tim Penyusun

Emi Sita Eriana, S. Kom., M. Kom.
NIDN. 0303028802