# PERTEMUAN 8

# BUBBLE SORT

## A. TUJUAN PEMBELAJARAN

Setelah mempelajari ini diharapkan mahasiswa mengerti dan paham teknik sorthing 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.1Hasilnya 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. Mengoptomalkan 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 melewatkan langkah-langkah yang tidak diperlukan.

### 3. Representasi Visual

Mengingat daftar lima elemen, gambar berikut menggambarkan bagaimana bubble sortiterates 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

2) Kasus terbaik – ini terjadi ketika daftar yang disediakan sudah diurutkan. Algoritma melakukan jumlah minimum eksekusi yang dinyatakan sebagai [Big-Omega]

3) Kasus rata-rata – ini terjadi ketika daftar dalam urutan acak. Kompleksitas rata-rata direpresentasikan sebagai [Big-theta]

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 menukarnya 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 penggortir 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 melewatkan iterasi yang tidak perlu saat memeriksa nilai yang telah diurutkan

## 6. Insertion Sort (Metode Penyisipan)

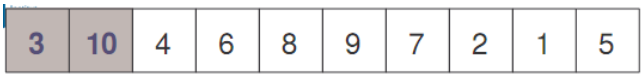
Dalam kebanyakan kasus, jenis penyisipan adalah yang terbaik dari jenis dasar yang dijelaskan di sini bab. Ini masih dijalankan dalam waktu O (N2), tetapi sekitar dua kali lebih cepat dari jenis sorting bubble/gelembung dan agak lebih cepat daripada jenis pemilihan dalam situasi normal. Ini juga tidak 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.

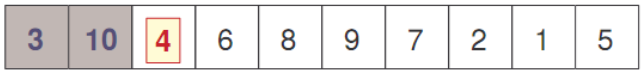
Insertion Sort bekerja seperti banyak orang yang sedang mengurutkan kartu di tangan. Dimulai dengan tangan kiri yang kosong dan kartunya tertumpuk di meja. Selanjutnya kita ambil satu persatu kartu di meja dan diletakkan di tangan kiri dengan posisi yang benar (terurut). Untuk menemukan posisi yang banar,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

## 7. Visualisasi Insertion Sort

Visualisasi pada insertsort digambarkan diabwah ini.

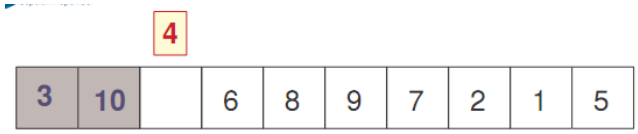


Bagian biru/abu-abu (dua bilangan pertama) sekarang dalam keadaan terurut secara relatif.

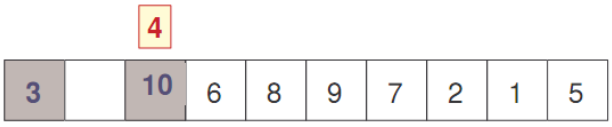



Berikutnya, kita perlu menyisipkan bilangan ketiga (4) ke dalam bagian biru/abu-abu sehingga setelah penyisipan tersebut, bagian biru/abu-abu tetap dalam keadaan terurut secara relatif. Caranya :

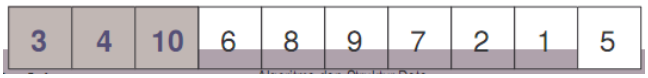
pertama : Ambil bilangan ketiga (4).



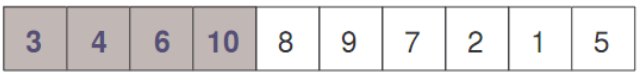

Kedua : Geser bilangan kedua (10) shg ada ruang untuk disisipi.



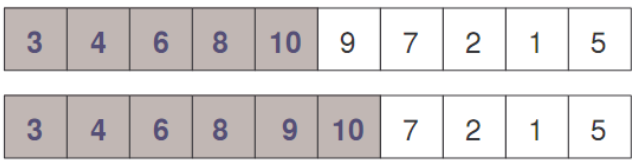

Ketiga : Sisipkan bilangan 4 ke posisi yang tepat

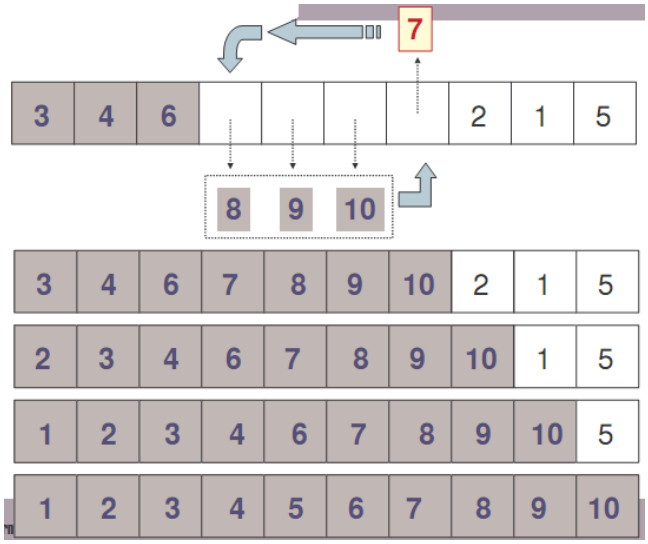



Sekarang, tiga bilangan pertama sudah terurut secara relatif dan kita sisipkan bilangan keempat kepada tiga bilangan pertama tsb. Setelah penyisipan, empat bilangan pertama haruslah dalam keadaan terurut secara relatif.




Ulangi proses tersebut sampai bilangan terakhir disisipkan





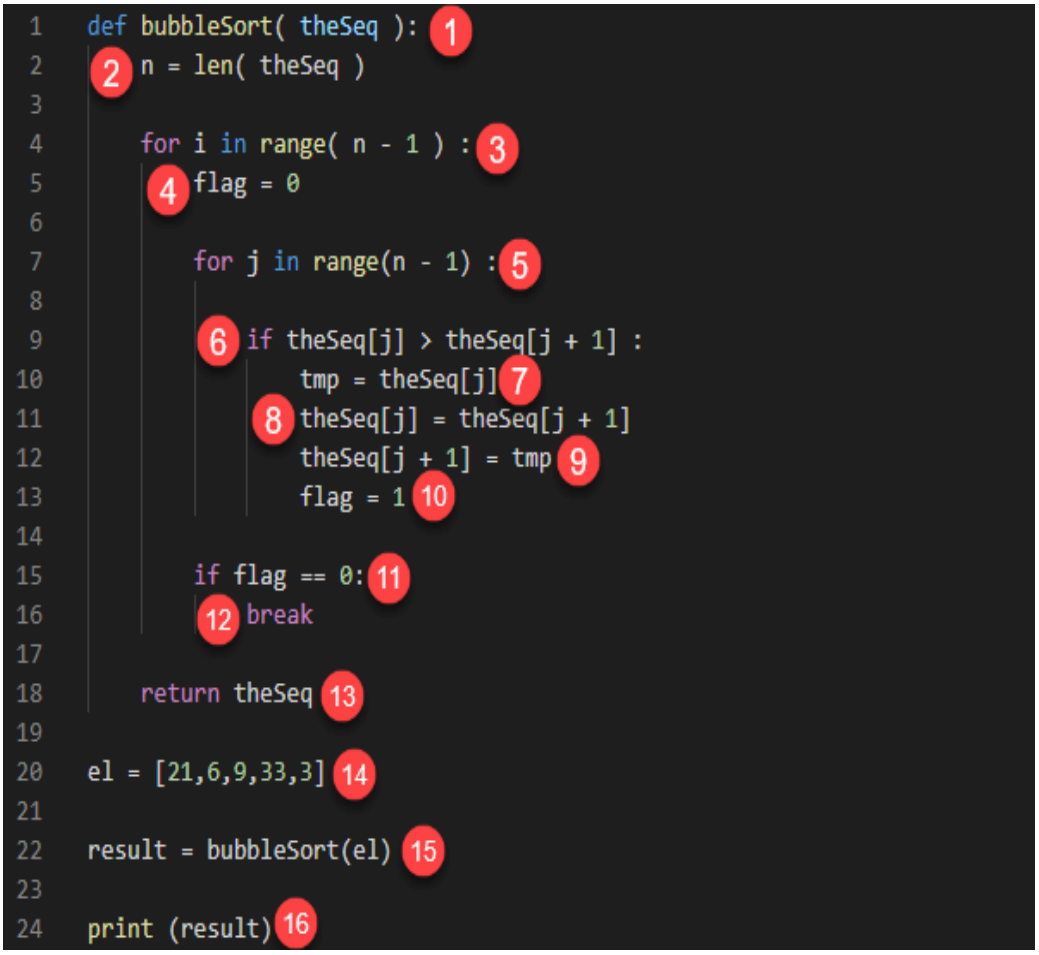

## 8. Implementasi Bubble Sort dan Insert Sort di Python

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   2 n = len( theSeq )
3
4       for i in range( n - 1 ) : 3
5       4 flag = 0
6
7           for j in range(n - 1) : 5
8
9           6 if theSeq[j] > theSeq[j + 1] :
10                  tmp = theSeq[j] 7
11              8 theSeq[j] = theSeq[j + 1]
12                  theSeq[j + 1] = tmp 9
13                  flag = 1 10
14
15          if flag == 0: 11
16          12 break
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.

a. Mendefinisikan fungsi bubbleSort yang menerima parameter theSeq. Kode tidak menghasilkan apapun.

b. Menperoleh panjang array dan menetapkan nilai ke variabel n. Kode tidak menghasilkan apa-apa

c. Memulai loop untuk yang menjalankan algoritma bubble sort (n – 1) kali. Ini adalah loop luar.

d. Mendefinisikan variabel yang akan digunakan untuk menentukan apakah swap telah terjadi atau tidak. Ini untuk tujuan optimasi

e. Mulai loop dalam yang membandingkan semua nilai dalam daftar dari yang pertama hingga yang terakhir.

f. 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.