LAPORAN TUGAS BESAR
MATA KULIAH STRUKTUR DATA

Pengimplementasian Struktur Data Stack Pada Data Pasien Rehabilitasi

Disusun Oleh Kelompok 1 :

Kanaya Dea Thalita Akhmad 121450001

Sarah Natalita Geraldine 121450022

Ivander Perdana Mokhtar 121450067

Silvia Azahrani 121450070

Deodry Siahaan 121450151

ABSTRAK

Di era teknologi yang berkembang pesat saat ini, dan bertambahnya jumlah data yang semakin membludak yang membuat tidak memungkinkan nya menginput data secara tradisional atau manual lagi. Pencarian data dan pembuatan laporan yang masih kurang efisien dan efektif, dikarenakan masih banyak nya instansi-instansi yang masih menggunakan sistem manual. Pengelolaan data secara manual mempunyai banyak kelemahan, selain membutuhkan waktu yang lama, keakuratannya juga kurang dapat diterima, karena kemungkinan kesalahan sangat besar dapat terjadi sehingga membutuhkan adanya sistem terkomputerisasi untuk membantu pengolahan data pasien. Sistem dibangun menggunakan untuk mengolah data pasien rehabilitas. Perancangan sistem menggunakan Flowchart dan pemrograman bahasa Python. Hasil perancangan memperlihatkan bahwa sistem dapat mengolah data-data dengan baik.
Kata kunci : data, sistem, data pasien rehabilitasi, efisien dan efektif

BAB I
PENDAHULUAN

1.2. Latar Belakang

Dalam kehidupan sehari hari seringkali manusia seringkali berkaitan dalam hal pengelolaan dokumen. Biasanya banyak digunakan oleh institusi dalam administrasi datanya. Seperti halnya sekolah, kantor pemerintah, ataupun instansi rumah sakit dan instansi masyarakat lainnya. Dalam pengelolaan dokumen ada banyak sekali platform yang dapat digunakan, bisa seperti microsoft office ataupun menggunakan bahasa pemrograman yang disesuaikan sesuai kebutuhan. Salah satu bahasa yang paling banyak digunakan dalam pemrosesan data saat ini yaitu python.

Salah satu instansi masyarakat yang aktif saat ini adalah Pusat Rehabilitasi BNN. Dalam instansi masyarakat yang semacam ini tentunya diperlukan pengelolaan dokumen bagi pasien pasien rehabilitasi, data dari setiap pasien sangat penting sebagai penggambaran kesehatan dari tiap tiap pasien. Setiap pasien memiliki perbedaan data tentunya, terlebih dalam hal penggunaan obat. Rehabilitasi disesuaikan dengan jenis obat yang digunakan akan mendukung penyembuhan pasien lebih baik. Oleh karena itu pengelolaan data pasien sangat penting bagi institusi pusat rehabilitasi.

Diperlukan suatu program dalam pengelolaan data pasien yang cukup banyak. Ada banyak jenis program yang dimiliki bahasa python, kami menggunakan pendekatan masalah dengan struktur data yang paling sederhana yaitu Stack. Berdasarkan kebutuhan penyusunan dokumen pusat rehabilitasi program ini sangat cocok karena bersifat sederhana dan praktis. Oleh karena itu dalam kajian program yang kami buat menggunakan program Stack sebagai jenis struktur datanya, dengan judul penelitian yaitu "*Pengimplementasian struktur data stack pada data pasien rehabilitasi*"

1.3. Masalah

Berdasarkan latar belakang, dapat diidentifikasi permasalahan dalam pengelolaan data pasien rehabilitas dan berdasarkan kebutuhan penyusunan data pasien, analisis yang akan dilakukan adalah data yang paling terakhir dimasukan ke dalam merupakan data yang pertama kali keluar, dengan menggunakan program Stack sebagai struktur datanya.

1.4. Deskripsi Data

Data yang disajikan berupa beragam data jenis kuantitatif yang disajikan dalam data berbentuk .csv, dengan 200 baris dan 6 kolom, dimana untuk kolom pertama berisi Usia, kemudian untuk kolom kedua ada jenis kelamin, untuk kolom ketiga berisi BP, kolom keempat ada cholestrol, dikolom lima ada Na_to_K, dan kolom terakhir atau kolom 6 berisi Drug.

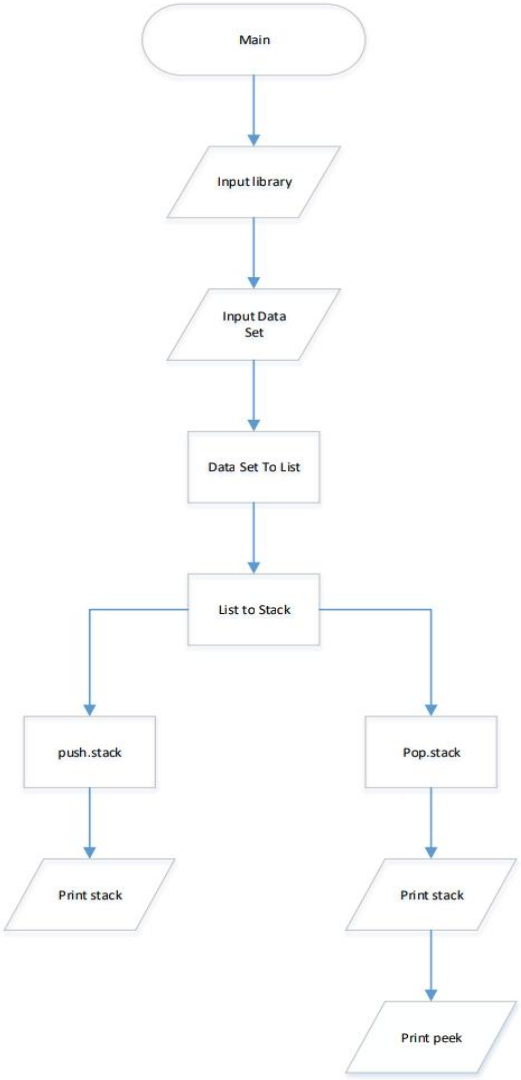
BAB II
METODE DAN RANCANGAN SISTEM

2.1. Gambaran Umum Metode yang Digunakan

Pada program yang telah dibuat, diterapkan teknik yang telah dipelajari pada mata kuliah Struktur Data. Pada program kali ini data yang digunakan yaitu data yang telah diunduh pada kaggle, data tersebut berisi data pasien rehabilitasi yang kemudian diolah ke dalam struktur data linear (Stack). Stack sendiri merupakan struktur data yang dapat ditunjukkan oleh tempat penyisipan dan penghapusan elemen terjadi hanya pada satu tempat yang disebut puncak tumpukan dan pada class Stack ini memiliki konsep LIFO (last in first out) atau dapat diartikan data yang terakhir masuk, maka ialah yang pertama akan dikeluarkan. Dalam program kali ini data yang diolah akan melewati beberapa proses yaitu:

1. Input Data
2. Ubah dataset menjadi data frame
3. Step pengolahan data
    - Insert
    - Hapus
    - Peek
4. Step penambah data secara manual
    - append list
    - list to stack

2.2. Flowchart Sistem



2.3. Dependency / Library yang digunakan

A. Pandas

```
import numpy as np
```

Pandas adalah open source pada Python yang sering digunakan untuk memproses suatu data yang meliputi pembersihan data, manipulasi data, hingga menganalisis data. Penggunaannya memudahkan analisis data, manipulasi data, dan pembersihan data. Pandas mendukung berbagai jenis operasi seperti penyortiran, pengindeksan ulang, iterasi, penggabungan, konversi data, visualisasi, agregasi, dan lain sebagainya.

Pada program ini Pandas digunakan untuk membuat data frame dan menghapus kolom yang tidak diperlukan.

2.4. Daftar Fungsionalitas

Fungsionalitas adalah sesuatu yang berisi tentang proses-proses apa saja yang nantinya akan dilakukan oleh sistem. Fungsionalitas berisi informasi-informasi apa saja yang harus ada dan dihasilkan oleh sistem. Berikut ini adalah daftar fungsionalitas yang digunakan dalam laporan ini:

- Using Data Frame
    - pd.read_csv('data.csv')
      Perintah diatas untuk mengimpor dataset ke dalam Pandas dataframe.
    - Drop Data Frame
      berfungsi untuk menghapus kolom dari dataset (digunakan dalam program ini agar efisien)
    - Values to List
      Data yang ada dalam data frame, perbarisnya diinput menjadi list
- Program Stack
    - Class Node
      Inisialisasi Stack (init.self)
    - Class Stack
        - String implementasi stack
        - Fungsi getSize():
          Digunakan untuk melihat ukuran stack.
        - Fungsi isEmpty():
          Digunakan untuk mengecek apakah stack kosong atau tidak.
        - Fungsi peek():
          Digunakan untuk melihat item paling teratas dalam stack.
        - Fungsi self.isEmpty():
          Digunakan untuk mengecek apakah kita mengembalikan stack yg kosong.
        - Fungsi push():
          Digunakan untuk menambah nilai dalam stack.
        - Fungsi pop():
          Digunakan untuk menghapus nilai dalam stacknya dan mengembalikannya.

    - Main Driver
      Digunakan untuk menjalankan program stack dari fungsi yang telah dibuat. Diberikan fungsi push yang nantinya akan memasukkan semua dataset yang telah diubah ke dalam bentuk list. Juga dilakukan fungsi pop untuk menghapus data paling belakang.
- Fungsi push dari list yang dibuat manual
    - Buat list, lalu isi list dengan fungsi append
    - Push list to Stack

BAB III
HASIL PROGRAM
(screenshot use case)

```
import pandas as pd
```

```
tb = pd.read_csv('/content/drug200.csv')
tb
```

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	DrugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	DrugY
...	...	...	...	...	...	...
195	56	F	LOW	HIGH	11.567	drugC
196	16	M	LOW	HIGH	12.006	drugC
197	52	M	NORMAL	HIGH	9.894	drugX
198	23	M	NORMAL	NORMAL	14.020	drugX
199	40	F	LOW	NORMAL	11.349	drugX

200 rows × 6 columns

```
[6] tb = tb.drop(["BP", "Cholesterol", "Na_to_K"], axis = 1)
```

```
data = tb.values.tolist()

# Printing dataframe ke bentuk list
print(data)
```

```
[[23, 'F', 'DrugY'], [47, 'M', 'drugC'], [47, 'M', 'drugC'], [28, 'F', 'drugX'], [61, 'F', 'DrugY'], [22, 'F', 'drugX
```

```
[ ] print(data[1])
    print(data[0])
```

```
[47, 'M', 'drugC']
[23, 'F', 'DrugY']
```

```python
# implementasi stack menggunakan linked list
# buat node class

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


class Stack:

  # inisialisasi stacknya
  # easier for handling edge cases.
  def __init__(self):
    self.head = Node("head")
    self.size = 0
```

```python
  # string ini untuk mengimplementasikan stacknya
  def __str__(self):
    cur = self.head.next
    out = ""
    while cur:
      out += str(cur.value) + "->"
      cur = cur.next
    return out[:-3]

  # ini untuk melihat ukuran si stack
  def getSize(self):
    return self.size

  # mengecek apakah stack kosong atau tidak
  def isEmpty(self):
    return self.size == 0

  # melihat item paling teratas dalam stack
  def peek(self):

    # untuk mengecek apakah kita mengembalikan stack yg kosong
    if self.isEmpty():
      raise Exception("Peeking from an empty stack")
    return self.head.next.value
```



```python
# menambah nilai dalam stack
def push(self, value):
  node = Node(value)
  node.next = self.head.next
  self.head.next = node
  self.size += 1

# menghapus nilai dalam stacknya dan mengembalikannya
def pop(self):
  if self.isEmpty():
    raise Exception("Popping from an empty stack")
  remove = self.head.next
  self.head.next = self.head.next.next
  self.size -= 1
  return remove.value
```

```
# menjalankan codenya, driver code
if __name__ == "__main__":
  stack = Stack()
  for i in data:
    stack.push(i)
  print(f"Stack: {stack}")


  for _ in range(0, 10):
    remove = stack.pop()
    print(f"Pop: {remove}")
  print(f"Stack: {stack}")

  if i in data : #liat top
    top = stack.peek()
    print(top)
```

Outputnya :

```
Stack: [40, 'F', 'drugX']->[23, 'M', 'drugX']->[52, 'M', 'drugX']->[16, 'M', 'drugC']->[56, 'F', 'drugC']->[46, 'F',
Pop: [40, 'F', 'drugX']
Pop: [23, 'M', 'drugX']
Pop: [52, 'M', 'drugX']
Pop: [16, 'M', 'drugC']
Pop: [56, 'F', 'drugC']
Pop: [46, 'F', 'DrugY']
Pop: [72, 'M', 'drugC']
Pop: [72, 'M', 'DrugY']
Pop: [23, 'M', 'drugA']
Pop: [58, 'M', 'DrugY']
Stack: [64, 'M', 'DrugY']->[65, 'M', 'DrugY']->[47, 'M', 'drugA']->[70, 'M', 'drugB']->[57, 'F', 'DrugY']->[18, 'F',
[64, 'M', 'DrugY']
Stack : (<__main__.Stack object at 0x7f82829d4100>, 20)
```

Menambahkan data baru :

```
  baru = []
  baru.append(34)
  baru.append('F')
  baru.append('drugA')
  print(baru)
  for _ in range(-1) :
    stack.push(baru)
  print(f"Stack: {baru}->{stack}")
```

```
[34, 'F', 'drugA']
Stack: [34, 'F', 'drugA']->[64, 'M', 'DrugY']->[65, 'M', 'DrugY']->[47, 'M', 'drugA']->[70, 'M', 'drugB']->[57, 'F',
```

BAB IV
KESIMPULAN

Berdasarkan hasil Pengimplementasikan Struktur Data Stack pada Data Pasien Rehabilitasi yang telah dilakukan, maka dapat disimpulkan bahwa :

- Dengan menggunakan metode Stack, kita dapat mengimplementasi data pasien dengan dengan menggunakan konsep LIFO (last in first out)
- Stack yang digunakan membutuhkan beberapa proses yaitu : Input Data , Ubah dataset menjadi data frame, Step pengolahan data (Insert, Hapus, Peek), Step penambah data secara manual (append list, list to stack)

Referensi

Goodrich, Michael T., et al. *Data structures and algorithms in Python*. John Wiley & Sons Ltd, 2013.

Hugo Lopes Tavares, and Gustavo Guimarães Rezende. “A tool stack for implementing Behaviour-Driven Development in Python Language.” 2010. https://arxiv.org/ftp/arxiv/papers/1007/1007.1722.pdf.