Nama : Tegar Adimas Nugroho
NIM : G.211.22.0088
Kelas : B1 (Pagi)

Algoritma Pada
“ANTRIAN”

```
from collections import deque
queue = deque(["Ram", "Tarun", "Asif", "John"])
print(queue)
queue.append("Akbar")
print(queue)
queue.append("Birbal")
print(queue)
print(queue.popleft())
print(queue.popleft())
print(queue)
```

A. Inisialisasi Deque:
    - Import modul deque dari collections.
    - Buat objek deque dengan elemen awal "Ram", "Tarun", "Asif", dan "John".
B. Operasi Pertama:
    - Cetak isi deque.
C. Operasi Kedua:
    - Tambahkan "Akbar" ke ujung kanan (rear) deque menggunakan metode append.
    - Cetak isi deque setelah penambahan.
D. Operasi Ketiga:
    - Tambahkan "Birbal" ke ujung kanan (rear) deque menggunakan metode append.
    - Cetak isi deque setelah penambahan.
E. Operasi Keempat:
    - Keluarkan (remove) dan cetak elemen paling kiri (front) deque menggunakan metode popleft.
F. Operasi Kelima:
    - Keluarkan (remove) dan cetak elemen paling kiri (front) deque menggunakan metode popleft.
G. Operasi Keenam:
    - Cetak isi deque setelah dua operasi sebelumnya

Jadi, secara umum, algoritma yang dijalankan oleh source code ini adalah: Inisialisasi deque, tambahkan elemen ke ujung kanan deque, hapus elemen dari ujung kiri deque, ulangi operasi sebelumnya sesuai kebutuhan.

“TOWER HANOI”

```
def TowerOfHanoi (n, dr, menuju, bantuan):
        if n==0:
          return
        TowerOfHanoi(n-1, dr, bantuan, menuju)
        print("Pindah disk", n, "dari", dr, "ke", menuju)
        TowerOfHanoi(n-1, bantuan, dr, menuju)
# Driver code
N = 3
# A, C, B are the name of rods
TowerOfHanoi(N, 'A', 'C', 'B')
```

1. **Fungsi Tower of Hanoi :**
    - Parameter :
        a. n: Jumlah disk yang akan dipindahkan.
        b. dr: Rod awal dari mana disk akan dipindahkan.
        c. menuju: Rod tujuan ke mana disk akan dipindahkan.
        d. bantuan: Rod bantuan yang digunakan selama proses pemindahan.
    - Jika n sama dengan 0, maka kembalikan dari fungsi (base case).
    - Rekursif :
        a. Pindahkan n-1 disk dari rod awal (dr) ke rod bantuan (bantuan), dengan rod tujuan (menuju) sebagai rod bantuan.
        b. Cetak pesan yang menyatakan pemindahan disk ke-n dari rod awal (dr) ke rod tujuan (menuju).
        c. Pindahkan n-1 disk dari rod bantuan (bantuan) ke rod tujuan (menuju), dengan rod awal (dr) sebagai rod bantuan.

2. **Driver Code :**
    a. Tetapkan nilai N sebagai 3.
    b. Panggil fungsi TowerOfHanoi dengan parameter N, dan rod awal, rod tujuan, serta rod bantuan yang sesuai.

3. **Output :**
    Setiap langkah pemindahan disk dicetak sebagai pesan pada setiap pemanggilan rekursif

Algoritma ini menggambarkan pemindahan yang diperlukan untuk menyelesaikan masalah Tower of Hanoi dengan menggunakan rekursi. Pada setiap tahap rekursif, fungsi memecah masalah menjadi submasalah yang lebih kecil hingga mencapai base case.

“**Mengurutkan Dalam Sebuah Stack”**

```
def sortStack(input):
        tmpStack = []
        while (len(input) > 0):
                # pop out the first element
                tmp = input[-1]
                input.pop()

                # while temporary stack is not empty
                # and top of stack is smaller than temp
                while (len(tmpStack) > 0 and tmpStack[-1] < tmp):

                        # pop temporary stack and
                        # append it to the input stack
                        input.append(tmpStack[-1])
                        tmpStack.pop()

                # append temp in temporary of stack
                tmpStack.append(tmp)

        return tmpStack

def sortArrayUsingStacks(arr, n):

        # append array elements to stack
        input = [] i = 0 while (i < n):
                input.append(arr[i])
                i = i + 1

        # sort the temporary stack
        tmpStack = sortStack(input)
        i = 0

        # put stack elements in arrp[]
        while (i < n):
                arr[i] = tmpStack[-1]
                tmpStack.pop()
                i = i + 1

                return arr
# Driver code
```

```
arr = [10, 5, 15, 45]
n = len(arr)

arr = sortArrayUsingStacks(arr, n)
i = 0

while (i < n):
        print(arr[i], end= " ")
        i = i + 1
```

Fungsi sortStack :

1. Inisialisasi tumpukan sementara (tmpStack) sebagai tumpukan kosong.
2. Selama tumpukan input tidak kosong:
    - Ambil elemen teratas dari tumpukan input.
    - Selama tumpukan sementara tidak kosong dan elemen teratasnya lebih kecil dari elemen yang diambil dari input:
        - Pop elemen teratas dari tumpukan sementara dan tambahkan ke tumpukan input.
    - Tambahkan elemen yang diambil dari input ke tumpukan sementara.
3. Kembalikan tumpukan sementara yang sudah diurutkan.

Fungsi sortArrayUsingStacks:

1. Inisialisasi tumpukan input sebagai tumpukan kosong.
2. Selama i kurang dari n (jumlah elemen dalam array):
    - Tambahkan elemen ke-i dari array ke tumpukan input.
    - Tingkatkan nilai i.
3. Panggil fungsi sortStack dengan parameter tumpukan input.
4. Selama i kurang dari n:
    - Ambil elemen teratas dari tumpukan hasil pengurutan.
    - Masukkan elemen tersebut ke array pada posisi ke-i.
    - Kurangkan nilai i.
5. Kembalikan array yang sudah diurutkan.

Driver Code :

1. Inisialisasi array [10, 5, 15, 45].
2. Panggil fungsi sortArrayUsingStacks dengan array dan panjang array sebagai parameter.
3. Selama i kurang dari n:
    - Cetak elemen ke-i dari array.
    - Tingkatkan nilai i.

Output :

Output dari kode ini adalah elemen-elemen array yang sudah diurutkan secara menurun. Algoritma ini menggunakan pendekatan tumpukan untuk mengurutkan elemenelemen array, dimulai dari tumpukan input dan menggunakan tumpukan sementara Output dari kode ini adalah elemen-elemen array yang sudah diurutkan secara menurun. Algoritma ini menggunakan pendekatan tumpukan untuk mengurutkan elemenelemen array, dimulai dari tumpukan input dan menggunakan tumpukan sementara