Nama : Andrea Prasetyo Hariawan

NIM : 20220010

Matakuliah : Praktikum Desain Analisis dan algoritma

Laporan praktikum analisis khusus terburuk, rata-rata dan terbaik

1. Algoritma bubble sort

```
def bubbleSort(arr):
    n = len(arr)
    for i in range(n-1):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr
```

Script di atas merupakan implementasi algoritma Bubble Sort untuk mengurutkan elemen-elemen dalam sebuah array. Algoritma Bubble Sort bekerja dengan membandingkan pasangan-pasangan elemen adjacent dalam array dan menukar posisinya jika urutan mereka tidak sesuai. Proses ini dilakukan secara berulang hingga seluruh elemen dalam array terurut dengan benar.

Langkah-langkah utama dalam algoritma Bubble Sort sebagai berikut:

a. Fungsi bubbleSort menerima parameter arr yang merupakan array yang akan diurutkan.
b. Variabel n digunakan untuk menyimpan panjang (jumlah elemen) dari array.
c. Dalam loop pertama (for i in range(n-1)), dilakukan iterasi sebanyak n-1 kali untuk mengatur posisi elemen terakhir pada setiap iterasi.
d. Dalam loop kedua (for j in range(0, n-i-1)), dilakukan iterasi sebanyak n-i-1 kali untuk membandingkan pasangan-pasangan elemen adjacent dan melakukan pertukaran jika diperlukan.
e. Pada setiap iterasi loop kedua, jika elemen saat ini (arr[j]) lebih besar dari elemen berikutnya (arr[j+1]), maka dilakukan pertukaran posisi elemen tersebut menggunakan teknik "swapping".
f. Setelah kedua loop selesai, array akan terurut dengan benar.
g. Fungsi mengembalikan array yang telah diurutkan.

Dengan menggunakan algoritma Bubble Sort ini, elemen-elemen dalam array akan bergerak seperti gelembung yang naik ke permukaan sesuai dengan urutan yang diinginkan

2. Algoritma marge sort

```
def mergeSort(arr):
    if len(arr) > 1:
        mid = len(arr) // 2
        L = arr[:mid]
        R = arr[mid:]

        mergeSort(L)
        mergeSort(R)

        i = j = k = 0

        while i < len(L) and j < len(R):
            if L[i] < R[j]:
                arr[k] = L[i]
                i += 1
            else:
                arr[k] = R[j]
                j += 1
            k += 1

        while i < len(L):
            arr[k] = L[i]
            i += 1
            k += 1

        while j < len(R):
            arr[k] = R[j]
            j += 1
            k += 1

    return arr
```

Script di atas merupakan implementasi algoritma Merge Sort untuk mengurutkan elemen-elemen dalam sebuah array. Algoritma Merge Sort menggunakan pendekatan "divide and conquer" dengan membagi array menjadi subarray-subarray yang lebih kecil, mengurutkan subarray tersebut secara terpisah, dan kemudian menggabungkannya kembali untuk mendapatkan array yang terurut.

Langkah-langkah utama dalam algoritma Merge Sort sebagai berikut:

a. Fungsi mergeSort menerima parameter arr yang merupakan array yang akan diurutkan.

b. Pada awalnya, fungsi melakukan pengecekan apakah panjang array arr lebih dari 1. Jika panjangnya lebih kecil atau sama dengan 1, maka array sudah terurut dan tidak perlu dilakukan pengurutan lebih lanjut.
c. Jika panjang array lebih dari 1, maka array akan dibagi menjadi dua bagian, yaitu subarray L (setengah bagian kiri) dan subarray R (setengah bagian kanan).
d. Langkah selanjutnya adalah melakukan rekursi pada mergeSort untuk kedua subarray L dan R tersebut. Hal ini dilakukan untuk mengurutkan kedua subarray secara terpisah.
e. Setelah kedua subarray terurut, dilakukan proses penggabungan (merge) subarray-subarray tersebut dalam langkah yang disebut "merge operation".
f. Dalam proses merge, dilakukan iterasi menggunakan tiga variabel: i untuk subarray L, j untuk subarray R, dan k untuk array hasil penggabungan.
g. Pada setiap iterasi, elemen terkecil antara L[i] dan R[j] dipilih dan dimasukkan ke array hasil penggabungan (arr[k]). Variabel k, i, dan j diperbarui sesuai dengan elemen yang telah dipilih.
h. Setelah salah satu subarray L atau R habis diiterasi, elemen yang tersisa dari subarray yang belum habis diiterasi langsung dimasukkan ke array hasil penggabungan.
i. Setelah proses merge selesai, array akan terurut dengan benar.
j. Fungsi mengembalikan array yang telah diurutkan.

Dengan menggunakan algoritma Merge Sort ini, array akan terbagi dan diurutkan secara rekursif hingga mencapai ukuran subarray yang paling kecil, kemudian subarray-subarray tersebut digabungkan kembali dengan cara yang benar untuk menghasilkan array yang terurut.

3. Algortima insertion sort

```
def insertionSort(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i - 1
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key
    return arr
```

Script di atas merupakan implementasi algoritma Insertion Sort untuk mengurutkan elemen-elemen dalam sebuah array. Algoritma Insertion Sort bekerja dengan membagi array menjadi dua bagian: bagian terurut (bagian awal) dan bagian belum terurut (bagian sisa). Kemudian, satu per satu, elemen-elemen dari bagian belum terurut akan diambil dan dimasukkan ke posisi yang tepat dalam bagian terurut.

Langkah-langkah utama dalam algoritma Insertion Sort sebagai berikut:

a. Fungsi `insertionSort` menerima parameter `arr` yang merupakan array yang akan diurutkan.
b. Iterasi dimulai dari indeks kedua (i = 1) hingga indeks terakhir array.
c. Pada setiap iterasi, elemen pada indeks `i` dianggap sebagai elemen yang akan dimasukkan ke dalam bagian terurut.
d. Variabel `key` digunakan untuk menyimpan elemen pada indeks `i`.
e. Variabel `j` diatur sebagai indeks sebelum `i` (j = i - 1).
f. Dalam loop while, dilakukan pergeseran elemen-elemen yang lebih besar dari `key` ke posisi kanan, untuk memberikan ruang bagi `key` untuk dimasukkan ke posisi yang tepat dalam bagian terurut.
g. Loop while akan berjalan selama `j` lebih besar atau sama dengan 0, dan elemen di indeks `j` lebih besar dari `key`.
h. Pada setiap iterasi loop while, elemen di indeks `j` digeser ke posisi kanan (`arr[j + 1] = arr[j]`), dan variabel `j` dikurangi 1.
i. Setelah loop while selesai, `key` dimasukkan ke posisi yang tepat dalam bagian terurut, yaitu pada indeks `j + 1`.
j. Proses ini dilakukan untuk semua elemen dalam array, sehingga menghasilkan array yang terurut dengan benar.
k. Fungsi mengembalikan array yang telah diurutkan.

Dengan menggunakan algoritma Insertion Sort ini, elemen-elemen dalam array akan secara berurutan "dimasukkan" ke bagian terurut, sehingga bagian terurut secara bertahap membesar hingga seluruh array terurut dengan benar.