DOKUMENTASI ALGORITMA SORTING MENGGUNAKAN TEKNIK QUICK SORT

DISUSUN OLEH :
NIM:
2209116022

NAMA MAHASISWA:
MUHAMMAD DIAN NURDIANSYAH

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS TEKNIK

UNIVERSITAS MULAWARMAN

2023

1. Penjelasan Algoritma

Algoritma *Quick Sort* adalah salah satu algoritma pengurutan data (sorting) yang efisien dan sering digunakan. Algoritma ini menggunakan teknik *divide* and *conquer,* yaitu membagi data menjadi dua bagian dan mengurutkannya secara terpisah, lalu menggabungkannya menjadi satu. Algoritma ini juga menggunakan pendekatan rekursif dalam pemrosesan data.

Pada metode *Quick Sort*, elemen pivot digunakan sebagai batas pembagi antara elemen yang lebih kecil dan lebih besar darinya. Proses partisi dilakukan dengan memindahkan elemen-elemen kecil ke sisi kiri pivot dan elemen-elemen besar ke sisi kanan pivot. Setelah partisi selesai dilakukan, elemen pivot sudah berada pada posisi yang tepat di dalam data yang terurut.

Walaupun algoritma *Quick Sort* memiliki waktu eksekusi yang cepat dan efisien, namun jika pivot yang dipilih adalah elemen yang paling besar atau paling kecil di dalam data, maka algoritma ini tidak akan efisien dan bahkan bisa menjadi algoritma yang lambat. Oleh karena itu, biasanya pivot dipilih secara acak atau dipilih berdasarkan teknik pivot selection yang lebih canggih.

2. Penjelasan Program dan Visualisasinya

A. Source Code dan Output

```python
# Library random
import random

# Fungsi quicksort yang berisikan parameter array
def quicksort(arr):
    if len(arr) <= 1:
        return arr
    else:
        pivot = arr[0]
        datakiri = []
        datakanan = []
        # Melakukan perulangan untuk memerikasa elemen-elemen di dalam (arr).
        for i in range(1, len(arr)):
            if arr[i] < pivot:
                datakiri.append(arr[i])
            else:
                datakanan.append(arr[i])
        return quicksort(datakiri) + [pivot] + quicksort(datakanan)

# Membuat list random dengan elemen berjumlah 11 dan diisi dengan bilangan interger acak antara 1 sampai 25
randomlist = [random.randint(1, 25) for i in range(11)]

# Data list sebelum diurutkan
print("\nData Sebelum diurutkan:", randomlist)
print(75*"=")
print()
randomlist = quicksort(randomlist)
# Data list setelah diurutkan
print("\nData Setelah diurutkan:", randomlist)
print(75*"=")
```

Penjelasan source code:

A. Import random: Memanggil library random yang dapat mengisi sebuah list kosong dengan angka acak.

B. Def quicksort(arr): Membuat fungsi quicksort yang memiliki parameter arr (array).

C. if len(arr) <= 1:

return arr: Membuat percabangan yang jika data array nilainya kurang dari satu, maka nilai tersebut akan dikembalikan ke fungsi (arr).

D. else:

pivot = arr[0]: Untuk menentukan pivot dengan memilih elemen pertama dari (arr).

E. datakiri[]

datakanan[]: membuat dua list kosong untuk menampung data yang lebih kecil dan lebih besar dari pivot.

F. for i in range(1, len(arr)): Melakukan perulangan untuk memerikasa elemen-elemen di dalam (arr).

G. if arr[i] < pivot:

datakiri.append(arr[i]): Jika elemen nilainya lebih kecil dari pivot, maka akan di masukkan ke dalam list datakiri.

H. else:

datakanan.append(arr[i]): Jika elemen lebih besar atau sama dengan pivot maka akan dimasukkan ke dalam list datakanan.

I. return quicksort(datakiri) + [pivot] + quicksort(datakanan): Mengabungkan kembali elemen-elemen yang telah diurutkan secara rekursif.

J. randomlist = [random.randint(1, 50) for i in range(11)]: Membuat list random dengan elemen berjumlah 11 dan diisi dengan bilangan interger acak antara 1 sampai 25.

K. print("\nData Sebelum diurutkan:", randomlist): Cetak list yang berisikan data sebelum diurutkan.

L. randomlist = quicksort(randomlist): Memanggil fungsi quicksort untuk mengurutkan list.

M. print("\nData Setelah diurutkan:", randomlist): Cetak list setelah datanya diurutkan.

Dari code tersebut akan menghasilkan output sebagai berikut:

```
Sebelum diurutkan: [13, 5, 25, 20, 9, 8, 10, 14, 15, 18, 12]
================================================================================


Setelah diurutkan: [5, 8, 9, 10, 12, 13, 14, 15, 18, 20, 25]
================================================================================
PS C:\Users\mhddi\Desktop\Praktikum ASD>
```

B. Visualisasi Program

Dengan visualisasi, kita dapat melihat bagaimana data diurutkan dan bagaimana elemen pivot dipilih serta digunakan dalam setiap langkahnya. Selain itu, visualisasi juga membantu kita memahami waktu dan kompleksitas algoritma serta dapat membandingkannya dengan metode pengurutan data lainnya. Dengan demikian, visualisasi pada metode quicksort sangat berguna dalam mempercepat pemahaman dan implementasi algoritma. Berikut visualiasi dari program atau data yang telah saya buat dengan menggunakan data pivot kanan.