Perbandingan Runtime Code

Untuk Sorting Array Metode Merge, Bubble, Quick

Disusun untuk memenuhi tugas mata kuliah

Pendidikan Pancasila

Dosen Pengampu : Indira Syawanodya, M. Kom.

Disusun oleh:

Fadhil Anwar Ahsani

NIM 2407136

PROGRAM STUDI REKAYASA PERANGKAT LUNAK

KAMPUS UPI CIBIRU

UNVERSITAS PENDIDIKAN INDONESIA

2024

Tugas:
Dari algoritma yang telah anda pilih pada studi kasus 1, buatlah perbandinganrunning program (execution time) dengan 2 algoritma yang ada di slide (pilih: bubble sort, insertion sort, atau selection sort, quick sort, merge sort, radix sort counting sort).

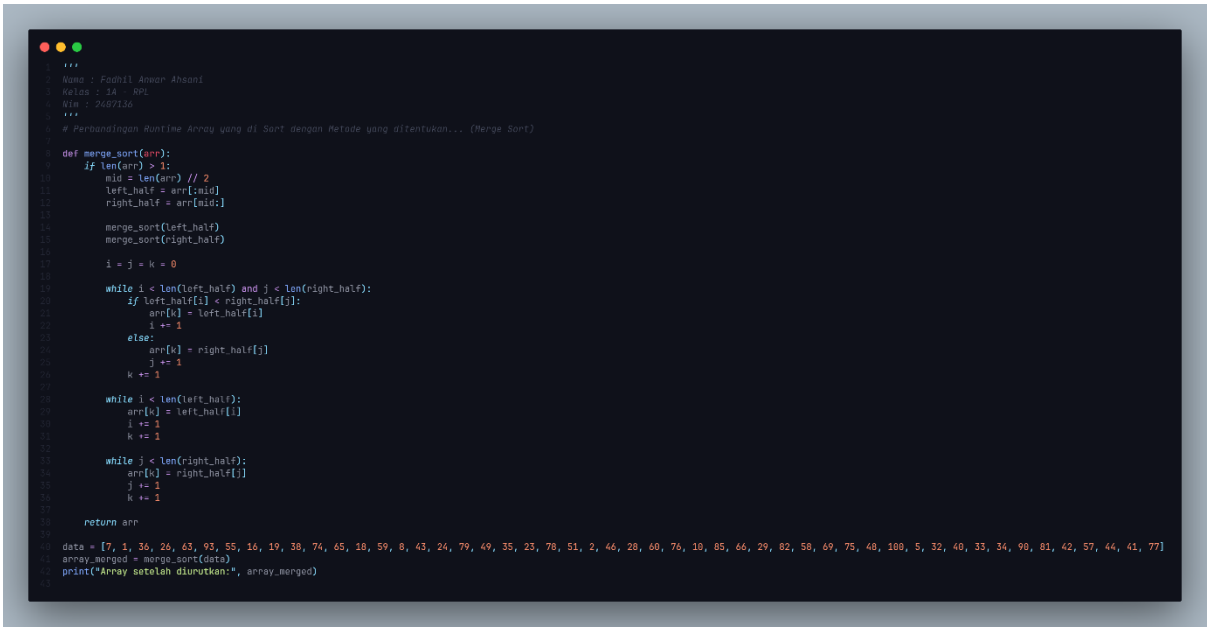
- Buatlah dengan menggunakan array acak berikut:

  [7, 1, 36, 26, 63, 93, 55, 16, 19, 38, 74, 65, 18, 59, 8, 43, 24, 79, 49, 35, 23, 78, 51, 2, 46, 28, 60, 76, 10, 85, 66, 29, 82, 58, 69, 75, 48, 100, 5, 32, 40, 33, 34, 90, 81, 42, 57, 44, 41, 77]

- Tentukan manakah dari ke 3 algoritma tersebut yang paling cepat?
- Kumpulkan file dalam bentuk copy kodingan dan screenshot hasil program dan simpan ke dalam file dengan format .pdf

Metode #1 – Merge Sort

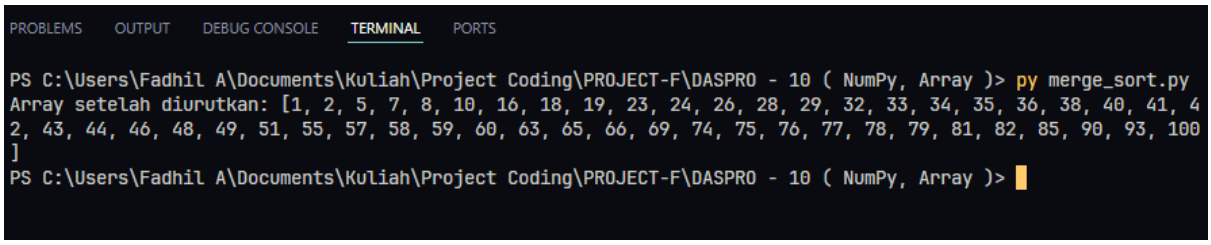
1. Code Merge Sort



```
'''
Nama : Fadhil Anwar Ihsani
Kelas : 1A - RPL
Nim : 2407136
'''
# Perbandingan Runtime Array yang di Sort dengan Metode yang ditentukan... (Merge Sort)

def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr) // 2
        left_half = arr[:mid]
        right_half = arr[mid:]

        merge_sort(left_half)
        merge_sort(right_half)

        i = j = k = 0

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

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

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

    return arr

data = [7, 1, 36, 26, 63, 93, 55, 16, 19, 38, 74, 65, 18, 59, 8, 43, 24, 79, 49, 35, 23, 78, 51, 2, 46, 28, 60, 76, 10, 85, 66, 29, 82, 58, 69, 75, 48, 100, 5, 32, 40, 33, 34, 90, 81, 42, 57, 44, 41, 77]
array_merged = merge_sort(data)
print("Array setelah diurutkan:", array_merged)
```

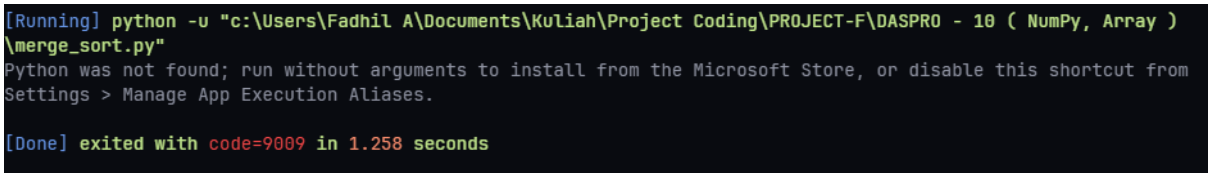
*untuk code yang ditampilkan untuk code merge_sort. Maaf terlalu kecil sudah terlanjur di snipping tools. Selanjutnya saya akan memakai CodeSnap Online di Internet agar kode selanjutnya lebih Jelas.

2. Output Merge Sort



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Fadhil A\Documents\Kuliah\Project Coding\PROJECT-F\DASPRO - 10 ( NumPy, Array )> py merge_sort.py
Array setelah diurutkan: [1, 2, 5, 7, 8, 10, 16, 18, 19, 23, 24, 26, 28, 29, 32, 33, 34, 35, 36, 38, 40, 41, 4
2, 43, 44, 46, 48, 49, 51, 55, 57, 58, 59, 60, 63, 65, 66, 69, 74, 75, 76, 77, 78, 79, 81, 82, 85, 90, 93, 100
]
PS C:\Users\Fadhil A\Documents\Kuliah\Project Coding\PROJECT-F\DASPRO - 10 ( NumPy, Array )>
```

3. Runtime Merge Sort



```
[Running] python -u "c:\Users\Fadhil A\Documents\Kuliah\Project Coding\PROJECT-F\DASPRO - 10 ( NumPy, Array )
\merge_sort.py"
Python was not found; run without arguments to install from the Microsoft Store, or disable this shortcut from
Settings > Manage App Execution Aliases.

[Done] exited with code=9009 in 1.258 seconds
```

Waktu Eksekusi Array dengan Merge Sort: 1.258 seconds.

Metode #2 – Bubble Sort

1. Code Bubble Sort

```python
'''
Nama : Fadhil Anwar Ahsani
Kelas : 1A - RPL
Nim : 2407136
'''
# Perbandingan Runtime Array yang di Sort dengan Metode yang ditentukan... (Bubble Sort)

def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        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

data = [7, 1, 36, 26, 63, 93, 55,
        16, 19, 38, 74, 65, 18, 59, 8,
        43, 24, 79, 49, 35, 23, 78, 51,
        2, 46, 28, 60, 76, 10, 85, 66,
        29, 82, 58, 69, 75, 48, 100, 5,
        32, 40, 33, 34, 90, 81, 42, 57,
        44, 41, 77]

array_bubble = bubble_sort(data)
print("Array setelah diurutkan:", array_bubble)
```

2. Output Bubble Sort

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Fadhil A\Documents\Kuliah\Project Coding\PROJECT-F\DASPRO - 10 ( NumPy, Array )> py bubble_sort.py
Array setelah diurutkan: [1, 2, 5, 7, 8, 10, 16, 18, 19, 23, 24, 26, 28, 29, 32, 33, 34, 35, 36, 38, 40, 41, 42, 43
, 44, 46, 48, 49, 51, 55, 57, 58, 59, 60, 63, 65, 66, 69, 74, 75, 76, 77, 78, 79, 81, 82, 85, 90, 93, 100]
PS C:\Users\Fadhil A\Documents\Kuliah\Project Coding\PROJECT-F\DASPRO - 10 ( NumPy, Array )>
```

3. Runtime Bubble Sort

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  Code

[Running] python -u "c:\Users\Fadhil A\Documents\Kuliah\Project Coding\PROJECT-F\DASPRO - 10 ( NumPy, Array )
\bubble_sort.py"
Python was not found; run without arguments to install from the Microsoft Store, or disable this shortcut from
Settings > Manage App Execution Aliases.

[Done] exited with code=9009 in 1.104 seconds
```

Waktu Eksekusi Array dengan Bubble Sort: 1.204 seconds.

Metode #3 – Quick Sort

1. Code Quick Sort

```python
'''
Nama : Fadhil Anwar Ahsani
Kelas : 1A - RPL
Nim : 2407136
'''
# Perbandingan Runtime Array yang di Sort dengan Metode yang ditentukan... (Quick Sort)

def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    # Menentukan pivot
    pivot = arr[-1]
    smaller = [x for x in arr[:-1] if x <= pivot]
    bigger = [x for x in arr[:-1] if x > pivot]
    return quick_sort(smaller) + [pivot] + quick_sort(bigger)

arr = [7, 1, 36, 26, 63, 93, 55,
       16, 19, 38, 74, 65, 18, 59,
        8, 43, 24, 79, 49, 35, 23,
        78, 51, 2, 46, 28, 60, 76,
        10, 85, 66, 29, 82, 58, 69,
        75, 48, 100, 5, 32, 40, 33
        , 34, 90, 81, 42, 57, 44, 41
        , 77]

array_quick = quick_sort(arr)
print("Array Setelah diurutkan:", array_quick)
```

2. Output Quick Sort

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Fadhil A\Documents\Kuliah\Project Coding\PROJECT-F\DASPRO - 10 ( NumPy, Array )> py quick_sort.py
Array Setelah diurutkan: [1, 2, 5, 7, 8, 10, 16, 18, 19, 23, 24, 26, 28, 29, 32, 33, 34, 35, 36, 38, 40, 41, 42, 43
, 44, 46, 48, 49, 51, 55, 57, 58, 59, 60, 63, 65, 66, 69, 74, 75, 76, 77, 78, 79, 81, 82, 85, 90, 93, 100]
```

3. Runtime Quick Sort

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  Code

[Running] python -u "c:\Users\Fadhil A\Documents\Kuliah\Project Coding\PROJECT-F\DASPRO - 10 ( NumPy, Array )
\quick_short.py"
Python was not found; run without arguments to install from the Microsoft Store, or disable this shortcut from
Settings > Manage App Execution Aliases.

[Done] exited with code=9009 in 0.35 seconds
```

Waktu Eksekusi Array dengan Quick Sort: 0.350 seconds. (Tercepat)

Kesimpulan

Dengan demikian perbandingan runtime code untuk setiap metode sorting array yang dipilih dari: Merge, Bubble, dan Quick mendapatkan hasil sebagai berikut:

1. Waktu Eksekusi Array dengan Merge Sort: 1.204 seconds
2. Waktu Eksekusi Array dengan Bubble Sort: 1.204 seconds.
3. Waktu Eksekusi Array dengan Quick Sort: 0.350 seconds. ***(Tercepat)***

Terimakasih…

Nama: Fadhil Anwar Ahsani

Kelas: 1A – RPL

NIM: 2407136