

Worksheet pertemuan 12 - 2

Algoritma dan Struktur Data

Quick Sort

NIM: 20523164

Nama: Fajrun Shubhi

A. Membuat Folder Untuk Menyimpan Hasil Praktikum

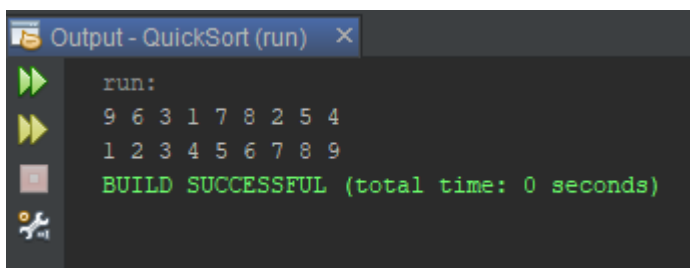
1. Siapkan folder kosong dengan nama menggunakan NIM masing-masing. Jika folder NIM pada pertemuan sebelumnya mau dimanfaatkan, jangan lupa pindahkan dulu isinya ke folder lain sebagai arsip.
2. Folder ini akan dijadikan tempat untuk menyimpan semua pdf dari worksheet ini beserta fail praktikum lainnya.

B. Mengimplementasikan Quick Sort

1. Buat prosedur Quick Sort Quick Sort seperti yang ada di slide
2. Kemudian jalankan Main method berikut

```
public static void main(String[] args) {  
    int[] arr = {9, 6, 3, 1, 7, 8, 2, 5, 4};  
  
    for(int i=0;i<arr.length;i++){  
        System.out.print(arr[i]+" ");  
    }  
    System.out.println("");  
  
    quickSort(arr, 0, arr.length-1);  
  
    for(int i=0;i<arr.length;i++){  
        System.out.print(arr[i]+" ");  
    }  
    System.out.println("");  
}
```

3. Jika implementasi Quick Sort sudah benar, maka output adalah array yang sudah terurut



```
Output - QuickSort (run) X  
run:  
9 6 3 1 7 8 2 5 4  
1 2 3 4 5 6 7 8 9  
BUILD SUCCESSFUL (total time: 0 seconds)
```

B. Membandingkan Quick Sort dengan Bubble Sort

1. Pada tutorial ini kita akan membandingkan kecepatan waktu eksekusi dua algoritme tersebut. Dengan array yang cukup besar akan tampak perbedaan dua algoritme tersebut.
2. Pertama-tama, siapkan prosedur bubbleSort seperti yang telah dibuat sebelumnya

```
static void bubbleSort(int array[]) {
    int n = 1;
    int ukuran = array.length;
    //iterasi
    while(n < ukuran){
        //proses pengecekan
        for (int i = 0; i < (ukuran-1); i++){
            if (array[i] > array[i+1]){
                //proses pertukaran posisi elemen
                int temp = array[i];
                array[i] = array[i+1];
                array[i+1] = temp;
            }
        }
        n++;
    }
}
```

3. Pada main program kita akan buat array dengan ukuran besar. Array ini berisi elemen dengan urutan terbalik [50000, 49999, 49998, ..., 1]

```
int[] arr = new int[50000];
for(int i=0;i<50000;i++){
    arr[i]=50000-i;
}
```

4. Untuk membandingkan waktu proses, kita tidak perlu menampilkan elemen array sebelum ataupun setelah proses pengurutan cukup jalankan perintah Quick Sort pada array tersebut. Kita tambahkan output sederhana dan fungsi untuk mengukur waktu eksekusi pengurutan:

```
long startTime = System.currentTimeMillis();

System.out.println("MULAI SORTING");
quickSort(arr, 0, arr.length-1);
System.out.println("SELESAI SORTING");

long stopTime = System.currentTimeMillis();
long duration = stopTime - startTime;
System.out.println("Waktu eksekusi algoritme: " + duration + " ms");
```

5. Pada output window di Netbeans akan tampak seberapa cepat program tereksekusi.
6. Salin output Anda ketika menjalankan Quick Sort pada array tersebut di kolom di bawah ini.

```
MULAI SORTING
SELESAI SORTING
Waktu eksekusi algoritme: 9 ms
BUILD SUCCESSFUL (total time: 0 seconds)
```

7. Ubah QuickSort dengan BubbleSort lalu salin output Anda ketika menjalankannya pada array tersebut di kolom di bawah ini. Bandingkan perbedaan kecepatannya.

```
MULAI SORTING
SELESAI SORTING
Waktu eksekusi algoritme: 3448 ms
BUILD SUCCESSFUL (total time: 3 seconds)
```

Sangat berbeda jauh sekali, pada quicksort Cuma membutuhkan waktu sekitar 9ms, sedangkan pada bubblesort membutuhkan waktu kurang lebih sekitar 3400ms

***Catatan**

- **Jangan lupa simpan juga fail worksheet ini (yang sudah diisi) sebagai fail pdf di folder NIM anda.**
- **Sertakan juga fail **Main.java** di dalam folder yang Anda gunakan**
- **Kompres folder ini sebagai fail ZIP kemudian kumpulkan di classroom atau ruang pengumpulan lain di kelas masing-masing.**