**MODUL 3**

**HARD**

Roman adalah seorang sport analyst, ia bekerja mengumpulkan data-data pemain NBA dan mengambil kesimpulan dari data-data tersebut.

Pada hari pertama ia bekerja, ia diberikan data 3 tim NBA.

|  |
| --- |
|  |

Kemudian, ia ditugaskan untuk mengurutkan tim tersebut. Roman merasa tugasnya terlalu mudah, untuk menambah tantangan, ia menggunakan quicksort untuk mengurutkannya berdasarkan jumlah kemenangan tim.

|  |
| --- |
|  |

Dikarenakan tugas yang dilakukannya terlalu mudah, ia meminta tugas lagi untuk dikerjakan. Untuk tugas selanjutnya, ia diberi data seluruh pemain dari setiap tim tersebut.

|  |
| --- |
|  |

Roman ditugaskan untuk mencari calon MVP NBA, karena tugas sebelumnya masih terlalu mudah bagi Roman, ia memutuskan menggunakan merge sort untuk mengurutkan seluruh pemain dari setiap tim berdasarkan PPG (Points Per Game).

|  |
| --- |
|  |

Setelah mengurutkan seluruh pemain dari seluruh tim, Roman merasa cara ini tidak efektif untuk mencari MVP, ia kemudian mencoba menggabungkan seluruh pemain ke dalam 1 list.

|  |
| --- |
|  |

Setelah membuat list seluruh pemain yang ada pada nba, Roman mengurutkan seluruh pemain berdasarkan PPG menggunakan merge sort.

|  |
| --- |
|  |

Karena kemarin Roman mempelajari quick sort, ia mencoba menggunakannya pada tugas ini

|  |
| --- |
|  |

Kemudian Roman ingin mencari pemain berdasarkan PPG-nya, ia menggunakan linear search dan binary search untuk mencari di daftar pemain yang telah diurutkan (bebas hasil merge atau quick).

|  |
| --- |
|  |

Karena Roman ingin mencari pemain terbaik sejauh ini, ia menggabungkan linear search dan binary search (dengan cara mencari PPG pemain menggunakan binary search, kemudian menggunakan linear search untuk iterasi ke-atas)untuk mencari kandidat MVP berdasarkan minimal PPG.

|  |
| --- |
|  |

**Notes :**

* Untuk menghitung waktu searching dan sorting, dapat menggunakan kode berikut atau algoritma lain :

|  |
| --- |
| long startTime = System.nanoTime();  //kode  long elapsedTime = System.nanoTime() - startTime;  System.out.println("Elapsed Time is " + (elapsedTime / 1000000.0) + " msec"); |

**Aturan :**

* Gunakan metode sorting merge sort dan selection sort.
* Gunakan metode searching linear search dan binary search.
* Lakukan sorting dan searching ke data tersebut dan bandingan efektifitas waktu setiap sorting dan searching. (Data yang dibandingkan adalah 100 data, 200 data dan 300 data).
* Contoh hasil perbandingan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Jumlah Data | Linear Search | Binary Search | Merge Sort | Quick Sort |
| 100 | waktu | waktu | waktu | waktu |
| 200 | waktu | waktu | waktu | waktu |
| 300 | waktu | waktu | waktu | waktu |

* Lakukan analisa dari hasil pengurutan dan pencarian yang telah dilakukan, bandingan hasil setiap algoritma sorting dan searching. Manakah algoritma yang paling efektif dan efisien sesuai dengan data yang ada untuk tiap sorting dan searching.