2023-2024-3 | Perancangan dan Analisis Algoritma | Tugas Kelompok | PAA – A

Nama	NRP
Mohammad Hanif Furqan Aufa Putra	5025221161
Rafli Raihan Pramudya	5025221266

Pada tugas kali ini, kita diminta untuk membuat program sorting dengan berbagai algoritma yang ada. Kemudian dibandingkan dengan grafik antara Size array dan waktu penyelesaian. Setelah itu akan dibandingkan hasil grafik kalian (Analisa empiris) dengan Analisa theoritis yang sudah kalian pelajari dan jelaskan kenapa beda atau sama!

Untuk pembuatan algortima kami menggunakan bahaca C++ karena efektif dan cepat. Berikut ini adalah algoritma yang akan kami pakai

- Insertion sort
- Merge sort
- Heap sort
- Quick sort
- Counting sort

A. Analisa Empiris

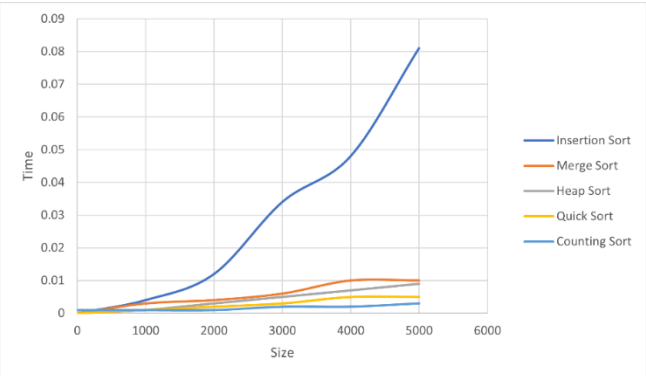
Untuk Code program yang kami pakai akan kami lampirkan pada Link Drive dibawah ini

Https://drive.google.com/file/d/1zarylkt3xwv38hn0jvjjm9pq-PPITW3m/view?Usp=sharing

Setelah program dijalankan, berikut adalah output yang kami dapatkan

```
PS E:\Semester 4\PAA> ./tugasruntime
Size    Insertion       Merge           Heap            Quick           Counting
1000    0.004017        0.002982        0.001002        0.001015        0.001000
2000    0.011987        0.004004        0.003019        0.002000        0.001001
3000    0.034024        0.005983        0.005000        0.003001        0.002001
4000    0.048029        0.010000        0.007004        0.004988        0.002014
5000    0.081017        0.010002        0.009021        0.005001        0.002982
```

Berikut ini adalah grafik yang dibuat berdasarkan output dari program sebelumnya



Dari hasil uji coba program sorting yang dibuat dalam bahasa C++, terlihat bahwa Counting Sort punya performa terbaik untuk dataset dengan rentang nilai yang sesuai, sementara Quick Sort juga sangat cepat untuk ukuran array besar, sesuai dengan kompleksitas waktu rata-rata O(n log n). Merge Sort dan Heap Sort menunjukkan kinerja yang stabil dan efisien, juga sesuai dengan kompleksitas O(n log n). Di sisi lain, Insertion Sort performanya paling buruk karena waktu eksekusinya meningkat drastis, sesuai dengan kompleksitas waktu O(n^2).

B. Analisa Theoritis

- Insertion Sort
  - Insertion Sort memiliki kompleksitas waktu O(n^2) dalam kasus terburuk, karena setiap elemen harus dibandingkan dengan elemen sebelumnya dan dipindahkan ke posisi yang benar dalam array.

  - Pada grafik diatas menunjukkan bahwa waktu eksekusi Insertion Sort meningkat secara drastis seiring dengan bertambahnya ukuran array. Hal ini sesuai dengan kompleksitas waktu yang diberikan, mengingat Insertion Sort melakukan banyak perbandingan dan pertukaran, terutama untuk array yang tidak terurut

- Merge Sort
  - Merge Sort memiliki kompleksitas waktu O (n log n), karena pembagian dan penggabungan array dilakukan secara rekursif.

  - Hasil pengukuran menunjukkan waktu eksekusi yang tidak konsisten, karena pada saat run antara size 4000 dan 5000 terjadi penurunan waktu eksekusi. Mungkin ini terjadi karena kebetulan program memberikan kasus best case untuk metode merge sort.

- Heap Sort
  - Heap Sort memiliki kompleksitas waktu O(n log n), karena proses pembuatan heap dan pengurutan dilakukan dalam logaritmik.

  - Data menunjukkan waktu eksekusi yang linear setiap bertambahnya size, Heap Sort juga menunjukkan kinerja yang konsisten dan efisien dalam mengurutkan array berukuran besar.

- Quick Sort
  - Quick Sort memiliki kompleksitas waktu O(n^2) dalam kasus terburuk, tetapi O(n log n) dalam kasus rata-rata karena pembagian dan pengurutan sub-array yang efektif.

  - Quick Sort umumnya menunjukkan kinerja yang sangat cepat, terbukti pada grafik ini menjadi urutan ke dua tercepat dalam mengurutkan array. Namun, variasi waktu eksekusi dapat terjadi tergantung pada pemilihan pivot, yang bisa membuat performa mendekati dalam skenario terburuk.

- Counting Sort
  - Counting Sort memiliki kompleksitas waktu O(n + k), di mana k adalah rentang nilai dalam array. Algoritma ini sangat efisien untuk dataset dengan rentang nilai yang kecil atau relatif terhadap jumlah elemen.

  - Counting Sort menunjukkan waktu eksekusi yang sangat cepat untuk ukuran array yang relatif kecil dengan rentang nilai yang kecil. Dalam grafik yang dibuat algoritma ini menempati peringkat 1 dalam mengurutkan array. Namun, kita tidak tahu apa yang terjadi jika pada size yang lebih besar. Karena efisiensi Counting Sort berkurang jika rentang nilai (k) besar, karena memori dan waktu yang dibutuhkan untuk menghitung frekuensi elemen meningkat.

C. Perbandingan antara Analisa Empiris dan Theoritis

Dari analisa teoretis dan empiris, kita bisa lihat ada kesamaan yang signifikan:

- Insertion Sort: Menunjukkan peningkatan waktu eksekusi yang drastis seiring dengan bertambahnya ukuran array (O(n^2)).
- Merge Sort dan Heap Sort: Dengan kompleksitas O(n log n), menunjukkan kinerja yang stabil dan efisien dalam pengujian, meskipun terdapat sedikit variasi pada hasil Merge Sort karena mungkin best case.
- Quick Sort: Sangat cepat dan konsisten dengan kompleksitas rata-rata O(n log n), menjadi salah satu yang tercepat dalam pengujian ini, walaupun performanya dapat bervariasi tergantung pemilihan pivot.
- Counting Sort: Muncul sebagai yang tercepat untuk dataset dengan rentang nilai kecil (O(n + k)), namun efisiensinya dapat menurun jika rentang nilai sangat besar.

Secara keseluruhan, hasil empiris sejalan dengan analisa teoretis, menunjukkan bagaimana kompleksitas waktu mempengaruhi performa berbagai algoritma sorting pada ukuran array yang berbeda. Hasil pengujian ini mengonfirmasi teori yang telah dipelajari, menunjukkan bahwa algoritma dengan kompleksitas O(n log n) dan O(n + k) cenderung memiliki performa lebih baik dibandingkan dengan kompleksitas O(n^2), terutama untuk ukuran data yang besar.