Laporan *Sorting Algorithms*

Qornain Aji

21/481767/TK/53170

Pada laporan tugas kali ini, kita akan membahas mengenai 3 algoritma *sorting* yang telah kita pelajari pada pertemuan sebelumnya serta yang terdapat pada materi di ppt minggu lalu. Algoritma *sorting* yang kita pelajari adalah *bubble sort, selection sort, the insertion sort.* Sekarang kita akan menjelaskan satu persatu ke-3 algoritma tersebut dari segi langkah-langkah penyelesaian dan waktu yang diperlukan saat menggunakan algoritma tersebut. Informasi mengenai ke-3 algoritma *sorting* kami dapatkan dari buku [1] *Problem Solving with Algorithms and Data Structures* karya Brad Miller dan David Ranum.

- ***Bubble Sort***

Algoritma *bubble sort* menggunakan metode yakni, membandingkan item yang bersebelahan dan menukarkan urutan item yang bersebelahan tersebut ketika tidak sesuai urutan. Setelah item yang bersebelahan dibandingkan, algoritma tersebut akan melihat item yang berikutnya dan akan dilakukan perbandingan item yang bersebelahan sama seperti proses yang sebelumnya. Algoritma *bubble sort* membuat beberapa perulangan hingga semua item menjadi urut.

Untuk penjelasan detailnya sebagai berikut. Pada perulangan pertama, jika terdapat n buah item yang akan dilakukan perbandingan, maka terdapat n-1 buah proses membandingkan item yang dilakukan oleh algoritma pada perulangan pertama. Pada kondisi ini, item dengan nilai terbesar sudah dapat dipastikan berada pada posisi urutan yang sesuai.

Pada perulangan yang kedua, proses membandingkan item hanya bersisa n-2 dan item dengan nilai kedua terbesar sudah dipastikan berada pada posisi urutan yang sesuai yakni disebelah item dengan nilai terbesar. Proses dilakukan hingga semua item berada pada posisi yang sesuai dengan total perulangan yang diperlukan sebanyak n-1 perulangan.

Algoritma *bubble sort* membutuhkan waktu perbandingan dan penukaran dengan skenario terburuk sebesar $O(n^2)$. Skenario terbaik adalah semua item sudah pada urutan yang benar sehingga tidak memerlukan penukaran posisi urutan antar item.

- ***Selection Sort***

Algoritma selanjutnya adalah *selection sort*. Algoritma *selection sort* merupakan perbaikan dari *bubble sort* dimana mengurangi jumlah pertukaran antar item pada setiap perulangannya. *Selection sort* bekerja dengan mencari nilai terbesar dari semua item pada tiap perulangannya yang lalu akan dipindahkan ke urutan teratas sesuai dengan posisinya.

Jumlah perbandingan yang dibutuhkan pada *selection sort* tetap sama dengan *bubble sort* sebesar $O(n^2)$. Namun, jumlah pertukaran yang dilakukan oleh algoritma *selection sort* menurun drastis jika dibandingkan dengan *bubble sort*. Hal ini disebabkan karena pada setiap perulangannya, algoritma *selection sort* hanya akan melakukan sekali pertukaran saja dimana

bubble sort akan melakukan beberapa kali pertukaran pada setiap perulangannya. Dengan penurunan jumlah pertukaran ini, algoritma *selection sort* akan berjalan lebih cepat dalam studi tolak ukur algoritma.

- ***The Insertion Sort***

Algoritma *insertion sort* menggunakan prosedur yang agak berbeda dari algoritma-algoritma yang lain. Prosedur dimulai dengan memulai perulangan pertama. Ketika ditemukan item yang bersebelahan tidak sesuai urutan besarnya nilai, item yang bernilai lebih kecil akan ditukarkan dengan item di sebelahnya yang bernilai lebih besar. Pertukaran tidak hanya dilakukan sekali saja. Tetapi, berlanjut hingga item yang bernilai lebih kecil tersebut berpindah tempat hingga ke tempat dengan urutan yang sesuai. Perulangan berlanjut untuk perulangan yang kedua. Proses tersebut diulang hingga semua item berada pada urutan yang sesuai.

Waktu perbandingan dan waktu penukaran yang dibutuhkan dalam *insertion sort* dengan skenario terburuk adalah $O(n^2)$. Namun untuk skenario terbaiknya item telah berada pada urutan yang sesuai sehingga tidak diperlukan penukaran item.

- ***Merge Sort***

Algoritma ini menggunakan metode yang sangat berbeda. Algoritma ini memisahkan item-item menjadi 2 bagian. Lalu tiap bagian-bagian dipecah lagi menjadi 2 bagian sehingga tinggal hanya menyisakan 1 item. Setelah bersisa 1 bagian, maka pembandingan antar item di sebelahnya pada tiap-tiap bagian dilakukan dan diurutkan. Setelah urut, bagian yang awalnya terpisah disatukan kembali dan dilakukan pembandingan ulang dan diurutkan lagi hingga menjadi 1 bagian utuh seperti semula.

Algoritma tersebut sangatlah cepat dengan hanya membutuhkan waktu O(n log n). Namun, kelemahan dari algoritma tersebut adalah membutuhkan space alokasi memori yang lebih banyak daripada algoritma-algoritma sebelumnya.

- **Hasil pembahasan**

Setelah mengetahui kemampuan dari tiap-tiap algoritma, kita dapat menilai efisiensi waktu yang dibutuhkan dari tiap-tiap algoritma. Kita tahu bahwa algoritma dengan waktu paling efisien adalah *merge sort*. Kita juga setuju bahwa sulit menemukan dalam praktiknya kita menjumpai skenario terbaik dalam mengurutkan item-item. Karena dunia yang modern ini, kita mementingkan kecepatan, maka untuk waktu tercepat, *merge sort* memiliki waktu sebesar O(nlogn). Algoritma *bubble sort* dan *insertion sort* semuanya memiliki waktu penukaran dengan skenario terburuk sebesar $O(n^2)$. Algoritma *selection sort* memiliki waktu penukaran sebesar O(n). Hal tersebut tergolong tidak efisien karena algoritma *merge sort* memiliki waktu total sebesar O(nlogn), jauh lebih cepat dibandingkan ketiga algoritma sebelumnya. Oleh karena itu, algoritma *merge sort* memiliki kinerja yang lebih baik dibandingkan ketiga algoritma yang lainnya.

Daftar Pustaka

[1] B. Miller, D. Ranum, Problem Solving with Algorithms and Data Structures, vol. 3, September 2013.