Aplikasi Struktur Data Nonlinear: Heapsort

Tim Olimpiade Komputer Indonesia

Pendahuluan

Melalui dokumen ini, kalian akan:

- Menggunakan *heap* untuk *heapsort*.
- Mengamati bagaimana struktur data diaplikasikan dalam penyelesaian masalah.

Kilas Balik

Ingat dengan *selection sort*?

Berikut adalah algoritmanya:

- Pilih elemen terkecil, tempatkan di paling awal data.
- Ulangi hingga seluruh data terurut menaik.

Kilas Balik (lanj.)

- Pencarian elemen terkecil pada *selection sort* dilakukan dengan *linear search*.
- Sekarang kita telah mengetahui strategi pencarian elemen terbesar/terkecil dari sekumpulan data secara efisien.
- Bagian pencarian elemen terkecil pada *selection sort* dapat dioptimisasi menggunakan *heap*, dan algoritma pengurutan ini dinamakan **heapsort**.

Heapsort

Misalkan kita memiliki *array* A berisi N elemen yang hendak diurutkan menaik:

1. Muat seluruh elemen *array* A pada **min-heap**, menggunakan MAKEHEAP.
2. Untuk i dari 0 sampai $N - 1$, lakukan:
   2.1 Dapatkan elemen terkecil dari *heap* dengan POP.
   2.2 Tempatkan elemen ini pada $A[i]$.

Analisis Heapsort

- Operasi MAKEHEAP bekerja dalam $O(N)$.
- Kemudian dilakukan N kali POP, sehingga kompleksitasnya $O(N \log N)$.
- Jadi kompleksitas akhir *heapsort* adalah $O(N \log N)$.

Keuntungan Heapsort

- *Heapsort* memiliki keuntungan yang sama dengan *selection sort*, yaitu dapat melakukan *partial sort*.
- Artinya, bila Anda hanya membutuhkan K elemen terkecil, Anda dapat berhenti setelah melakukan POP sebanyak K kali.
- Kompleksitasnya menjadi $O(N + K \log N)$, yang lebih efisien ketika K jauh lebih kecil dari N.

Penutup

- Kini Anda telah memahami 3 pengurutan dengan kompleksitas $O(N \log N)$, setelah *quicksort* dan *merge sort*.
- Meskipun demikian, *heapsort* lebih jarang digunakan untuk pengurutan N data, karena implementasinya yang cukup rumit.
- *Heapsort* hanya digunakan untuk kebutuhan tertentu, seperti *partial sort*.