BAB V
SORTING (PENGURUTAN) INTERNAL

6. 4 Heap

Heap adalah sebuah binary tree dengan ketentuan sebagai berikut :

- Tree harus complete binary tree
  - Semua level tree mempunyai simpul maksimum kecuali pada level terakhir.
  - Pada level terakhir, node tersusun dari kiri ke kanan tanpa ada yang dilewati.
- Perbandingan nilai suatu node dengan nilai node child-nya mempunyai ketentuan berdasarkan jenis heap, diantaranya :
  - Max Heap mempunyai ketentuan bahwa nilai suatu node lebih besar atau sama dengan (>=) dari nilai childnya.
  - Min Heap mempunyai ketentuan bahwa nilai suatu node lebih kecil atau sama dengan (<=) dari nilai childnya.

6.5 Quick Sort

Motivasi metoda ini adalah meletakkan suatu elemen, katakanlah elemen ke 1, pada posisi yang sudah tetap pada deretan. Sewaktu proses ini berlangsung, sekaligus akan dilakukan penyusunan sehingga pada akhirnya semua elemen sebelah kiri mempunyai nilai key <, dan semua elemen sebelah kanan mempunyai ke yang > key elemen ke 1 tersebut.

Ilustrasi :

Keadaan awal

20	11	5	23	17	9	21	19	4	12
1	2	3	4	5	6	7	8	9	10

Ambil nilai I=2 dan j=10. Mulai dari i ke arah kanan cari elemen yang > elemen ke 1, berhasil ditemui ialah elemen ke 4, atau 23. Nilai i sekarang = 4. Mulai dari j ke arah kiri cari elemen yang < elemen ke 1, berhasil ditemui ialah elemen ke 10, atau 12. Nilai j tetap = 10 Tukarkan elemen ke i dan ke j tersebut

.

20	11	5	12	17	9	21	19	4	23
1	2	3	4	5	6	7	8	9	10
			i						J

Ulangi proses, maka diperoleh I=7 dan j=9 tukarkan.

20	11	5	12	17	9	4	19	21	23
1	2	3	4	5	6	7	8	9	10
						i		J	

Ulangi proses, pencariani i berhasil pada nilai i=9. Sedangkan pencarian j berhenti pada nilai j=8.

20	11	5	12	17	9	4	19	21	23
1	2	3	4	5	6	7	8	9	10
							J	i	

Karena i sudah >=j, maka lakukan penukaran elemen ke j dengan elemen ke 1.

19	11	5	12	17	9	4	20	21	23
1	2	3	4	5	6	7	8	9	10

Partisi kiri Partisi kanan

J I

Perhatikan bahwa elemen ke 1 (20) sudah mendapat posisi tetap. Semua elemen sebelah kirinya < 20 dan sebelah kanannya > 20. Perhatikan pula bahwa sekarang deretan nilai “terbelah” menjadi 2 bagian., bagian “kiri” dan bagian “kanan”. Proses selanjutnya ialah mengulang proses semula, mula-mula terhadap deretan kiri (sampai tuntas), kemudian terhadap deretan kanan.

Proses terhadap deretan kiri :

19	11	5	12	17	9	4
1	2	3	4	5	6	7

Ambil nilai i=2 dan j=7. Mulai dari i kearah kanan cari elemen yang > elemen ke 1, **pencarian gagal**, dihentikan dengan nilai i= 7. Mulai dari j ke arah kiri cari elemen yang < elemen ke 1, berhasil ditemui ialah elemen ke 7 atau 4. Nilai j tetap = 7.

19	11	5	12	17	9	4
1	2	3	4	5	6	7
						i j

Karena I sudah >= j, tukarkan elemen ke j dengan elemen ke 1.

4	11	5	12	17	9	19
1	2	3	4	5	6	7
						i j

Perhatikan bahwa 19 sudah mendapat posisi finalnya. Tinggal bagian kiri (mulai dari elemen ke 1 s/d 6) yang akan diproses lebih lanjut. Demikian seterusnya sehingga seluruh bagian kiri akhirnya akan terurut.

Prosesnya berturut turut :

4	11	5	12	17	9
	2	3	4	5	6
j	i				

Pencarian j gagal, tidak ada pertukaran.

11	5	12	17	9
2	3	4	5	6
		i		J

Tukar elemen ke j dengan ke i

11	5	9	17	12
2	3	4	5	6
	j		I	

Tukar elemen ke j dengan ke 1

11	9	17	12
3	4	5	6
	j	i	

Tukar elemen ke j dengan ke 1

11	17	12
4	5	6
j	i	

Pencarian j gagal, tidak ada pertukaran

17	12
5	6
	I j

12	17
5	6

Selesai

4	5	9	11	12	17
1	2	3	4	5	6

Lakukan Proses terhadap deretan kanan

Perhatikan bahwa metoda quicksort ini bersifat rekursif, karena proses pembuatan bagian (partisi) dilakukan kembali terhadap bagian yang terbentuk.

6.6 Memilih diantara Metode Sort

Setiap metode sort mempunyai beberapa keuntungan dan kerugian yang akan dipertimbangkan dalam meilih diantara metode sort-metode sort tersebut untuk tugas pemrograman tertentu. Umumnya metode Insertion sort (sisip) mungkin sangat bermanfaat pada saat data yang harus diurutkan sebgai data yang dimasukkan, jika diaplikasikan pada daftar yang sudah ada dalam memori, penambahn memori untuk membuat daftar baru yang diperlukan.

Bubble dan metode seleksi (selection sort) mempunyai keuntungan sederhana dan mudah diterapkan. Bubble sort merupakan metode yang dapat berakhir pada saat daftar yang sudah diurtkan diperoleh, yang mempunyai keuntungan jika data yang diurutkan mempunyai peluang tinggi sudah dalam keadaan urut atau hampir urut ketika pengurutan dikerjakan.

Analisi matematis terhadap heap sort cukup rumit. Hasilnya memperlihatkan bahwa performance terbaik, rata-rata dan terburuk dari algoritma ini = O(n log n) – O(n). Sehingga dapat disimpulkan bahwa metode ini adalah metode yang mempunyai performance cukup baik, bahkan dalam kasus terburuknya.

Quick sort di harapakan mempunyai proses yang terbaik, membuat mmetode ini dipilih untuk jumlah data yang besar, kerugiannya bahwa metode ini menerapkan paling sederhana dalam rekursif dan tidak berakhir secara automatis jika daftar yang diurutkan sudah diperoleh, pada beberapa tingkat lanjutan dalam proses sorting (dalam rekursifnya).