

ABSTRAK

Map adalah struktur data umum dan bagian yang penting dari pemrograman komputer. *Map* memungkinkan kita untuk menyimpan data berupa pasangan kunci-nilai, sehingga pengguna dapat memasukkan data ke dalam *map* dengan menentukan pasangan kunci-nilai dan kemudian dapat mengambil nilai yang sesuai dengan kunci tertentu dari dalam *map*. Terdapat bahasa pemrograman yang berbeda mendukung *map* dalam berbagai cara, dalam hal ini bahasa pemrograman Java memiliki implementasi tersendiri dari *map* seperti *TreeMap*. *TreeMap* menyimpan semua pasangan kunci-nilai ke dalam struktur data pohon yang disebut pohon *red-black* dan struktur tersebut diurutkan oleh kunci. *TreeMap* menyimpan setiap pasangan kunci-nilai ke setiap *node* di pohon sehingga jumlah *node* dan pasangan kunci-nilai bernilai sama. Ada cara yang berbeda untuk menempatkan pasangan kunci-nilai dalam *node* yang setiap *node* di pohon menyimpan lebih dari satu pasangan kunci-nilai sehingga dapat mengurangi jumlah *node* dan dengan menggunakan struktur data pohon lain yang *right-threaded AVL tree*. Terakhir adalah memeriksa kinerja kecepatan eksekusi dan penggunaan memori menggunakan aplikasi sederhana.

Kata Kunci : Map, *TreeMap*, AVL tree

ABSTRACT

Map is common data structures and important part of computer programming. A map allows us to store data as key-value pairs, so user can put data into the map by specifying a key-value pair and then can retrieve the value corresponding to a particular key from the map. There are different programming languages supported map in a different ways, in this case the Java Programming language had its own implementation of maps such as TreeMap. TreeMap stores all key-value pairs into a tree data structure called red-black tree and it is sorted by keys. TreeMap stores each key-value pairs into a single node in the tree thus the number of nodes and key-value pairs are equals. There are different way to put the key-value pairs in the nodes which is each node in the tree hold more than one key-value pairs so it can reduce the number of nodes and by using other tree data structures which is right-threaded AVL tree. Finally check the performance by the execution speed and memory usage by a simple application.

Keyword : Map, TreeMap, AVL tree

DAFTAR ISI

LEMBAR PENGESAHAN	ii
PERNYATAAN ORISINALITAS LAPORAN PENELITIAN.....	iii
PERNYATAAN PUBLIKASI LAPORAN PENELITIAN	iv
PRAKATA.....	v
ABSTRAK.....	vii
ABSTRACT.....	viii
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah.....	2
1.3 Tujuan Penelitian	3
1.4 Batasan Masalah	3
1.5 Sistematika Pembahasan	4
BAB II LANDASAN TEORI.....	6
2.1 Struktur Data.....	6
2.2 Tipe Data Abstrak.....	6
2.3 Tipe Data Abstrak Map.....	7
2.4 Array	8
2.5 Tree	9
2.6 Binary Search Tree.....	10
2.7 Struktur Pohon AVL	13
2.7.1 Deklarasi Node AVL Tree	15
2.7.2 Insert Node AVL Tree	15
2.7.3 Delete Node AVL Tree	20
2.7.4 Rebalance Node AVL Tree.....	25

2.7.5	Performansi AVL Tree	27
2.8	Struktur Pohon Right-Threaded Binary	28
2.8.1	Deklarasi Node RTBST	28
2.8.2	Penambahan Node RTBST	29
2.8.3	Penghapusan Node RTBST	30
2.8.4	Pencarian Node RTBST	33
2.8.5	Traversal RTBST	34
2.9	Right-Threaded AVL Tree	35
2.10	TreeMap	37
2.11	Bahasa Pemrograman Java	38
2.11.1	Kelas dan Objek	38
2.11.2	Tipe data	39
2.11.3	Penggunaan Array pada Java	39
2.11.4	Bitwise Operator	40
2.11.5	Tipe Generic	43
2.11.6	Menghitung Waktu Proses pada Java	43
2.11.7	Menghitung Memori pada Java	44
2.11.8	Javadoc	44
2.10.8	JUnit	45
BAB III	ANALISIS DAN DESAIN	46
3.1	Analisis	46
3.1.1	Analisis Interface Map	46
3.1.2	Analisis Interface Map.Entry	48
3.1.3	Analisis Interface SortedMap	48
3.1.4	Analisis Interface NavigableMap	49
3.1.5	Analisis Struktur Data Red-Black Tree	50

3.2	Gambaran Keseluruhan.....	52
3.2.1	Persyaratan Antarmuka Eksternal.....	53
3.2.2	Antarmuka dengan Pengguna	53
3.2.3	Antarmuka Perangkat Keras	53
3.2.4	Antarmuka Perangkat Lunak	54
3.2.5	Fitur-fitur Perangkat Lunak	54
3.2	Desain Perangkat Lunak	60
3.2.5	Perancangan Tipe Data Abstrak Map	60
3.2.6	Pemodelan Perangkat Lunak.....	61
3.2.7	Desain Penyimpanan Data	79
3.2.8	Algoritma	84
4.2.7	Desain Antarmuka	106
BAB IV PENGEMBANGAN PERANGKAT LUNAK		109
4.1	Implementasi Package	109
4.2	Implementasi Class	109
4.3	Implementasi Method	110
4.3.1	Implementasi Method pada Kelas IntKeySmallMap	110
4.3.2	Implementasi Method pada Kelas IntKeyArrayMap	112
4.3.3	Implementasi Method pada Kelas RightThreadedAVL	116
4.3.4	Implementasi Method pada Kelas IntKeyTreeMap	123
4.4	Implementasi Antarmuka Program Simulasi	130
4.4.1	Form Kelola Right-Threaded AVL Tree	130
4.4.2	Form Kelola Data ADT Map	130
4.4.3	Form View ADT Map.....	131
4.5	Antarmuka Program Testing.....	132
4.6	Antarmuka Javadoc.....	133

BAB V TESTING DAN EVALUASI SISTEM	134
5.1 Rencana Pengujian	134
5.2 Pelaksanaan Pengujian	134
5.2.1 Unit Testing	135
5.2.2 Black Box	140
5.2.3 Uji Performansi Map	144
5.3 Analisis Hasil Uji Performansi Map	153
5.4 Perbandingan Struktur AVL Tree dan Red-Black Tree	155
BAB VI KESIMPULAN DAN SARAN	156
6.1 Kesimpulan	156
6.2 Saran	157
DAFTAR PUSTAKA	159
RIWAYAT HIDUP	161

DAFTAR GAMBAR

Gambar 2.1 Ilustrasi dari sebuah <i>map</i> . <i>Key (label)</i> mempunyai sebuah <i>value (diskette)</i> dan dibungkus ke dalam <i>entry (labeled diskette)</i> lalu disusun.....	7
Gambar 2.2 Ilustrasi sebuah <i>array</i> (Zakaria, 2006)	9
Gambar 2.3 Struktur data pohon (Lafore, 2003).....	9
Gambar 2.4 Istilah-istilah pada struktur data pohon	10
Gambar 2.5 Contoh pohon biner (Rosa, 2010)	11
Gambar 2.6 <i>Binary search tree</i> menggunakan bilangan.....	12
Gambar 2.7 Representasi Elemen Pohon Biner	12
Gambar 2.8 Representasi pohon biner (Rosa, 2010)	13
Gambar 2.9 Deklarasi <i>node</i> pada <i>binary search tree</i>	13
Gambar 2.10 (a) <i>AVL tree</i> ; (b) bukan <i>AVL tree</i> (simpul tidak seimbang	14
Gambar 2.11 <i>AVL tree</i> dengan simbol bantu (Sanjaya, 2005).....	15
Gambar 2.12 Deklarasi <i>node</i> pada <i>AVL tree</i> link[0] merupakan anak kiri	15
Gambar 2.13 Penambahan <i>node n</i> pada anak kiri dari <i>node p</i> membuat <i>balance factor node p</i> dan <i>parent</i> dari <i>node p</i>	16
Gambar 2.14 Deklarasi pointer dan algoritma pencarian lokasi untuk.....	17
Gambar 2.15 Penempatan <i>pointer</i> tambahan t dan s pada	17
Gambar 2.16 Proses <i>update balance factor</i> dimulai dari s turun.....	18
Gambar 2.17 Algoritma <i>insert</i> pada <i>AVL tree</i> (Walker, 2008)	18
Gambar 2.18 Algoritma metode <i>insert_balance</i> (Walker, 2008)	19
Gambar 2.19 Ilustrasi <i>insertion</i> pada <i>AVL tree</i> (Walker, 2008)	20
Gambar 2.20 Deklarasi pointer dan algoritma pencarian lokasi <i>node</i> yang akan dihapus pada <i>AVL tree</i> (Walker, 2008).....	21
Gambar 2.21 Algoritma <i>delete</i> pada <i>AVL tree</i> (Walker, 2008).....	22
Gambar 2.22 Algoritma penghapusan <i>node</i> pada <i>AVL tree</i> (Walker, 2008)	22
Gambar 2.23 Algoritma metode <i>remove_balance</i> (Walker, 2008).....	23
Gambar 2.24 Ilustrasi <i>deletion</i> pada <i>AVL tree</i> (Walker, 2008).....	24
Gambar 2.25 <i>Single rotate</i> arah kanan pada <i>AVL tree</i>	25
Gambar 2.26 <i>Single rotate</i> arah kiri pada <i>AVL tree</i>	25
Gambar 2.27 Algoritma <i>single rotate</i> pada <i>AVL tree</i>	25

Gambar 2.28 <i>Double rotate</i> arah kanan pada AVL tree	26
Gambar 2.29 <i>Double rotate</i> arah kiri pada AVL tree	26
Gambar 2.30 Algoritma <i>double rotate</i> pada AVL tree	26
Gambar 2.31 Algoritma metode <i>adjust_balancee</i>	27
Gambar 2.32 <i>Right-threaded binary tree</i> (Walker, 2008)	28
Gambar 2.33 <i>Right-threaded binary tree</i> (Walker, 2008)	28
Gambar 2.34 Proses <i>insert</i> pada <i>right-threaded binary</i>	29
Gambar 2.35 Proses <i>remove</i> situasi ke-1 pada <i>right-threaded</i>	30
Gambar 2.36 Proses <i>remove</i> situasi ke-2 pada <i>right-threaded</i>	31
Gambar 2.37 Proses <i>remove</i> situasi ke-3 pada <i>right-threaded</i>	32
Gambar 2.38 Proses <i>remove</i> situasi ke-4 pada <i>right-threaded</i>	33
Gambar 2.39 Algoritma pencarian pada <i>right-threaded</i>	34
Gambar 2.40 Algoritma traversal pada <i>right-threaded</i>	34
Gambar 2.41 Deklarasi <i>node</i> pada <i>right-threaded AVL tree</i>	35
Gambar 2.42 <i>Single rotation</i> ke arah kanan pada	35
Gambar 2.43 <i>Single rotation</i> ke arah kiri pada	36
Gambar 2.44 <i>Double rotation</i> ke arah kanan pada	36
Gambar 2.45 <i>Double rotation</i> ke arah kiri pada	36
Gambar 2.46 Metode untuk menghitung waktu dalam	43
Gambar 2.47 Algoritma untuk menghitung waktu dalam Java (Knudsen, 2005). ..	44
Gambar 2.48 Metode untuk menghitung memori	44
Gambar 2.49 Algoritma untuk menghitung memori	44
Gambar 3.50 Struktur <i>Interface</i> pada <i>Java Collections Framework</i>	46
Gambar 3.51 Kelas-kelas implementasi Map di dalam <i>Java Collections Framework</i> (Naftalin, 2007)	37
Gambar 3.52 <i>Red-black tree</i> dengan jumlah 14 elemen dan maksimum tinggi 5 (Collins, 2011)	51
Gambar 3.53 <i>Use Case</i> diagram program simulasi	62
Gambar 3.54 Aktivitas pengolahan data <i>right-threaded AVL tree</i>	63
Gambar 3.55 Aktivitas tambah data <i>right-threaded AVL tree</i>	64
Gambar 3.56 Aktivitas hapus data AVL tree	65
Gambar 3.57 Aktivitas cari data AVL tree	65

Gambar 3.58 Aktivitas hapus semua data <i>right-threaded</i> AVL tree	66
Gambar 3.59 Aktivitas melihat data <i>right-threaded</i> AVL tree	66
Gambar 3.60 Aktivitas pengolahan data	67
Gambar 3.61 Aktivitas tambah data	68
Gambar 3.62 Aktivitas hapus data	69
Gambar 3.63 Aktivitas cari data	70
Gambar 3.64 Aktivitas ubah data	71
Gambar 3.65 Aktivitas melihat struktur <i>tree</i> ADT <i>map</i>	72
Gambar 3.66 Aktivitas <i>upload</i> data <i>file</i>	73
Gambar 3.67 <i>Class</i> diagram IntKeySmallMap	74
Gambar 3.68 <i>Class</i> diagram IntKeyArrayMap	75
Gambar 3.69 <i>Class</i> diagram RightThreadedAVL	76
Gambar 3.70 <i>Class</i> diagram IntKeyAVLMap	77
Gambar 3.71 <i>Class</i> diagram MiniBench	78
Gambar 3.72 <i>Class</i> diagram data uji coba	79
Gambar 3.73 Representasi penyimpanan pada kelas	80
Gambar 3.74 Representasi penyimpanan pada kelas IntKeyArrayMap	81
Gambar 3.75 Penyimpanan pada kelas RightThreadedAVL	82
Gambar 3.76 Penyimpanan pada kelas IntKeyTreeMap	83
Gambar 3.77 Representasi proses masukkan <i>key</i> dan <i>value</i> dalam kelas IntKeySmallMap	84
Gambar 3.78 Proses input <i>key</i> dan <i>value</i> dalam kelas IntKeyArrayMap	87
Gambar 3.79 Proses penyimpanan <i>key</i> pada kelas RightThreadedAVL	92
Gambar 3.80 Proses input <i>key</i> dan <i>value</i> dalam kelas IntKeyTreeMap	100
Gambar 3.81 Menghitung <i>key</i> dalam suatu NodeEntry	106
Gambar 3.82 <i>Form</i> kelola <i>right-threaded</i> AVL <i>tree</i>	107
Gambar 3.83 <i>Form</i> kelola <i>customer</i>	107
Gambar 3.84 <i>Form</i> view ADT <i>map</i>	108
Gambar 4.85 Tampilan <i>form</i> kelola <i>right-threaded</i> AVL <i>tree</i>	130
Gambar 4.86 Tampilan <i>form</i> kelola data ADT <i>map</i>	131
Gambar 4.87 Tampilan <i>form</i> kelola ADT <i>map</i>	131
Gambar 4.88 Tampilan program tester menu utama	132

Gambar 4.89 Tampilan program tester menu pilih tipe <i>generic</i>	132
Gambar 4.90 Tampilan output hasil pengetesan	132
Gambar 4.91 Tampilan output hasil pengetesan di simpan ke dalam sebuah <i>file</i>	133
Gambar 4.92 Tampilan Javadoc kelas <i>IntKeyTreeMap</i>	133
Gambar 5.93 <i>Sample</i> data <i>Integer-Customer</i> dengan <i>random key</i>	145

DAFTAR TABEL

Tabel 2.1 Contoh penggunaan dan hasil operasi pada <i>map</i> dengan <i>key</i> bertipe <i>integer</i> dan <i>value</i> bertipe <i>single character</i> (Goodrich, 2010).....	8
Tabel 2.2 Performansi kompleksitas kecepatan operasi pada AVL tree (Goodrich, 2010)	27
Tabel 2.3 Konstruktor di dalam kelas TreeMap (Schildt, 2007)	38
Tabel 2.4 Tipe data bilangan bulat pada Java (Schildt, 2007)	40
Tabel 2.5 Bitwise operator di java (Schildt, 2007)	40
Tabel 2.6 Tabel Logical Operator (Schildt, 2007)	41
Tabel 2.7 Metode <i>Assertion</i> pada JUnit (Minella, 2008).....	45
Tabel 3.8 Metode-metode di dalam <i>Interface Map</i> (Schildt, 2007).....	47
Tabel 3.9 Metode-metode di dalam <i>Interface Map.Entry</i> (Schildt, 2007).....	48
Tabel 3.10 Metode-metode di dalam <i>Interface SortedMap</i> (Schildt, 2007)	49
Tabel 3.11 Metode di dalam <i>Interface NavigableMap</i> (Schildt, 2007)	49
Tabel 3.12 Metode di dalam <i>Interface NavigableMap</i> (lanjutan).....	50
Tabel 3.13 Performansi kompleksitas kecepatan operasi pada <i>red-black tree</i> (Goodrich, 2010).....	51
Tabel 4.14 Implementasi <i>package</i>	109
Tabel 4.15 Implementasi <i>class</i>	109
Tabel 4.16 Fungsi-fungsi pendukung pada <i>IntKetSmallMap</i>	112
Tabel 4.17 Fungsi-fungsi pendukung pada <i>IntKeyArrayMap</i>	116
Tabel 4.18 Fungsi-fungsi pendukung pada <i>RightThreadedAVL</i>	123
Tabel 4.19 Fungsi-fungsi pendukung pada <i>IntKeyTreeMap</i>	128
Tabel 5.20 Pengujian halaman <i>right-threaded AVL</i>	141
Tabel 5.21 Pengujian halaman data ADT Map.....	142
Tabel 5.22 Pengujian halaman ADT Map Tree	143
Tabel 5.23 Pengujian program test performansi <i>map</i>	143
Tabel 5.24 Pengujian performansi penggunaan memori pada dua buah <i>map</i>	146
Tabel 5.25 Pengujian performansi penggunaan memori pada dua buah <i>map</i>	147
Tabel 5.26 Pengujian performansi penggunaan memori pada dua buah <i>map</i>	147
Tabel 5.27 Pengujian performansi penggunaan memori pada dua buah <i>map</i>	148

Tabel 5.28 Pengujian performansi kecepatan waktu pada dua buah <i>map</i>	149
Tabel 5.29 Pengujian performansi kecepatan waktu pada dua buah <i>map</i>	150
Tabel 5.30 Pengujian performansi kecepatan waktu pada dua buah <i>map</i>	151
Tabel 5.31 Pengujian performansi kecepatan waktu pada dua buah <i>map</i>	152
Tabel 5.32 Hasil rata-rata pengujian alokasi memori pada kelas TreeMap dan IntKeyTreeMap	153
Tabel 5.33 Hasil rata-rata pengujian kecepatan waktu pada kelas TreeMap dan IntKeyTreeMap	153
Tabel 5.34 Rangkuman perbandingan antara TreeMap pada Java dan IntKeyTreeMap	154

DAFTAR PROGRAM

Kode Program 3.1 Deklarasi kelas Entry di dalam TreeMap	52
Kode Program 3.2 Deklarasi kelas IntKeySmallMap	80
Kode Program 3.3 Deklarasi kelas IntKeyArrayMap	81
Kode Program 3.4 Deklarasi kelas RightThreadedAVLTree	82
Kode Program 3.5 Deklarasi kelas RightThreadedAVL	83
Kode Program 4.6 Implementasi metode <i>put()</i> pada IntKeySmallMap.....	111
Kode Program 4.7 Implementasi metode <i>remove()</i> pada IntKeySmallMap	111
Kode Program 4.8 Implementasi metode <i>get()</i> pada IntKeySmallMap.....	112
Kode Program 4.9 Implementasi metode <i>put()</i> pada IntKeyArrayMap.....	112
Kode Program 4.10 Implementasi metode <i>remove()</i> pada IntKeyArrayMap	113
Kode Program 4.11 Implementasi metode <i>get()</i> pada IntKeyArrayMap.....	114
Kode Program 4.12 Implementasi metode <i>iterator()</i> pada IntKeyArrayMap.....	115
Kode Program 4.13 Implementasi metode <i>get()</i> pada.....	116
Kode Program 4.14 Implementasi metode <i>get()</i> pada.....	117
Kode Program 4.15 Implementasi metode <i>insert()</i> pada RightThreadedAVL ...	117
Kode Program 4.16 Implementasi metode <i>insertBalanceFactor()</i> pada RightThreadedAVL.....	118
Kode Program 4.17 Implementasi metode <i>remove()</i> pada RightThreadedAVL.	119
Kode Program 4.18 Implementasi metode <i>removeBalanceFactor()</i> pada RightThreadedAVL.....	120
Kode Program 4.19 Implementasi metode <i>adjustBalanceFactor()</i> pada RightThreadedAVL.....	121
Kode Program 4.20 Implementasi metode <i>singleRotate()</i> pada RightThreadedAVL.....	121
Kode Program 4.21 Implementasi metode <i>doubleRotate()</i> pada RightThreadedAVL.....	121
Kode Program 4.22 Implementasi metode <i>search()</i> pada RightThreadedAVL..	122
Kode Program 4.23 Implementasi metode <i>get()</i> pada NodeEntry	123
Kode Program 4.24 Implementasi metode <i>get()</i> pada NodeEntry	123
Kode Program 4.25 Implementasi metode <i>put()</i> pada IntKeyTreeMap.....	123

Kode Program 4.26 Implementasi metode <i>remove()</i> pada <i>IntKeyTreeMap</i>	125
Kode Program 4.27 Implementasi metode <i>get()</i> pada <i>IntKeyTreeMap</i>	127
Kode Program 4.28 Implementasi metode <i>iterator()</i> pada <i>IntKeyTreeMap</i>	127
Kode Program 5.29 Deklarasi kelas untuk unit testing dengan JUnit	135
Kode Program 5.30 Deklarasi fungsi <i>testMethodPut()</i>	136
Kode Program 5.31 Deklarasi fungsi <i>testReturnMethodPut()</i>	136
Kode Program 5.32 Deklarasi fungsi <i>testMethodRemove()</i>	137
Kode Program 5.33 Deklarasi fungsi <i>testReturnMethodRemove()</i>	137
Kode Program 5.34 Deklarasi fungsi <i>testMethodSize()</i>	138
Kode Program 5.35 Deklarasi fungsi <i>testKeySet()</i>	138
Kode Program 5.36 Deklarasi fungsi <i>testValues()</i>	139
Kode Program 5.37 Deklarasi fungsi <i>testIterator()</i>	139