TUGAS AKHIR

ANALISA ALGORITMA MERGESORT MENGGUNAKAN TEKNIK DIVIDE AND CONQUER

Di susun sebagai salah satu syarat untuk memperoleh gelar sarjana sains
Program Studi Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Diponegoro Semarang

Disusun Oleh :

ARIEF TEGUH RAHARDJO
J 101 95 1180

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNVERSITAS DIPONEGORO
SEMARANG
2000

HALAMAN PENGESAHAN

Lembar 1

Judul : Analisa Algoritma Mergesort Menggunakan Teknik Divide and Conquer

Nama : Arief Teguh Rahardjo

NIM : J 101 95 1180

Jurusan : Matematika

Telah selesai dan layak untuk diujikan pada tanggal 5 Februari 2001

Semarang, Februari 2001

Pembimbing Anggota

Awalina Kurniastuti, SSi

NIP : 132 205 525

Pembimbing Utama

Drs. Kushartantya, MIKomp

NIP : 130 805 062

HALAMAN PENGESAHAN

Lembar 2

Judul : Analisa Algoritma Mergesort Menggunakan Teknik Divide and Conquer

Nama : Arief Teguh Rahardjo

NIM : J 101 95 1180

Jurusan : Matematika

Telah lulus ujian sarjana pada tanggal 5 Februari 2001

Semarang, Februari 2001

Panitia Penguji Ujian Sarjana
Jurusan Matematika
Ketua

Drs. Kushartantya, MIKomp
NIP : 130 805 062

Ketua
Jurusan Matematika

Drs. Bayu Surarso, MSc. PhD
NIP. 131 764 886

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Allah SWT yang telah memberikan rahmat dan hidayahnya, sehingga penulis dapat menyelesaikan Tugas Akhir dengan judul "ANALISA ALGORITMA MERGESORT MENGGUNAKAN TEKNIK DIVIDE AND CONQUER ".

Tugas akhir ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata Satu pada Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Diponegoro.

Pada kesempatan ini penulis mengucapkan terima kasih kepada :

1. Bapak Drs. Kushartantya, MIKomp, sebagai Pembimbing Utama yang telah membimbing dan mengarahkan penulis dalam penyusunan Tugas Akhir ini.
2. Ibu Awalina Kurniastuti, SSi , sebagai Pembimbing Anggota yang telah membimbing dan mengarahkan penulis dalam penyusunan Tugas Akhir ini.
3. Bapak Drs. Bayu Surarso, MSc. PhD, sebagai Ketua Jurusan Matematika yang telah membantu dalam menyelesaikan Tugas Akhir ini.
4. Ibu Dra. Tatik Widiharih, MSi , selaku Dosen Wali yang telah membantu dalam menyelesaikan Tugas Akhir ini.
5. Bapak, Ibu, dan Adik-adikku yang telah membantu dan memberikan doa restu dalam menyelesaikan Tugas Akhir ini.
6. Rekan-rekanku " Angkatan 95 ".
7. Semua pihak yang telah membantu penulis dalam menyelesaikan Tugas Akhir ini yang tidak dapat saya sebutkan satu persatu.

Penulis menyadari bahwa Tugas Akhir ini masih jauh dari sempurna, oleh karena itu penulis mengharapkan kritik dan saran agar dapat memperbaikinya di masa yang akan datang.

Penulis berharap semoga Tugas Akhir ini bermanfaat bagi pembaca dan perkembangan ilmu pengetahuan di masa yang akan datang.

Semarang, Februari 2001

Penulis

DAFTAR ISI

ABSTRAK

Teknik Divide and Conquer (DANDC) merupakan salah satu teknik untuk mendesain suatu algoritma, yang pada prinsipnya membagi n input menjadi k subset input ($1 < k \leq n$) dengan ukuran yang sama. Setiap k subset input akan terdapat k sub problem sedemikian sehingga setiap sub problem mempunyai solusi. Selanjutnya dari k sub solusi dikombinasi sehingga diperoleh solusi optimal. Algoritma mergesort menggunakan teknik Divide and Conquer terdiri dari dua prosedur yaitu prosedur MERGE dan MERGESORT. Misalkan terdapat suatu barisan dengan n elemen yang ditempatkan dalam suatu array A. Barisan tersebut akan diurutkan secara ascending. Dengan menggunakan algoritma mergesort dengan teknik DANDC barisan tersebut dibagi menjadi dua bagian sehingga dihasilkan A(1), A(2), ..., A(n/2) dan A(n/2+1), A(n/2+2), ..., A(n). Jika pembagian barisan tersebut ukuran setiap bagiannya masih besar (ukuran > 1), maka setiap bagian dibagi lagi menjadi dua bagian yang lebih kecil sedemikian sehingga setiap bagiannya mempunyai satu elemen. Selanjutnya setiap pasang bagian dikombinasi sehingga dihasilkan urutan yang tunggal dari n elemen semula.

ABSTRACT

Divide and conquer technique (DANDC) is a technique to design an algorithm, which principally divides n inputs into k subset inputs ($1 < k \leq n$) in the same size. Every k subset of input containts k sub problems and each of sub problem has solution. Then k sub solutions are combined to get the optimum solution. The mergesort algorithm using divide and conquer technique consist of two procedures, MERGE and MERGESORT procedures. Let, there is a list with n element placed into an array A. The list will be ascendingly sorted. Using mergesort algorithm, the list is divided into two parts so there will be A(1), A(2), ..., A(n/2) and A(n/2+1), A(n/2+2), ..., A(n). If the size of each part of the list division remains large (size > 1), then each part is divided again into two smaller parts so that there will be one element for each part. Next, every pair of parts is combined to get a single sort of the previous n element.

DAFTAR SIMBOL

$>$	= Lebih dari
$<$	= Kurang dari
$\geq$	= Lebih dari atau sama dengan
$\leq$	= Kurang dari atau sama dengan
Σ	= Jumlahan
T(n)	= Fungsi Running Time
O	= Notasi Big-O
θ	= Notasi Theta
Ω	= Notasi Omega
$\lvert\ \rvert$	= Harga mutlak
Div	= Pembagian menghasilkan bilangan integer
$\log_2$	= Logaritma dengan basis dua

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pengurutan data (sorting) secara umum dapat didefinisikan sebagai suatu proses untuk menyusun kembali himpunan obyek menggunakan aturan tertentu. Terdapat dua jenis pengurutan data yaitu pengurutan secara urut naik (ascending) dan pengurutan secara urut turun (descending).

Pengurutan data merupakan salah satu bagian yang penting dalam ilmu komputer, karena waktu yang diperlukan untuk melakukan proses pengurutan perlu dipertimbangkan. Data yang harus diurutkan sangat bervariasi meliputi jumlah data dan jenis data yang akan diurutkan.

Keuntungan yang dapat diperoleh dari data yang telah terurutkan antara lain adalah data mudah dicari (misalnya dalam buku telepon atau kamus bahasa), mudah dibetulkan, mudah dihapus, mudah disisipi atau digabung. Dalam keadaan terurutkan dengan mudah dapat dilihat apakah ada data yang hilang (misalnya dalam tumpukan kartu bridge). Pengurutan memegang peranan yang penting dalam proses pencarian data yang harus dilakukan berulang kali. Adapun metode pengurutan yang telah dikenal adalah bublesort, quicksort, radixsort, heapsort, mergesort, dan sebagainya.

Algoritma merupakan suatu urutan dari barisan langkah - langkah atau instruksi yang digunakan untuk menyelesaikan suatu masalah. Jika terdapat suatu masalah, maka kemungkinan terdapat lebih dari satu algoritma yang dapat digunakan untuk menyelesaikan masalah tersebut. Pada saat memilih suatu algoritma dari sekian algoritma yang ditawarkan

terdapat beberapa kriteria yang perlu diperhatikan adanya output, efektifitas dan efisiensi, jumlah langkah yang berhingga, berakhir, dan terstruktur.

Hal - hal yang dipelajari dalam studi algoritma meliputi merencanakan algoritma, menyatakan algoritma, validitas algoritma, menganalisa algoritma, dan menguji program dari suatu algoritma. Pada saat menganalisa suatu algoritma, terdapat dua hal yang menyangkut tingkat efisiensi suatu algoritma yaitu besarnya running time dan banyaknya memori (storage) yang digunakan. Terdapat beberapa teknik mendisain suatu algoritma diantaranya teknik iteratif, rekursif, backtracking, greedy, divide and conquer, dan sebagainya. Masing - masing teknik tersebut memiliki keunggulan sesuai dengan masalah yang dihadapi.

1.2 Perumusan Masalah

Pada penulisan Tugas Akhir ini akan dibahas :

1. Bagaimana teknik divide and conquer (DANDC).
2. Bagaimana algoritma mergesort menggunakan teknik divide and conquer (DANDC).
3. Bagaimana analisa algoritma (running time) mergesort menggunakan teknik divide and conquer (DANDC).
4. Bagaimana implementasi algoritma mergesort menggunakan teknik DANDC dengan Bahasa Pascal.

1.3 Pembatasan Masalah

Pengurutan data yang akan dibentuk adalah pengurutan data dalam suatu array secara urut naik (ascending). Pada algoritma mergesort menggunakan teknik DANDC setiap data akan selalu dibagi menjadi dua bagian dengan menggunakan operator div 2, dimana bagian pertama merupakan banyaknya data div 2, sedangkan bagian kedua merupakan banyaknya data dikurangi banyaknya elemen pada bagian pertama. Proses tersebut diulang sedemikian sehingga setiap bagiannya hanya mempunyai satu elemen. Adapun implementasi algoritma mergesort menggunakan teknik DANDC dengan Bahasa Pascal hanya ditunjukkan pada pengurutan angka.

1.4 Sistematika Pembahasan

BAB I berisi tentang latar belakang, perumusan masalah, dan pembatasan masalah dari tugas akhir ini, disertai dengan sistematika pembahasannya.

BAB II berisi tentang relasi rekurensi, bahasa pemrograman Pascal yang meliputi prosedur rekursif dan grafik. Di samping itu dibahas pula tentang algoritma beserta kompleksitasnya, running time yang meliputi notasi Big-O, kombinasi fungsi pertumbuhan, aturan umum analisa algoritma, dan analisa struktur kontrol.

BAB III berisi tentang teknik divide and conquer (DANDC), algoritma mergesort menggunakan teknik DANDC, serta analisanya.

BAB IV berisi tentang penjelasan program grafik running time mergesort menggunakan teknik DANDC.

BAB V berisi kesimpulan dari tugas akhir ini.

BAB II

MATERI PENUNJANG

2.1 Relasi Berulang (Rekurensi)

Definisi 2.1.1

Sebuah relasi rekurensi (recurrence relation) untuk deret $a_0, a_1, \ldots$ merupakan sebuah persamaan yang mengkaitkan a_n dengan pendahulunya $a_0, a_1, \ldots, a_{n-1}$.

Contoh

Diberikan instruksi untuk membentuk sebuah deret berikut

1. Awali dengan 5.
2. Jika diberikan sebarang suku, tambahkan 3 untuk mendapatkan suku berikutnya.

Sehingga jika dibuat daftar suku - suku deret tersebut, maka didapatkan

5, 8, 11, 14, 17, ... (2.1)

Jika deret (2.1) dinyatakan sebagai $a_1, a_2, \ldots$ maka instruksi ke-1 dapat dinyatakan ulang sebagai

$a_1 = 5$ (2.2)

dan instruksi ke-2 dinyatakan sebagai

$a_n = a_{n-1} + 3 \quad , n \geq 2$ (2.3)

Dengan mengambil $n = 2$ dalam persamaan (2.3), maka didapatkan

$a_2 = a_1 + 3$

$= 5 + 3 = 8$

Dengan mengambil n = 3 dalam persamaan (2.3), maka didapatkan

$a_3 = a_2 + 3$

$= 8 + 3$

$= 11$

Jadi dengan menggunakan persamaan (2.2) dan (2.3) dapat dihitung sebarang suku dalam deret tersebut.

Relasi rekurensi seperti pada persamaan (2.3), nilai - nilai awal seperti pada (2.2) harus diberikan. Nilai - nilai awal ini disebut syarat awal (initial conditions).

Menyelesaikan suatu relasi rekurensi yang melibatkan a_0, a_1, ... sama halnya dengan mencari rumus eksplisit untuk bentuk umum a_n. Terdapat beberapa metode penyelesaian relasi rekurensi, salah satu diantaranya adalah dengan cara iterasi (iteration).

Contoh

Carilah bentuk umum pada contoh relasi berulang di atas.

Penyelesaian

Pada contoh di atas, terdapat

$a_1 = 5$

$a_n = a_{n-1} + 3 \quad , n \geq 2$ (2.4)

Dengan mengganti n dengan n -1, maka persamaan (2.4) dapat dituliskan

$a_{n-1} = a_{n-2} + 3 \quad , n \geq 3$ (2.5)

Jika persamaan (2.5) disubstitusikan ke dalam persamaan (2.4), maka didapatkan

$a_n = a_{n-2} + 3 + 3 \quad , n \geq 3$

$= a_{n-2} + 2.3 \quad , n \geq 3 \qquad (2.6)$

Dengan mengganti n dengan n - 2, maka persamaan (2.4) dapat dituliskan

$a_{n-2} = a_{n-3} + 3 \quad , n \geq 4 \qquad (2.7)$

Jika persamaan (2.7) disubstitusikan ke dalam persamaan (2.6), maka didapatkan

$a_n = a_{n-3} + 3 + 2.3 \quad , n \geq 4$

$= a_{n-3} + 3.3 \quad , n \geq 4$

Jadi secara umum didapatkan

$a_n = a_{n-k} + k.3 \quad , k = 0, 1, 2, 3, \ldots \qquad (2.8)$

misalkan k = n - 1, maka persamaan (2.8) dapat dinyatakan

$a_n = a_{n-(n-1)} + (n - 1).3$

$= a_{n-n+1} + (n - 1).3$

$= a_1 + 3.(n - 1)$

karena $a_1 = 5$, maka didapatkan bentuk umumnya yaitu

$a_n = 5 + 3.(n - 1) \qquad (2.9)$

2.2 Bahasa Pemrograman Pascal

2.2.1 Rekursif

Rekursif berarti suatu proses yang dapat memanggil dirinya sendiri. Rekursif merupakan teknik pemrograman yang berdayaguna yang digunakan pada pemrograman

dengan mengekspresikannya dalam suku - suku dari program lain dengan menambahkan langkah - langkah sejenis.

Contoh

```
function fakt ( n : integer ) : integer ;
begin
    if  n = 0 then
       fakt := 1 ;
    else
       fakt := n * fakt (n - 1) ;
end ;
```

2.2.2 Grafik

Terdapat syarat - syarat yang harus dipenuhi sebelum membuat program grafik dengan Bahasa Pascal, yaitu :

1. Adanya perangkat keras yang sesuai artinya suatu perangkat keras yang ditunjang Turbo Pascal. Hal ini tidak akan banyak mengalami kesulitan karena telah banyaknya perangkat keras yang ditunjang Turbo Pascal.
2. Adanya berkas BGI

   Berkas - berkas BGI merupakan contoh graphics driver yang disediakan Borland supaya program yang dibuat dengan Turbo Pascal dapat mengendalikan berbagai macam adapter tampilan. Tidak semua berkas BGI digunakan, cukup yang menunjang perangkat lunak saja. Berikut ini adalah tabel berkas BGI beserta perangkat keras yang ditunjangnya.

Tabel 1

NAMA BERKAS	PERANGKAT KERAS
CGA . BGI	IBM CGA , MCGA
EGA VGA . BGI	IBM EGA , VGA
HERC . BGI	Hercules Monokrom
ATT . BGI	AT & T
PC3270 . BGI	IBM 3270 PC

3. Berkas Graph .TPU

Berkas Graph . TPU merupakan berkas yang diperlukan pada saat kompilasi, sehingga harus dapat dijangkau oleh compiler.

4. Berkas - berkas huruf (font)

Berkas - berkas *.CHR berisikan vektor - vektor model huruf yang diperlukan jika akan menggunakan fasilitas gambar teks pada program grafik.

Setiap program Turbo Pascal yang menggunakan kemampuan grafik memiliki kerangka sebagai berikut :

```
Program (nama program) ;
Uses graph ;
Var GraphDrv, GraphMode, SandiSalah : integer ;
Begin
   GraphDrv := detect ;
   Initgraph ( GraphDrv, GraphMode, ' ' ) ;
   SandiSalah := graph_result ;
   if SandiSalah < > grok then
                Writeln (' kesalahan graphics') ;
   Grapherrormsg (SandiSalah) ;
   Writeln (' program dihentikan ') ;
   Halt (1) ;
End .
```

Penjelasan

Uses graph menyatakan bahwa program tersebut akan menggunakan unit graph. GraphDrv, GraphMode, SandiSalah merupakan tiga variabel global. GraphDrv menyatakan nomor graphics driver yang akan digunakan. Nomor ini dapat dinyatakan dengan angka ataupun dengan nama konstanta yang sudah dideklarasikan dalam unit graph. Berikut ini adalah tabel nilai dan nama konstanta graphics driver.

Tabel 2

NAMA KONSTANTA	NILAI
Current Driver	- 128
Detect	0
CGA	1
MCGA	2
EGA	3
EGA 64	4
EGA Mono	5
IBM 8514	6
Herc Mono	7
ATT 400	8
VGA	9
PC 3270	10

Pada kerangka program di atas, GraphDrv diisi dengan detect (0), yang artinya program menentukan sendiri jenis perangkat keras yang terpasang. Hal ini dimungkinkan karena biasanya terdapat daftar perlengkapan yang tersimpan dalam memori utama. Keuntungan cara ini adalah program tidak akan mengalami masalah, walaupun dijalankan pada berbagai komputer dengan perangkat keras grafiknya. Dengan detect, mode yang otomatis dipilih adalah mode tertinggi yaitu CGA Hi

atau 4. Jika tidak menggunakan detect, maka harus menetapkan nilai GraphMode. Contohnya jika diisikan CGA kedalam Graph Drv, maka harus mengubah pernyataan pada bagian atas program utama, yaitu :

```
...
GraphDrv := CGA ;
GraphMode := CGA Co
InitGraph ( GraphDrv, GraphMode, ' ' ) ;
...
```

Pada saat prosedur Initgraph (GraphDrv, GraphMode, ' ') dieksekusi, adapter tampilan dideteksi, selanjutnya graphics driver yang sesuai akan berpindah dari berkas ke memori, dan komputer pindah ke mode grafik. String kosong ' ' menunjukkan tempat grafik driver ada di direktori yang aktif saat itu. Jika berkas BGI yang digunakan terdapat di direktori lain, maka string kosong dapat diganti dengan nama jalur (path name) lengkap, contohnya 'C : \ Driver'.

Fungsi graph result menghasilkan suatu nilai yang merupakan sandi kesalahan yang dimasukkan ke dalam variabel SandiSalah. Unit graph telah menetapkan apa saja sandi - sandi tersebut. Langkah selanjutnya adalah memeriksa nilai SandiSalah, dengan membandingkan grok (0). Jika tidak sama berarti pada operasi Initgraph terjadi kesalahan. Jika SandiSalah dijadikan argumen untuk prosedur Grapherrormsg, maka dapat langsung diketahui kesalahan apa yang terjadi di layar. Selanjutnya program akan dihentikan dengan Halt (1). Jika operasi Initgraph

berhasil, maka SandiSalah akan sama dengan grok (0) dan proses berlanjut ke program grafik.

Titik (pixel) dapat ditampilkan dengan prosedur put pixel yang memerlukan tiga argumen yaitu koordinat x (baris), koordinat y (lajur), dan warna pixel yang bersangkutan. Pembuatan bilangan acak dapat dilakukan dengan fungsi random, yang memerlukan satu argumen untuk menentukan batas atas bilangan acak yang dihasilkannya, sedangkan batas bawahnya adalah nol.

Contoh

Put pixel (random (x), random (y), red) ;

Sebelum menghasilkan bilangan acak agar disertakan randomize supaya urutan bilangan acak yang dihasilkan tidak selalu sama.

Terdapat tiga prosedur menggambar garis yang disediakan oleh Turbo Pascal yaitu line, line to, dan line rel. Prosedur line membutuhkan 4 argumen yang semuanya integer menyatakan koordinat dua titik yang akan dihubungkan dengan garis. Prosedur line to hanya membutuhkan dua argumen yaitu koordinat x dan y yang merupakan titik tujuannya, sedangkan line rel memerlukan dua argumen yang menyatakan banyaknya pergeseran yang harus dilakukan untuk mencapai ujung garis.

Contoh

Line (10, 20, 30, 40) ;

line to (10, 20);

Line rel (50, - 50) ;

2.3 Algoritma

Algoritma merupakan suatu urutan dari barisan langkah - langkah atau instruksi yang digunakan untuk menyelesaikan suatu masalah. Istilah algoritma pertama kali diperkenalkan oleh seorang ahli matematika yaitu Abu Ja'far Muhammad Ibnu Musa Alkhawarizmi. Jika terdapat suatu masalah, maka kemungkinan lebih dari satu algoritma yang dapat digunakan untuk menyelesaikan masalah tersebut. Suatu algoritma harus memenuhi beberapa kriteria berikut :

1. Adanya output

   Suatu algoritma harus mempunyai output. Output tersebut tentunya merupakan solusi dari masalah yang ada.

2. Efektif dan efisien

   Suatu algoritma dikatakan efektif jika algoritma tersebut menghasilkan solusi yang sesuai dengan masalah yang diselesaikan (tepat guna), sedangkan suatu algoritma dikatakan efisien jika waktu proses algoritma tersebut relatif singkat dan penggunaan memorinya relatif kecil.

3. Jumlah langkah berhingga

   Banyaknya langkah atau instruksi dalam suatu algoritma harus berhingga, karena jika tidak demikian, proses yang dilakukan memerlukan waktu yang lama dan diperoleh hasil yang tidak sesuai dengan masalah yang ada.

4. Berakhir

   Proses mencari solusi suatu masalah harus berakhir.

5. Terstruktur

Urutan dari barisan langkah - langkah yang digunakan harus disusun sedemikian rupa agar proses penyelesaiannya tidak berbelit - belit, sehingga waktu prosesnya akan relatif singkat. Hal ini akan memperlihatkan bahwa bagian - bagian dari proses tersebut dapat dibedakan secara jelas (input, proses, output) yang memudahkan dalam melakukan proses pemeriksaan ulang.

Hal - hal yang menyangkut studi tentang algoritma yaitu :

1. Merencanakan suatu algoritma

Sebelum menyatakan suatu algoritma, terlebih dahulu menentukan model atau desain penyelesaiannya. Suatu masalah dapat diselesaikan dengan berbagai macam model atau desain. Tentunya ada satu model yang terbaik dari sekian algoritma yang ada, sehingga teknik variasi desain harus dikuasai sebaik - baiknya.

2. Menyatakan suatu algoritma

Menyatakan suatu algoritma adalah membuat barisan langkah - langkah atau konstruksi secara terurut yang digunakan untuk menyelesaikan suatu masalah. Pernyataan tersebut hendaknya dibuat secara singkat dan terstruktur. Menyatakan suatu algoritma dapat dilakukan dengan dua cara yaitu diagram alur (flow chart) dan bahasa semu.

3. Validitas suatu algoritma

Indikasi suatu algoritma valid (absah) adalah jika penyelesaian yang diperoleh dapat memecahkan permasalahan yang ada, selain itu solusi, perhitungan, dan prosedurnya selalu benar untuk semua kemungkinan input yang ada.

4. Menganalisa suatu algoritma

Studi tentang analisa algoritma menyangkut 2 hal yaitu running time dan banyaknya memori (storage) yang digunakan.

5. Menguji suatu algoritma

Algoritma yang akan digunakan untuk menyelesaikan suatu masalah dengan cepat dan tepat diperoleh solusinya dengan alat bantu komputer. Algoritma tersebut disajikan terlebih dahulu ke dalam bahasa pemrograman. Pengujian terhadap program tersebut dilakukan dengan dua fase yang dilaksanakan secara simultan yaitu :

a. Fase debugging

Suatu fase dari proses program eksekusi yang melakukan koreksi terhadap kesalahan program yaitu kesalahan dalam bahasa pemrograman baik secara logika maupun sintaks.

b. Fase profiling

Suatu fase yang akan bekerja jika program tersebut sudah benar. Fase ini juga digunakan untuk mengukur waktu tempuh (running time) dan penggunaan memori (storage).

2.4 Kompleksitas Algoritma

Analisa algoritma mengacu pada proses memperoleh perkiraan dari waktu dan ruang yang diperlukan untuk mengeksekusi algoritma. Kompleksitas algoritma mengacu pada jumlah waktu dan ruang yang dibutuhkan untuk mengeksekusi algoritma. Waktu yang

diperlukan untuk mengeksekusi sebuah algoritma merupakan sebuah fungsi dari masukan. Misalkan untuk sebarang masukan berukuran n, algoritma A membutuhkan tepat c_1n satuan waktu dan algoritma B membutuhkan c_2n^2 satuan waktu. Untuk ukuran masukan tertentu algoritma B mungkin menjadi superior. Sebagai contoh andaikan untuk sebarang masukan berukuran n, algoritma A membutuhkan 300n satuan waktu dan algoritma B membutuhkan $5n^2$ satuan waktu. Untuk masukan berukuran n = 5, algoritma A membutuhkan 1500 satuan waktu dan algoritma B membutuhkan 125 satuan waktu. Dengan demikian B lebih cepat. Tentu saja untuk n > 60 algoritma A lebih cepat dari pada B.

Misalkan terdapat sebuah komputer yang dapat mengeksekusi satu langkah dalam 1 mikrodetik (10^{-6} detik). Tabel 3 menunjukkan waktu eksekusi berdasarkan asumsi tersebut, untuk berbagai ukuran masukan (log yang ada merupakan log dengan basis dua).

Tabel 3

Banyaknya langkah penyelesaian untuk masukan berukuran n	waktu untuk mengeksekusi jika n =			
	3	6	9	12
1	10^{-6} detik	10^{-6} detik	10^{-6} detik	10^{-6} detik
log log n	10^{-6} detik	10^{-6} detik	2×10^{-6} detik	2×10^{-6} detik
log n	2×10^{-6} detik	3×10^{-6} detik	3×10^{-6} detik	4×10^{-6} detik
n	3×10^{-6} detik	6×10^{-6} detik	9×10^{-6} detik	10^{-5} detik
n log n	5×10^{-6} detik	2×10^{-5} detik	3×10^{-5} detik	4×10^{-5} detik
n^2	9×10^{-6} detik	4×10^{-5} detik	8×10^{-5} detik	10^{-4} detik
n^3	3×10^{-5} detik	2×10^{-4} detik	7×10^{-4} detik	2×10^{-3} detik
2^n	8×10^{-6} detik	6×10^{-5} detik	5×10^{-4} detik	4×10^{-3} detik

Perhatikan bahwa eksekusi algoritma yang memerlukan 2^n langkah untuk sebuah masukan berukuran n hanya layak untuk ukuran masukan yang sangat kecil. Algoritma yang membutuhkan n^2 dan n^3 langkah juga menjadi tidak layak untuk ukuran masukan yang relatif besar.

2.5 Running Time Program

Running time suatu program adalah banyaknya waktu yang digunakan oleh suatu program dalam menyelesaikan suatu masalah. Running time suatu program biasanya dinotasikan T(n). Hal - hal yang mempengaruhi running time suatu program adalah sebagai berikut :

1. Besar dan jenis input data

   Adanya kenyataan bahwa running time program tergantung pada input menunjukkan bahwa running time suatu program harus ditetapkan sebagai suatu fungsi dari input. Running time program tidak tergantung pada nilai input, akan tetapi tergantung pada ukuran input. Misalnya dalam suatu pengurutan (sorting) terdapat sebuah daftar item sebagai input. Item - item tersebut harus diurutkan, sehingga dihasilkan item - item yang sama sebagai output baik secara ascending atau descending. Ukuran input pada sebuah sorting program adalah jumlah item - item yang diurutkan atau panjang dari daftar item. Di samping ukuran input, running time program juga tergantung pada jenis input data. Jika jenis data yang diinginkan dengan tingkat ketelitian tunggal (single precision) maka running timenya relatif cepat dari pada menggunakan data dengan tingkat

ketelitian ganda (double precision). Demikian pula jika dibandingkan dengan data - data yang triple precision atau quadruple precision.

2. Banyaknya langkah

Makin banyak langkah atau instruksi yang digunakan maka makin lama running time yang diperlukan dalam proses tersebut.

3. Jenis operasi

Jenis operasi tersebut meliputi operasi aritmatika, operasi nalar atau logika, dan sebagainya.

4. Komputer dan kompilator

Meskipun suatu program yang dibuat telah mencapai running time yang sangat efisien (meliputi 3 hal di atas), namun jika digunakan komputer yang berkemampuan lambat maka running timenya akan menjadi lambat. Kompilator yang digunakan juga berpengaruh terhadap running time program.

Adanya kenyataan bahwa running time program tergantung pada komputer dan kompilator yang digunakan, hal ini menunjukkan bahwa running time tidak dapat dinyatakan sebagai satuan waktu standar (detik), akan tetapi hanya dapat dinyatakan running time suatu program adalah sebanding dengan n atau n^2 atau yang lainnya. Oleh karena itu dalam menentukan running time suatu program diasumsikan bahwa running time yang diperoleh merupakan jumlahan operasi yang dilaksanakan oleh komputer yang ideal. Salah satu cara untuk menentukan jumlahan operasi yang dilaksanakan adalah menggunakan notasi Big-O

Dalam melakukan analisa algoritma terdapat tiga hal yang sering diperhitungkan yaitu :

1. Best case running time yaitu suatu keadaan yang merupakan nilai minimum dari fungsi T(n) untuk setiap input yang mungkin atau keadaan terbaik dimana algoritma memerlukan waktu yang singkat.
2. Worst case running time yaitu suatu keadaan yang merupakan nilai maksimum dari fungsi T(n) untuk setiap input yang mungkin atau keadaan terburuk dimana algoritma memerlukan waktu yang paling lama.
3. Average case running time yaitu keadaan dari waktu tempuh yang ekuivalen dengan nilai ekspektasi dari fungsi T(n) untuk setiap input data yang mungkin. Nilai ekspektasi dari fungsi T(n) = E, yang didefinisikan sebagai :

   $E = n_1p_1 + n_2p_2 + ... + n_kp_k$

   dengan

   $n_1, n_2, ..., n_k$ merupakan nilai - nilai yang muncul.

   $p_1, p_2 ..., p_k$ merupakan probabilitas dari setiap nilai - nilai yang muncul.

Dalam menentukan running time suatu algoritma dipergunakan worst case running time, hal ini berdasarkan alasan - alasan sebagai berikut :

1. Worst case running time merupakan waktu maksimal yang diperlukan suatu algoritma dalam menyelesaikan suatu masalah.

2. Worst case running time dari suatu algoritma merupakan suatu batas atas dari suatu running time. Dengan demikian akan memberikan jaminan bahwa algoritma yang digunakan akan mempunyai akhir proses.
3. Untuk beberapa algoritma, worst case running time merupakan kejadian yang sering terjadi. Sebagai contoh dalam proses pencarian data base untuk mendapatkan suatu informasi, worst case running time selalu terjadi ketika informasi yang dibutuhkan tidak berada dalam data base.

2.6 Menghitung Running Time

2.6.1 Notasi Big - O

Definisi 2.6.1.1

T(n) adalah O(f(n)) jika terdapat 2 buah konstanta bulat positif c dan n_0 sedemikian hingga $T(n) \leq cf(n)$, untuk setiap $n \geq n_0$.

Contoh

Tunjukkan bahwa $T(n) = (n + 1)^2$ adalah $O(n^2)$.

Penyelesaian

Misalkan $T(0) = n_0 = 1$ dan $T(1) = c = 4$, maka berdasarkan definisi 2.6.1.1 berlaku $(n + 1)^2 \leq 4n^2$, untuk setiap $n \geq 1$

Teorema 2.6.1.1

Jika T(n) adalah fungsi polinomial dalam n dengan derajad m, yang ditulis dengan notasi $T(n) = a_m n^m + a_{m-1} n^{m-1} + ... + a_1 n + a_0$, maka T(n) adalah $O(n^m)$

Bukti

$$T(n) \leq \sum_{i=0}^{m} |a_i| n^i$$

$$\leq n^m \sum_{i=0}^{m} |a_i| n^{i-m}$$

$$\leq n^m \sum_{i=0}^{m} |a_i|, \text{ untuk } n \geq 1$$

sehingga T(n) adalah $O(n^m)$

■

Contoh

Tunjukkan $T(n) = 3n^3 + 2n^2$ adalah $O(n^3)$

Penyelesaian

Berdasarkan teorema 2.6.1.1 karena $T(n) = 3n^3 + 2n^2$ merupakan fungsi polinomial dalam n dengan derajad 3, maka T(n) adalah $O(n^3)$.

Contoh

Tunjukkan $T(n) = 3^n$ bukan $O(2^n)$

Penyelesaian

Andaikan $T(n) = 3^n$ adalah $O(2^n)$ sehingga berdasarkan definisi 2.6.1.1 terdapat dua buah konstanta bulat positif c dan n_0 sedemikian sehingga $3^n \leq c2^n$ untuk $n \geq 1$ Dengan demikian $c \geq (3/2)^n$, untuk $n \geq 1$. Akan tetapi $(3/2)^n$ besarnya berubah - ubah sesuai dengan besarnya n. Jadi tidak ada konstanta c yang dapat melebihi $(3/2)^n$ untuk semua n. Jadi pengandaian diingkar, sehingga didapatkan $T(n) = 3^n$ bukan $O(2^n)$.

Definisi 2.6.1.2

$T(n) = \Omega(g(n))$ jika terdapat 2 buah konstanta c dan n_0 sedemikian hingga $T(n) \geq cg(n)$, untuk setiap $n \geq n_0$.

Definisi 2.6.1.3

$T(n) = \theta(h(n))$ jika dan hanya jika $T(n) = O(h(n))$ dan $T(n) = \Omega(h(n))$.

2.6.2 Kombinasi Fungsi Pertumbuhan

Banyak algoritma yang dibuat dalam dua atau lebih sub prosedur. Jumlah langkah yang digunakan oleh komputer untuk menyelesaikan sebuah permasalahan dengan input tertentu pada sebuah algoritma adalah jumlahan setiap langkah yang digunakan oleh sub prosedur tersebut. Untuk mendapatkan estimasi Big-O pada jumlah langkah yang dibutuhkan, sangat perlu untuk menentukan estimasi Big-O pada jumlah langkah yang digunakan pada setiap sub prosedur dan selanjutnya estimasi tersebut dikombinasikan.

Teorema 2.6.2.1

Jika $T_1(n)$ adalah $O(f(n))$ dan $T_2(n)$ adalah $O(g(n))$, maka $T_1(n) + T_2(n)$ adalah $O(\max (f(n), g(n)))$.

Bukti

Berdasarkan definisi 2.6.1.1 terdapat 4 buah konstanta c_1, c_2, n_1, dan n_2 sedemikian sehingga

$T_1(n) \leq c_1 f(n)$, untuk $n \geq n_1$

$T_2(n) \leq c_2 g(n)$, untuk $n \geq n_2$.

Misalkan $n_0 = \max(n_1, n_2)$, maka berlaku

$T_1(n) \leq c_1 f(n)$, untuk $n \geq n_0$

$T_2(n) \leq c_2 g(n)$, untuk $n \geq n_0$, sedemikian hingga berlaku

$T_1(n) + T_2(n) \leq c_1 f(n) + c_2 g(n)$.

Misalkan $c_3 = \max(c_1, c_2)$, maka

$T_1(n) + T_2(n) \leq c_3 f(n) + c_3 g(n)$

$\leq c_3(f(n) + g(n))$

$\leq c_3 [\max(f(n), g(n)) + \max(f(n), g(n))]$

$\leq 2c_3 \max(f(n), g(n)) \leq c \max(f(n), g(n))$

untuk $c = 2c_3$ dan $n \geq n_0$

■

Teorema 2.6.2.1 di atas disebut dengan aturan penjumlahan (rule of sums). Aturan penjumlahan tersebut dapat digunakan untuk menghitung running time barisan atau urutan dari langkah - langkah suatu program.

Contoh

Tunjukkan running time 3 buah langkah yang berurutan berikut $O(n^2)$, $O(n^3)$, dan $O(n\log n)$ adalah $O(n^3)$.

Penyelesaian

Running time dua langkah yang pertama dieksekusi secara berurutan adalah $O(\max(n^2, n^3))$ yaitu $O(n^3)$, sehingga untuk running time seluruh langkah adalah $O(\max n^3, n \log n))$ yaitu $O(n^3)$.

Jadi secara umum running time suatu barisan langkah - langkah adalah running time terbesar dari barisan langkah - langkah tersebut. Pada suatu keadaan tertentu terdapat dua langkah atau lebih yang running timenya tidak seimbang (yang satu lebih besar dari pada yang lain).

Contoh

Tunjukkan running time untuk dua langkah berikut :

$$f(n) = \begin{cases} n^4 & \text{, jika n genap.} \\ n^2 & \text{, jika n ganjil.} \end{cases}$$

$$g(n) = \begin{cases} n^2 & \text{, jika n genap.} \\ n^3 & \text{, jika n ganjil.} \end{cases}$$

adalah $O(n^4)$ jika n genap dan $O(n^3)$ jika n ganjil.

Penyelesaian

Pada kasus tersebut aturan penjumlahan harus diaplikasikan secara langsung, yaitu $O(\max(f(n), g(n))$ adalah $O(n^4)$ jika n genap dan $O(n^3)$ jika n ganjil.

Teorema 2.6.2.2

Jika $T_1(n)$ adalah $O(f(n))$ dan $T_2(n)$ adalah $O(g(n))$, maka $T_1(n).T_2(n)$ adalah $O(f(n).g(n))$.

Bukti

Berdasarkan definisi 2.6.1.1 terdapat 4 buah konstanta c_1, c_2, n_1, dan n_2 sedemikian sehingga

$T_1(n) \leq c_1 f(n)$, untuk $n \geq n_1$

$T_2(n) \leq c_2 g(n)$, untuk $n \geq n_2$

Misalkan $n_0 = \max(n_1, n_2)$ maka

$T_1(n) \leq c_1 f(n)$, untuk $n \geq n_0$

$T_2(n) \leq c_2 g(n)$, untuk $n \geq n_0$

sehingga berlaku

$$T_1(n) \cdot T_2(n) \leq c_1 f(n) \cdot c_2 g(n)$$

$$\leq c_1 . c_2 (f(n).g(n))$$

$$\leq c\,(f(n).g(n))$$

untuk $c = c_1 . c_2$ dan $n \geq n_0$

■

Teorema 2.6.2.2 di atas disebut dengan aturan perkalian (rule of product).

2.6.3 Aturan Umum Analisa Algoritma

Jika ditemukan beberapa algoritma yang berbeda dalam menyelesaikan permasalahan yang sama, maka harus dipilih salah satu yang sesuai dengan kebutuhan. Cara yang digunakan yaitu dengan analisa algoritma. Analisa algoritma dapat digunakan untuk

menentukan efisiensi dari beberapa algoritma, dengan demikian dapat diambil keputusan terbaik untuk menggunakan suatu algoritma.

Secara umum running time pada sebuah statemen atau kelompok statemen mungkin terparameterisasi oleh ukuran input atau beberapa variabel. Parameter untuk running time keseluruhan algoritma (program) adalah n (ukuran input). Adapun aturan umum untuk menganalisa algoritma yaitu :

1. Running time setiap assigment (tugas), baca (read), dan statement write dapat dinyatakan dalam O(1).
2. Running time pada barisan statemen ditentukan dengan aturan penjumlahan (rule of sums) yaitu running time terbesar dari beberapa barisan statemen.
3. Running time pada statemen if adalah harga pada kondisi statemen - statemen yang dieksekusi ditambah waktu untuk mengevaluasi kondisi. Waktu untuk mengevaluasi kondisi tersebut biasanya adalah O(1). Adapun running time untuk susunan if then else adalah waktu untuk mengevaluasi kondisi ditambah waktu terbesar yang dibutuhkan untuk statemen -statemen yang dieksekusi ketika kondisi benar dan salah.
4. Running time pada sebuah loop adalah jumlah seluruh waktu sekitar loop meliputi waktu mengeksekusi sekumpulan statemen dan waktu mengevaluasi kondisi untuk penghentian (biasanya yang terakhir adalah O(1)). Total waktu dari statemen - statemen suatu loop bersarang adalah waktu statemen - statemen yang merupakan hasil kali dari ukuran - ukuran seluruh loop.

2.6.4 Analisa Struktur Kontrol

Analisa algoritma biasanya proses berjalannya terbalik. Pertama menetapkan waktu yang diperlukan oleh instruksi tunggal (sering kali dibatasi dengan konstanta) kemudian menggabungkan waktu tersebut berdasarkan struktur kontrol, selanjutnya menggabungkan instruksi - instruksi pada program.

2.6.4.1 Loop for, while, dan repeat.

Loop for adalah loop termudah untuk dianalisa dibandingkan dengan loop - loop yang lain.

for i := 1 to m do P(i).

Misalkan loop tersebut bagian dari suatu algoritma yang panjang. Terdapat P(i) yang merupakan statemen atau blok statemen yang berbeda di dalam loop. Diberikan t yang merupakan waktu yang dibutuhkan untuk mengeksekusi P(i). Pada kasus ini jika loop digunakan m kali, setiap iterasi membutuhkan waktu t, maka total waktu yang dibutuhkan oleh loop adalah m.t.

Loop while dan repeat biasanya lebih sulit untuk dianalisa dari pada loop for, karena tidak adanya langkah pasti untuk mengetahui berapa besar waktu yang dibutuhkan untuk melalui loop tersebut.

Contoh

Hitung running time Prosedur Bublesort berikut ini

```
Procedure Buble ( var A : array [1..n] of integer ) ;
Var i, j, temp : integer ;
Begin
1.  For i := 1 to n - 1 do
2.      For j := n down to i + 1 do
3.          if A[j - 1] > A[j] then
            Begin
4.              temp := A[j - 1] ;
5.              A[j - 1] := A[j] ;
6.              A[j] := temp ;
            End;
End.
```

Penyelesaian

Misalkan T(n) adalah running time prosedur Buble. Pertama akan dianalisa setiap pernyataan penugasan yang memerlukan waktu konstan, yaitu pada baris 4, 5, dan 6 yang masing - masing memerlukan waktu O(1). Dengan aturan penjumlahan (rule of sums) running time pernyataan pada baris 4, 5, dan 6 adalah O(max(1,1,1)) = O(1). Selanjutnya akan dianalisa pernyataan - pernyataan kondisi dan looping. Untuk pernyataan if, pengujian kondisi memerlukan O(1). Dengan demikian worst case running time untuk group if dari pernyataan 3 - 6 memerlukan waktu O(1).

Selanjutnya untuk baris 2 - 6 tubuh loop memerlukan waktu O(1) bagi setiap iterasi. Jumlah iterasi loop adalah n - i , sehingga dengan aturan perkalian (rule of product) waktu yang diperlukan loop pada baris 2 - 6 adalah O((n - i) x 1) yaitu O(n - i).

Pada baris 1 dieksekusi n - 1 kali, sehingga running time prosedur tersebut adalah :

$$T(n) = \sum_{i=1}^{n-1} (n - i) = n(n - 1) / 2 = n^2/2 - n/2$$

Dengan demikian T(n) adalah $O(n^2)$.

2.6.4.3 Prosedur Rekursif

Rekursif berarti suatu proses yang dapat memanggil dirinya sendiri. Rekursif merupakan teknik pemrograman berdayaguna. Analisa prosedur rekursif meliputi solusi persamaan rekurensi.

Contoh

Hitung running time fungsi faktorial berikut ini.

```
Function Fact ( n : integer ) : integer ;
Begin
1.      if n ≤ 1 then
2.          fact := 1 ;
        Else
3.          Fact := n * fact (n - 1) ;
End ;
```

Penyelesaian

Misalkan T(n) adalah running time untuk fungsi fact(n). Running time untuk baris 1 dan 2 adalah O(1) dan untuk baris 3 running timenya adalah O(1) + T(n -1). Dengan

demikian untuk beberapa konstanta c dan d didapatkan persamaan rekurensi sebagai berikut :

$$T(n) = \begin{cases} d & , \text{jika } n \leq 1 \\ c + T(n-1) & , \text{jika } n > 1 \end{cases} \qquad (2.10)$$

Misalkan $n > 2$, dengan mensubstitusikan $n - 1$ pada n dalam persamaan (2.10) maka didapatkan $T(n-1) = c + T(n-2)$ sedemikian hingga persamaan (2.10) dapat ditulis $T(n) = c + c + T(n-2)$

$= 2c + T(n-2)$, jika $n > 2$

Untuk $n > 3$ dengan menggunakan persamaan (2.10) dapat memperluas $T(n-2)$ sehingga dihasilkan $T(n) = 3c + T(n-3)$, dan seterusnya sehingga secara umum dapat dituliskan :

$T(n) = ic + T(n-i)$, jika $n > i$

jika $i = n - 1$, maka didapatkan

$T(n) = c(n-1) + T(1)$

$= c(n-1) + d \qquad (2.11)$

Berdasarkan persamaan(2.11) dapat disimpulkan running time fungsi fact(n) adalah O(n).

2.7 Pengurutan (Sorting)

Pengurutan diartikan sebagai proses pengurutan kembali sekumpulan obyek ke dalam suatu urutan tertentu. Tujuan pengurutan adalah untuk mendapatkan kemudahan dalam pencarian anggota dari suatu himpunan.

Dalam suatu pengurutan larik (array) terdapat aspek ekonomis yang perlu diperhatikan, meliputi kapasitas pengingat yang tersedia dan waktu yang diperlukan untuk melakukan permutasi sehingga semua elemen akhirnya terurutkan. Ukuran efisiensi yang baik dapat diperoleh dari banyaknya perbandingan dan pemindahan yang dilakukan.

2.7.1 Penggabungan (Merging)

Merging yaitu penggabungan dua kumpulan data yang keduanya dalam keadaan urut menjadi satu kumpulan yang juga dalam keadaan urut. Misalkan terdapat dua buah larik (array) yang akan dimerge sudah dalam keadaan urut naik sebagai berikut :

larik 1

11	23	45	67	68

larik 2

9	12	21	42	56

Ambil elemen pertama dari larik 1 (l_1) dan larik 2 (l_2). Bandingkan nilai kedua elemen ini, jika $l_1 > l_2$, maka l_2 dikopikan ke larik ke-3, jika tidak l_1 dikopikan ke larik ke-3. elemen berikutnya yang dibandingkan adalah elemen yang terletak pada subskrip berikutnya. Dalam contoh di atas ($l_1 = 11$ dan $l_2 = 9$) yang terjadi adalah $l_1 > l_2$ sehingga l_2 dikopikan ke larik ke - 3. Untuk perbandingan berikutnya digunakan $l_1 = 11$ dan $l_2 = 12$. Proses ini

diulang terus sampai salah satu larik habis dikopikan terlebih dahulu. Kemudian larik satunya dikopikan ke larik ke-3. Hasil merging kedua larik di atas adalah sebagai berikut :

9	11	12	21	23	42	45	56	67	68

2.7.2 Mergesort

Metode Mergesort dapat dibedakan menjadi dua macam yaitu

1. Iteratif Mergesort

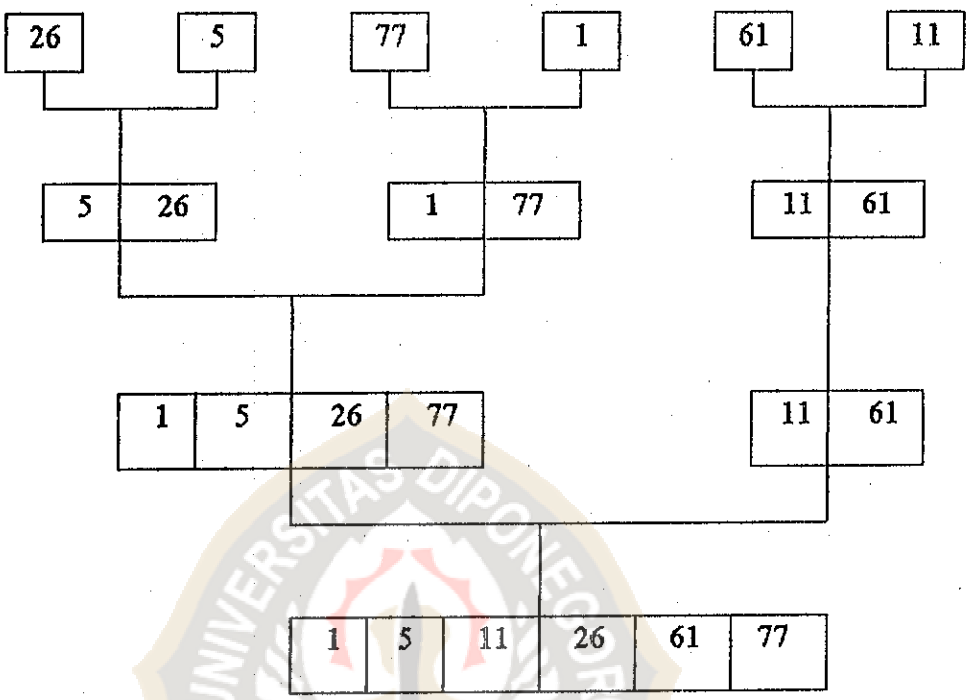
Larik yang mempunyai n elemen dianggap sebagai n buah larik. Untuk setiap pasang larik dilakukan proses merging sehingga akan diperoleh n/2 larik yang masing - masing terdiri dari dua elemen (jika n ganjil, maka terdapat sebuah larik dengan elemen satu). Selanjutnya dilakukan merging kembali untuk setiap pasang larik sehingga diperoleh n/4 buah larik. Langkah ini diteruskan sampai diperoleh sebuah larik yang sudah dalam keadaan urut.

Contoh Larik yang akan diurutkan sebagai berikut

{ 26, 5, 77, 1, 61, 11 }

Penyelesaian

Dengan menggunakan Iteratif Mergesort didapatkan



2. Rekursif Mergesort

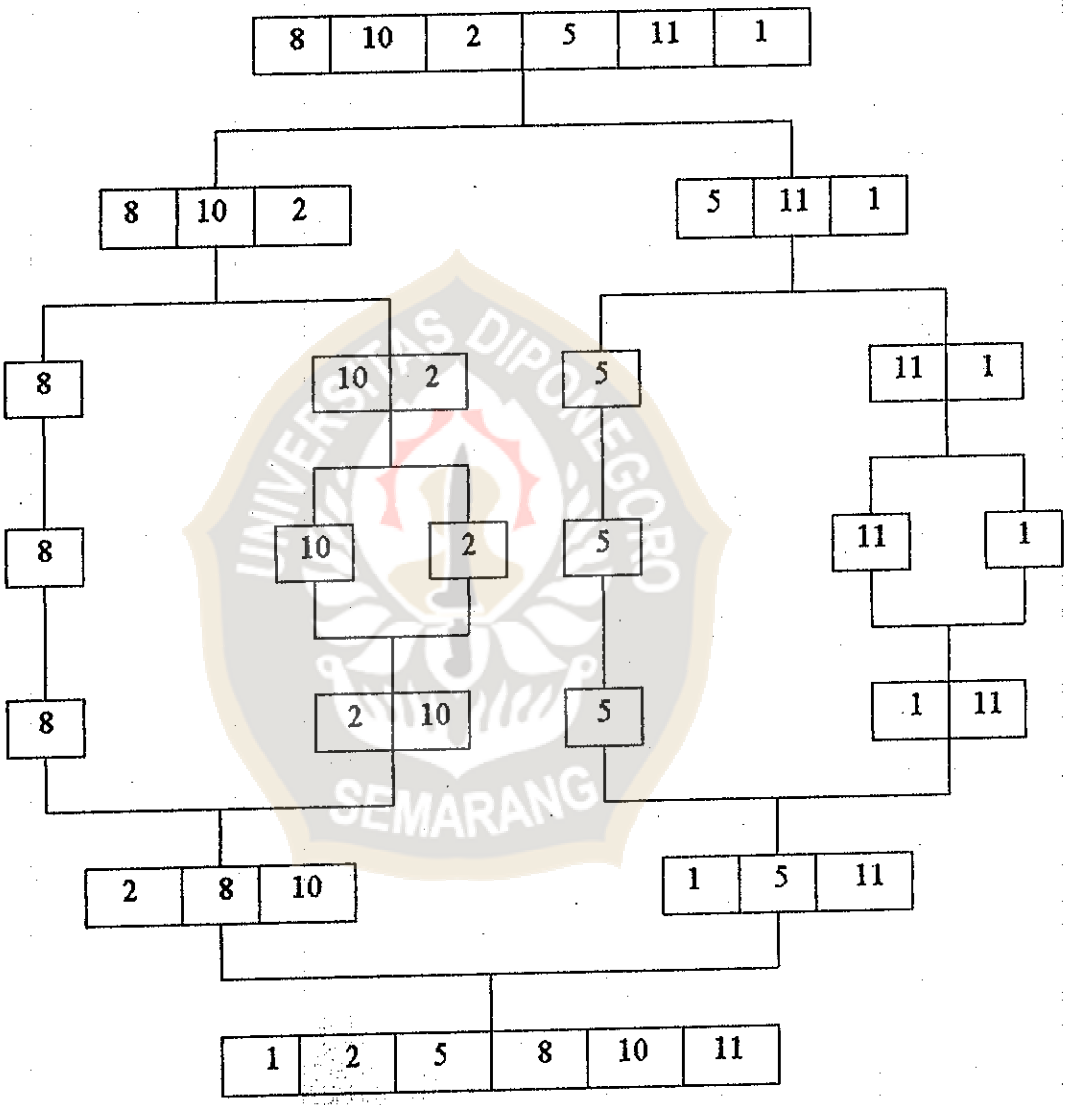
Terdapat barisan dengan n elemen yang ditempatkan dalam suatu array A. Dari array A(1), A(2), ..., A(n) pisahkan menjadi dua bagian sehingga didapatkan A(1), A(2), ..., A(n/2) dan A(n/2 + 1), A(n/2 + 2), ..., A(n). Jika pembagian array A tersebut hasilnya masih terlalu besar, maka setiap bagian tadi dibagi lagi menjadi dua bagian yang lebih kecil sehingga dari setiap bagian akan diurutkan secara rekursif, selanjutnya hasil sortir dari setiap bagian tersebut digabungkan dan diurutkan lagi sedemikian sehingga akan dihasilkan urutan yang tunggal dari n elemen semula.

Contoh : Array yang akan diurutkan adalah sebagai berikut

{ 8, 10, 2, 5, 11, 1}

Penyelesaian

Dengan menggunakan Rekursif Mergesort didapatkan :



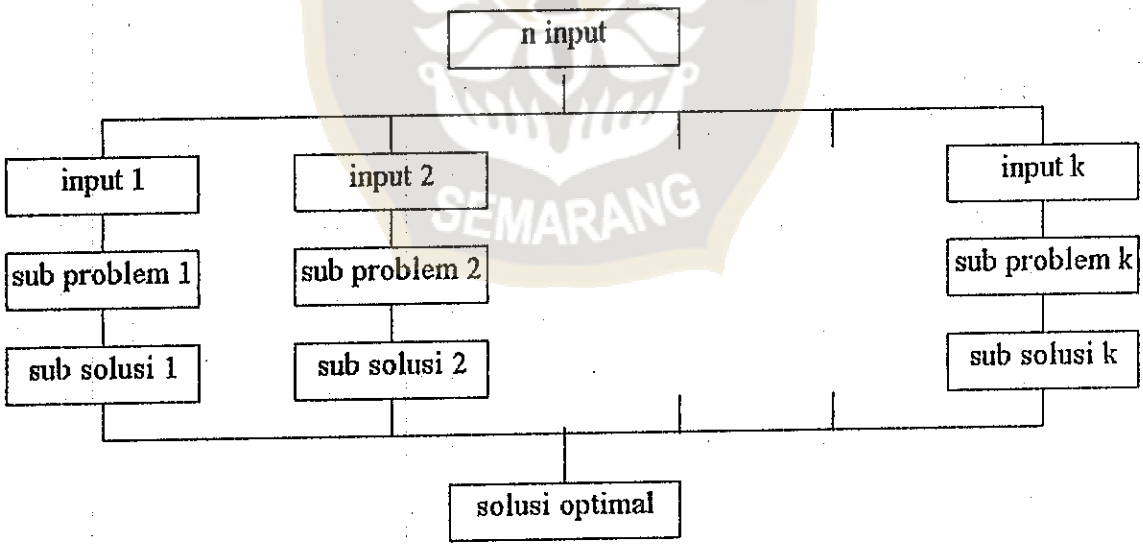
BAB III

ANALISA ALGORITMA MERGESORT

MENGGUNAKAN TEKNIK DIVIDE AND CONQUER (DANDC)

3.1 Teknik Divide and Conquer (DANDC)

Prinsip dasar dari metode ini adalah dengan membagi n input menjadi k sub set input yang berbeda ($1 < k \leq n$) dengan ukuran yang sama. Dari k sub set input yang berbeda akan terdapat k sub problem. Setiap sub problem mempunyai solusi, sehingga akan diperoleh k sub solusi. Selanjutnya dari k sub solusi dikombinasi sehingga diperoleh solusi yang optimal. Bentuk umum teknik divide and conquer adalah sebagai berikut :



gambar 3.1
Bentuk Umum metode DANDC

Jika sub problem ukurannya masih relatif besar (ukuran > 1), maka teknik DANDC dapat digunakan lagi secara rekursif sedemikian sehingga setiap sub problem mempunyai satu elemen. Adapun algoritma teknik DANDC adalah sebagai berikut :

```
Procedure A(D : Data ; Var R : Result ) ;
Var     D1, ..., Dk : Data ;
        R1, ..., Rk : Result ;
Begin
 If simple (D) then R := A0(D)
                Else
                 Begin
                   D1, ..., Dk := partition(D) ;
                   R1 := A(D1) ; ... Rk := A(Dk) ;
                   R := Combine (R1, ..., Rk) ;
                 End
End.
```

Keterangan

- Simple (D) adalah fungsi bernilai boole yang menentukan apakah data (D) cukup kecil sedemikian sehingga solusinya dapat dihitung tanpa pemecahan. Jika demikian maka R := A_0(D) yang akan diproses atau dieksekusi. Pada keadaan lain, D_1, ..., D_k := partition(D) yang akan dieksekusi.
- Fungsi partition(D) mempartisi data (D) menjadi D_1, ..., D_k. Untuk setiap i, A(D_i) merupakan hasil R_i- Fungsi Combine(R_1, ..., R_k) merupakan fungsi yang menentukan solusi umum atau solusi yang diharapkan dari persoalan semula dengan memanfaatkan sub solusi R_1, ..., R_k yang hasilnya adalah R.

Suatu kelas dari metode DANDC memenuhi syarat - syarat berikut :

1. Untuk masalah dengan ukuran n = 1, nilai pemecahan masalah atau running timenya adalah c, dimana c adalah konstanta positif.
2. Masalah - masalah dengan ukuran $n = b^m$, untuk $b > 1$, $m > 0$ dibagi menjadi a sub masalah yang berbeda dengan ukuran n/b.
3. Untuk masalah dengan ukuran $n > 1$, nilai pemecahan masalah atau running timenya adalah hasil penjumlahan dari nilai pembagian masalah menjadi sub masalah dengan nilai penggabungan solusi - solusi dari sub masalah menghasilkan suatu solusi dari masalah semula (h(n)).

Berdasarkan kondisi - kondisi di atas, maka dihasilkan suatu persamaan rekurensi sebagai berikut :

$$T(n) = \begin{cases} c & , \text{untuk } n = 1 \\ a\,T(n/b) + h(n) & , \text{untuk } n > 1 \end{cases} \qquad (3.1)$$

Teorema 3.1.1

Solusi persamaan $T(n) = a\,T(n/b) + \theta(n^k)$, untuk $a \geq 1$ dan $b > 1$ adalah

$T(n) = O(n^k \log n)$ jika $a = b^k$

Bukti

Misalkan $n = b^m$, maka $n/b = b^{m-1}$

$$n^k = (b^m)^k = b^{mk} = b^{km} = (b^k)^m$$

Asumsikan $T(1) = 1$. Dengan menggunakan definisi 2.6.1.3 maka persamaan $T(n) = a\,T(n/b) + \theta(n^k)$ dapat ditulis

$T(n) = a\,T(n/b) + c\,n^k$. Misalkan $c = 1$, maka didapatkan

$T(n) = a\,T(n/b) + n^k$

$$T(b^m) = a\,T(b^{m-1}) + (b^k)^m \qquad (3.2)$$

Jika setiap sisi dibagi dengan a^m, maka didapatkan

$$\frac{T(b^m)}{a^m} = \frac{T(b^{m-1})}{a^{m-1}} + \left[\frac{b^k}{a}\right]^m \qquad (3.3)$$

sehingga persamaan tersebut dapat diaplikasikan untuk nilai m yang lain sebagai berikut :

$$\frac{T(b^{m-1})}{a^{m-1}} = \frac{T(b^{m-2})}{a^{m-2}} + \left[\frac{b^k}{a}\right]^{m-1} \qquad (3.4)$$

$$\frac{T(b^{m-2})}{a^{m-2}} = \frac{T(b^{m-3})}{a^{m-3}} + \left[\frac{b^k}{a}\right]^{m-2} \qquad (3.5)$$

$$\vdots$$

$$\frac{T(b^1)}{a^1} = \frac{T(b^0)}{a^0} + \left[\frac{b^k}{a}\right]^1 \qquad (3.6)$$

Dengan menggunakan persamaan (3.2) sampai (3.6), maka dihasilkan :

$$\frac{T(b^m)}{a^m} = 1 + \sum_{i=1}^{m} \left[\frac{b^k}{a} \right]^i$$

$$= \sum_{i=0}^{m} \left[\frac{b^k}{a} \right]^i \qquad (3.7)$$

Dengan demikian

$$T(n) = a^m \sum_{i=0}^{m} \left[\frac{b^k}{a} \right]^i \text{, untuk } n = b^m \qquad (3.8)$$

Jika $a = b^k$, maka setiap suku dalam jumlahan adalah 1. Dengan demikian persamaan (3.8) dapat ditulis

$T(n) = b^{km} (1 + m)$

$T(n) = n^k (1 + \log_b n)$

$= n^k + n^k \log_b n$

$= O(n^k \log_b n)$

$= O(n^k \log n)$

Jadi terbukti bahwa solusi persamaan $T(n) = aT(n/b) + \theta(n^k)$, untuk $a \geq 1$ dan $b > 1$ adalah $T(n) = O(n^k \log n)$, jika $a = b^k$

■

3.2 Algoritma Mergesort Menggunakan Teknik DANDC

Misalkan terdapat suatu deret dengan n elemen yang ditempatkan dalam suatu array A. Deret tersebut akan diurutkan secara ascending. Dengan menggunakan algoritma mergesort dengan teknik DANDC deret tersebut dibagi menjadi dua bagian yaitu A(1), A(2), ..., A(n/2) dan A(n/2 + 1), A(n/2 + 2), ..., A(n). Jika pembagian deret tersebut hasilnya masih terlalu besar (ukuran setiap bagiannya > 1), maka masing-masing bagian dibagi lagi menjadi dua bagian yang lebih kecil, sedemikian sehingga setiap bagiannya mempunyai satu elemen. Selanjutnya setiap pasang bagian dikombinasi sehingga akan dihasilkan urutan yang tunggal dari n elemen semula.

Algoritma Mergesort dengan teknik DANDC terdiri dari dua prosedur yaitu Prosedur MERGESORT dan Prosedur MERGE. Algoritmanya adalah sebagai berikut :

Masukan : s_i, ..., s_j, i, dan j
Keluaran : s_i, ..., s_j teratur dalam urutan menaik.

```
1.  Procedure MERGESORT (s, i, j, p)
2.  // Kasus dasar i = j //
3.  If  i = j  then
4.          p := i
5.  // Bagilah deret dan urutkan //
6.  m := (i + j) div 2
7.  Call  MERGESORT (s, i, m, q)
8.  Call  MERGESORT (s, m + 1, j, r)
9.  // Gabungkan //
10. Call  MERGE (s, i, m, j, c)
11. End MERGESORT.
```

Masukan : Dua deret menaik s_i, ..., s_m dan s_{m+1}, ..., s_j dengan indeks i, m, j.

Keluaran : Deret s_i, ..., s_j dalam urutan yang menaik.

```
1.  Procedure MERGE (s, i, m, j, c)
2.  // p adalah posisi di deret s_i, ..., s_m. //
3.  // q adalah posisi di deret s_{m+1}, ..., s_j. //
4.  // r adalah posisi di deret c_i, ...,s_j. //
5.  p := i
6.  q := m + 1
7.  r := i
8.  // Salin nilai yang lebih kecil dari s_p dan s_q. //
9.  While p ≤ m dan q ≤ j  do
10. Begin
11.   If s_p < s_q  then
12.         Begin
13.           c_r := s_p
14.           p := p + 1
15.         End
16.   Else
17          Begin
18.           c_r :=  s_q
19.           q := q + 1
20.         End
21.   r := r + 1
22. End
23  // Salin sisa dari deret pertama //
24. While p ≤ m  do
```

```
25.  Begin
26.     c_r := s_p
27.     p := p + 1
28.     r := r + 1
29.  End
30.  // Salin sisa dari deret kedua //
31.  While q ≤ j  do
32.  Begin
33.     c_r := s_q
34.     q := q + 1
35.     r := r + 1
36.  End
37.  // Salinlah c ke dalam s //
38.  For k := i to j  do
39.           s_k := c_k
40.  End MERGE.
```

Teorema 3.2.1

Dua daftar yang telah urut dengan panjang m dan n dapat digabung (dimerge) menjadi satu daftar yang urut memerlukan kurang dari atau sama dengan m + n - 1 perbandingan.

Bukti

Misalkan list 1 dan list 2 adalah dua daftar dengan panjang m dan n, dan keduanya telah terurut secara naik (ascending). Dengan menggunakan algoritma MERGE, masukan - masukan pada bagian awal dari daftar list 1 dan list 2 dibandingkan dan

nilai yang lebih kecil dari kedua masukan tersebut diletakkan pada suatu list yang kosong. Proses ini diulang sampai salah satu dari kedua daftar kosong, dan selanjutnya daftar yang lain (tidak kosong), masukan - masukannya diletakkan pada urutan yang terakhir sampai daftar tersebut kosong. Karena setiap perbandingan dari elemen - elemen list 1 dan elemen - elemen list 2 menghasilkan suatu elemen yang diletakkan pada suatu list yang lain, maka terdapat kurang dari m + n perbandingan. Karena tidak ada perbandingan ketika salah satu list kosong, maka paling banyak terdapat m + n - 1 perbandingan. Dengan demikian algoritma MERGE akan menggabungkan dua daftar yang telah terurut menjadi satu daftar terurut memerlukan kurang dari atau sama dengan m + n - 1 perbandingan diantara elemen - elemen dari kedua list tersebut.

■

3.3 Analisa Algoritma Mergesort Menggunakan Teknik DANDC

Algoritma mergesort terdiri dari dua prosedur yaitu Prosedur MERGESORT dan Prosedur MERGE. Analisa algoritma mergesort menggunakan teknik DANDC dilakukan hanya pada Prosedur MERGESORT. Berdasarkan aturan umum analisa algoritma dan analisa struktur kontrol maka analisa algoritma mergesort dapat diuraikan sebagai berikut :

1. Pada baris ke-1 merupakan nama dari prosedur. Karena hanya berupa nama prosedur, maka baris tersebut tidak mempengaruhi analisa.

2. Pada baris ke-2 terdapat suatu komentar yang menyatakan suatu pengujian apakah i = j. Karena hanya berupa komentar, maka baris tersebut tidak berpengaruh terhadap analisa.
3. Pada baris ke-3 terdapat statemen kondisi if, sehingga running timenya adalah harga pada kondisi statemen yang dieksekusi. Jika statemen kontrol terpenuhi, maka running timenya adalah running time statemen atau blok statemen sesudah then (prosedur akan mengeksekusi baris ke-4) dan jika statemen kontrol tidak terpenuhi, maka running timenya adalah running time pada pengujian statemen kontrol yaitu O(1).
4. Pada baris ke-4 terdapat suatu statemen pemberian yaitu p := i yang dieksekusi ketika daftar mempunyai panjang 1 atau i = j, Sehingga running timenya adalah O(1).
5. Pada baris ke-5 terdapat suatu komentar yang menyatakan deret dibagi dan disortir. Karena hanya berupa komentar, maka hal ini tidak mempengaruhi analisa.
6. Pada baris ke-6 terdapat statemen pemberian, yaitu pemberian harga pada suatu variabel m = (i + j) div 2. Karena berupa statemen pemberian, maka running timenya adalah O(1).
7. Pada baris ke-7 dan ke-8 terdapat pemanggilan Prosedur MERGESORT secara rekursif yaitu MERGESORT (s, i, m, q) dan MERGESORT (s, m+1, j, r). Karena kedua Prosedur MERGESORT masing - masing mempunyai daftar dengan panjang n/2, maka running time untuk setiap pemanggilan prosedur tersebut adalah T(n/2). Dengan demikian running time pemanggilan secara rekursif kedua Prosedur MERGESORT tersebut adalah 2 T(n/2).

8. Pada baris ke-9 terdapat suatu komentar gabungkan. Karena hanya berupa komentar maka tidak berpengaruh terhadap analisa.
9. Pada baris ke-10 terdapat pemanggilan Prosedur MERGE (s, i, m, j, c). Karena terdapat dua sub daftar dengan panjang masing - masing n/2, maka berdasarkan teorema 3.2.1 dibutuhkan paling banyak n/2 + n/2 + 1 = n - 1 perbandingan, sehingga running time untuk mengeksekusi Prosedur MERGE adalah O(n).
10. Pada baris ke-11 merupakan akhir dari prosedur MERGESORT dengan waktu O(1).

Misalkan T(n) adalah worst case running time prosedur MERGESORT. Dengan demikian untuk beberapa konstanta c_1 dan c_2 dapat dituliskan persamaan rekurensi sebagai berikut :

$$T(n) = \begin{cases} c_1 & , \text{jika } n = 1 \\ 2\,T(n/2) + c_2 n & , \text{jika } n > 1 \end{cases} \qquad (3.9)$$

Keterangan

Suku c_1 pada persamaan (3.9) merupakan suatu konstanta dari langkah - langkah yang ditempuh ketika daftar mempunyai panjang 1. Pada kasus n > 1, waktu yang ditempuh oleh Prosedur MERGESORT dapat dibagi menjadi dua bagian yaitu

1. Pemecahan daftar menjadi dua bagian yang sama yang memerlukan waktu O(1) dan pemanggilan secara rekursif dua Prosedur MERGESORT dengan panjang daftar masing-masing n/2 dengan waktu tempuh 2 T(n/2), sehingga berdasarkan aturan penjumlahan waktu tempuh kedua proses tersebut adalah 2 T(n/2).

2. Pemanggilan Prosedur MERGE yang memerlukan waktu O(n).

Misalkan waktu yang dibutuhkan untuk mengeksekusi Prosedur MERGE c_2n. Dengan berdasarkan aturan penjumlahan (rule of sums), maka waktu yang ditempuh oleh Prosedur MERGESORT untuk $n > 1$ adalah $2\,T(n/2) + c_2n$.

Persamaan rekurensi (3.9) dapat diselesaikan dengan dua cara yaitu menggunakan metode pemecahan persamaan rekurensi secara iteratif atau menggunakan teorema 3.1.1.

1. Pemecahan persamaan rekurensi secara iteratif.

Misalkan untuk $n = 1$, maka $c_1 = T(n) = T(1) = 1$

Untuk $n > 1$, maka $T(n) = 2\,T(n/2) + c_2n$. Dengan memisalkan $c_2 = 1$, maka didapatkan

$$T(n) = 2\,T(n/2) + n \qquad (3.10)$$

Dengan mengganti n dengan n/2, maka persamaan (3.10) menjadi :

$$T(n/2) = 2\,T(n/4) + n/2$$

Jika masing - masing ruas dikalikan dengan dua, maka didapatkan :

$$2\,T(n/2) = 2\,(\,2\,T(n/4) + n/2\,) = 4\,T(n/4) + n$$

Dengan demikian persamaan (3.10) dapat ditulis kembali sebagai berikut :

$$T(n) = 4\,T(n/4) + 2\,n \qquad (3.11)$$

Dengan mengganti n dengan n/4, maka persamaan (3.10) dapat ditulis :

$$T(n/4) = 2\,T(n/8) + n/4$$

Jika masing - masing ruas dikalikan dengan 4, maka didapatkan :

$$4\,T(n/4) = 4\,(\,2\,T(n/8) + n/4\,)$$

$$= 8\,T(n/8) + n$$

Dengan demikian persamaan (3.11) dapat ditulis kembali sebagai berikut :

$T(n) = 8\ T(n/8) + n + 2n$

$= 8\ T(n/8) + 3n$

Jika langkah tersebut diteruskan, maka akan diperoleh bentuk umum :

$T(n) = 2^k\ T(n/2^k) + k\ n \quad , k = 1, 2, 3, \ldots$ (3.12)

Dengan memisalkan $k = \log_2 n$, maka persamaan (3.12) dapat ditulis :

$T(n) = 2^{\log_2 n}\ T(n / 2^{\log_2 n}) + \log_2 n\ .\ n$

$= n\ T(n/n) + n \log_2 n$

$= n\ T(1) + n \log_2 n$

$= n + n \log_2 n$

Dengan demikian berdasarkan aturan penjumlahan, running time Prosedur MERGESORT menggunakan teknik Divide and Conquer adalah O(n log n).

2. Menggunakan teorema 3.1.1

Misalkan untuk $n = 1$, maka $c_1 = T(n) = T(1) = 1$

Untuk $n > 1$, maka $T(n) = 2\ T(n/2) + c_2 n$.

Misalkan $c_2 n = \theta(n)$, sedemikian sehingga didapatkan

$T(n) = 2\ T(n/2) + \theta(n)$, untuk $n > 1$

Berdasarkan teorema 3.1.1 didapatkan $a = b = 2$, $k = 1$ sehingga memenuhi syarat $a = b^k$. Dengan demikian $T(n) = O(n \log n)$.

BAB IV

PROGRAM GRAFIK RUNNING TIME MERGESORT

Secara keseluruhan inti dari program adalah melakukan pengurutan sejumlah data dari data ke-1 (disimpan dalam variabel awal) sampai data ke-n (disimpan dalam variabel jml_dt) secara ascending. Selanjutnya waktu yang diperlukan untuk mengurutkan data tersebut digambarkan dalam sebuah grafik dengan sumbu x menunjukkan data dan sumbu y menunjukkan waktu yang dibutuhkan komputer untuk mengeksekusi (dalam 1/100 detik).

Algoritma program grafik running time mergesort dapat diuraikan sebagai berikut :

1. Membersihkan layar.
2. Memasukkan empat buah item yaitu :
   a. Random angka yaitu dengan memasukkan bilangan tertentu maka bilangan random yang dibangkitkan dengan fungsi random adalah antara nol sampai bilangan tersebut dikurangi satu.
   b. Banyak data yaitu jumlah maksimal data yang akan diurutkan.
   c. Awal ploting data pada grafik yaitu jumlah data mula - mula yang akan digambarkan pada grafik.
   d. Step ploting data yaitu jeda atau sela dari data yang akan digambar pada grafik.

3. Penginisialisasian grafik.

Tujuan dari langkah ini adalah menginisialisasi komputer ke mode grafik. Pada inisialisasi ini akan di tentukan :

a. Driver yang digunakan pada komputer. Jika menggunakan konstanta detect maka program akan menyesuaikan dengan driver yang digunakan oleh komputer tersebut.

b. Mode grafik yang dipergunakan untuk driver yang bersangkutan. Jika menggunakan auto detection, maka mode grafik akan dipilih secara otomatis oleh pascal.

c. Direktori letak dari driver grafik pada disk atau hard disk.

Penginisialisasian grafik akan dieksekusi dengan menggunakan prosedur inisialisasi. Untuk memenuhi semua keperluan dalam prosedur inisialisasi, harus mengaktifkan unit graph pada awal program.

4. Pembuatan perlengkapan grafik meliputi :

a. Prosedur batas yang bertujuan untuk menentukan batas bawah dari grafik.

b. Prosedur layar yang bertujuan memberikan latar belakang dan warna garis tepi layar pada grafik.

c. Prosedur sumbu xy. Tujuannya adalah membuat sumbu x dan y serta meletakkan suatu pointer pada titik pusat sumbu xy.

d. Prosedur skala sumbu x. Tujuannya adalah memberi skala pada sumbu x sekaligus bilangannya sesuai dengan awal pengeplotan grafik dan jumlah data seluruhnya. Pada prosedur ini juga ditampilkan judul sumbu x dan grafik.

e. Prosedur skala sumbu y yang bertujuan memberi skala pada sumbu y dan juga ditampilkan judul sumbu y.

5. Pengeplotan grafik.

Pada bagian ini diawali dengan penginisialisasian pembangkit bilangan random dengan prosedur standart randomize yang selanjutnya disimpan dalam variabel array s. Dengan variabel array s yang telah terisi data random tersebut, maka pengeplotan grafik dapat dilakukan. Pengeplotan grafik ini berdasarkan jumlah data sebagai absis dan waktu untuk melakukan pengurutan sebagai ordinatnya. Jumlah data yang akan diplotkan pada grafik adalah jumlah data mula - mula sampai data yang terakhir dengan jeda tertentu. Waktu komputer untuk melakukan pengurutan data dilakukan dengan mengatur waktu pada komputer menjadi nol yaitu jam 00, menit 00, detik 00, dan seperseratus detik 00. Selanjutnya waktu terakhir setelah pengurutam diambil dengan fungsi gettime sebagai waktu komputer melakukan proses pengurutan. Adapun proses pengurutan data dilakukan dengan menggunakan prosedur MERGESORT dan Prosedur MERGE.

6. Menutup mode grafik pada layar dan merubah ke mode layar dengan prosedur close graph. Untuk mengakhiri program dengan menekan sebarang tombol pada papan ketik.

KESIMPULAN

Dalam penulisan tugas akhir ini dapat ditarik kesimpulan :

1. Teknik Divide and Conquer lebih mudah dideskripsikan jika pemecahan masalah menjadi sub masalah mempunyai ukuran yang sama.
2. Algoritma mergesort dapat didesain menggunakan teknik Divide and Conquer dimana setiap n data dibagi menjadi dua bagian dengan bagian pertama sama dengan n div 2, sedangkan bagian kedua adalah n - (n div 2). Selanjutnya proses tersebut diulang sehingga setiap bagian mempunyai satu elemen.
3. Analisa algoritma mergesort menggunakan teknik Divide and Conquer menghasilkan running time O(n log n).

DAFTAR PUSTAKA

Aho, A.V, J.E. Hopcroft, dan J.D. Ullman : " The Design and Analysis of Computer Algorithms ", Addison Wesley Publishing Company, Reading Mass, 1974.

Horowits, E. dan S. sahni : " Fundamentals of Data Structures in Pascal ", Computer Science Press, New York, 1982.

Mark Allen Weiss : " Data Structures and Algorithms Analysis in C++ ", Florida International University.

Richard Johnsonbaugh : " Matematika Diskrit ", Edisi Bahasa Indonesia, Jilid 1, Simon dan Schuster (Asia) Pte.Ltd, 1998.

Stanat, D.F, dan D.F. McAllister : " Discrete Mathematics in Computer Science ", Prentice Hall, Enggliwood Cliffs, N.J, 1977.

Suryadi M.T : " Pengantar Analisis Algoritma :, Penerbit Gunadarma, 1994.

```
{#-------------------------------------------------------------------------#
 #  JUDUL PROGRAM      : GRAFIK RUNNING TIME MERGESORT                      #
 #  DISUSUN OLEH       : ARIEF TEGUH RAHARDJO                               #
 #  N I M              : J 101 95 1180                                      #
 #  KEPENTINGAN        : TUGAS AKHIR                                        #
 #  DIBUAT             : NOVEMBER 2000                                      #
 #-------------------------------------------------------------------------# }

program Running_Time_Mergesort ;
{$M 65000,0,655360}
uses crt,graph,dos ;
type alist = array [1..2500] of integer ;
var  s : alist ;
   jml_dt,wkt_max,y,randm,z,n,b,jml,awl,bts_bwh,step,waktu : integer ;
   jam,menit,sec,secper100 : word ;
   sbx,sby : real ;

procedure inisialisasi ;
var gd,gm : integer ;
begin
 gd := detect ;
 gm := VGAHi ;
 initgraph (gd,gm,' ') ;
 if graphresult <> grok then
 halt(1) ;
end ;

procedure batas ;
begin
 bts_bwh := round(getmaxy*0.65) ;
end;
procedure layar ;
begin
 setlinestyle(0,0,3) ;
 setcolor(blue) ;
 setbkcolor(white) ;
 rectangle(3,3,getmaxx-3,getmaxy-3) ;
end;

procedure sumbuxy ;
begin
 setlinestyle(0,0,3) ;
 setcolor(red) ;
 line(80,5,80,bts_bwh+10) ;
 moveto(80,5) ;
 lineto(75,10) ;
 lineto(85,10) ;
 lineto(80,5) ;
 line(80,bts_bwh+10,getmaxx-20,bts_bwh+10) ;
```

```
 moveto(getmaxx-20,bts_bwh+10) ;
 lineto(getmaxx-25,bts_bwh+5) ;
 lineto(getmaxx-25,bts_bwh+15) ;
 lineto(getmaxx-20,bts_bwh+10) ;
 moveto(80,bts_bwh+10) ;
end ;

procedure skalasbx(mulai,usai :integer) ;
var s:integer ; noskx,judul1,judul2,judul3:string ;
begin
 for s:=1 to 11 do
  begin
   setcolor(red) ;
   outtextxy(77+(s-1)*52,bts_bwh+8,'I') ;
   str(mulai+((usai-mulai) div 10)*(s-1),noskx) ;
   outtextxy(60+(s-1)*52,bts_bwh+20,noskx) ;
  end ;
 setcolor(red) ;
 judul1:='Jumlah Data';
 judul2:='Gambar : Grafik Running Time Mergesort';
 judul3:='              Menggunakan Teknik DANDC';
 settextstyle(2,0,5) ;
 outtextxy(300,bts_bwh+40,judul1) ;
 settextstyle(2,0,6) ;
 outtextxy(150,bts_bwh+70,judul2) ;
 outtextxy(150,bts_bwh+90,judul3) ;
 settextstyle(0,0,1) ;
end ;

procedure skalasby ;
var s:integer ; tot:real ;
    nosky,judul3:string ;
begin
 wkt_max := bts_bwh + 10 - 10 ;
 step := bts_bwh div 6 ;
 for s:=1 to 6 do
  begin
   outtextxy(80,bts_bwh+10-s*step,'-') ;
   str(s*500,nosky) ;
   outtextxy(40,bts_bwh+10-s*step,nosky) ;
   setcolor(red) ;
  end ;
 setcolor(red) ;
 judul3:='Dalam 1/100 detik' ;
 settextstyle(2,1,4) ;
 outtextxy(15,bts_bwh-200,judul3) ;
 settextstyle(0,0,1) ;
 moveto(80,bts_bwh+10) ;
end ;
```

```
procedure merge(var x: alist; l,m,n : integer; var z : alist) ;
var i,j,k,t : integer ;
begin
 i:=l ;
 k:=l ;
 j:=m+1 ;
 while ((i <= m) and (j <= n)) do
  begin
   if x[i] < x[j] then
    begin
     z[k]:=x[i] ;
     i:=i+1 ;
    end
   else
    begin
     z[k]:=x[j] ;
     j:=j+1
    end;
   k:=k+1
  end ;
 if i > m then
  for t := j to n do z[k+t-j] := x[t]
 else
  for t := i to m do z[k+t-i] := x[t] ;
end ;

procedure merge_sort(var x : alist; l,u : integer; var p : integer) ;
var i,mid,q,r : integer ;  y:alist ;
begin
 if l = u then
  p:=l
 else
  begin
   mid := (l+u) div 2 ;
   merge_sort(x,l,mid,q) ;
   merge_sort(x,mid+1,u,r) ;
   merge(x,l,mid,u,y) ;
  end;
end;

procedure tutup ;
begin
 writeln('') ;
 settextstyle(2,0,2) ;
 settextjustify(1,2) ;
 outtextxy(getmaxx div 2,19*getmaxy div 20,'Tekan <Esc>') ;
 repeat until readkey = #27 ;
 closegraph ;
end ;
```

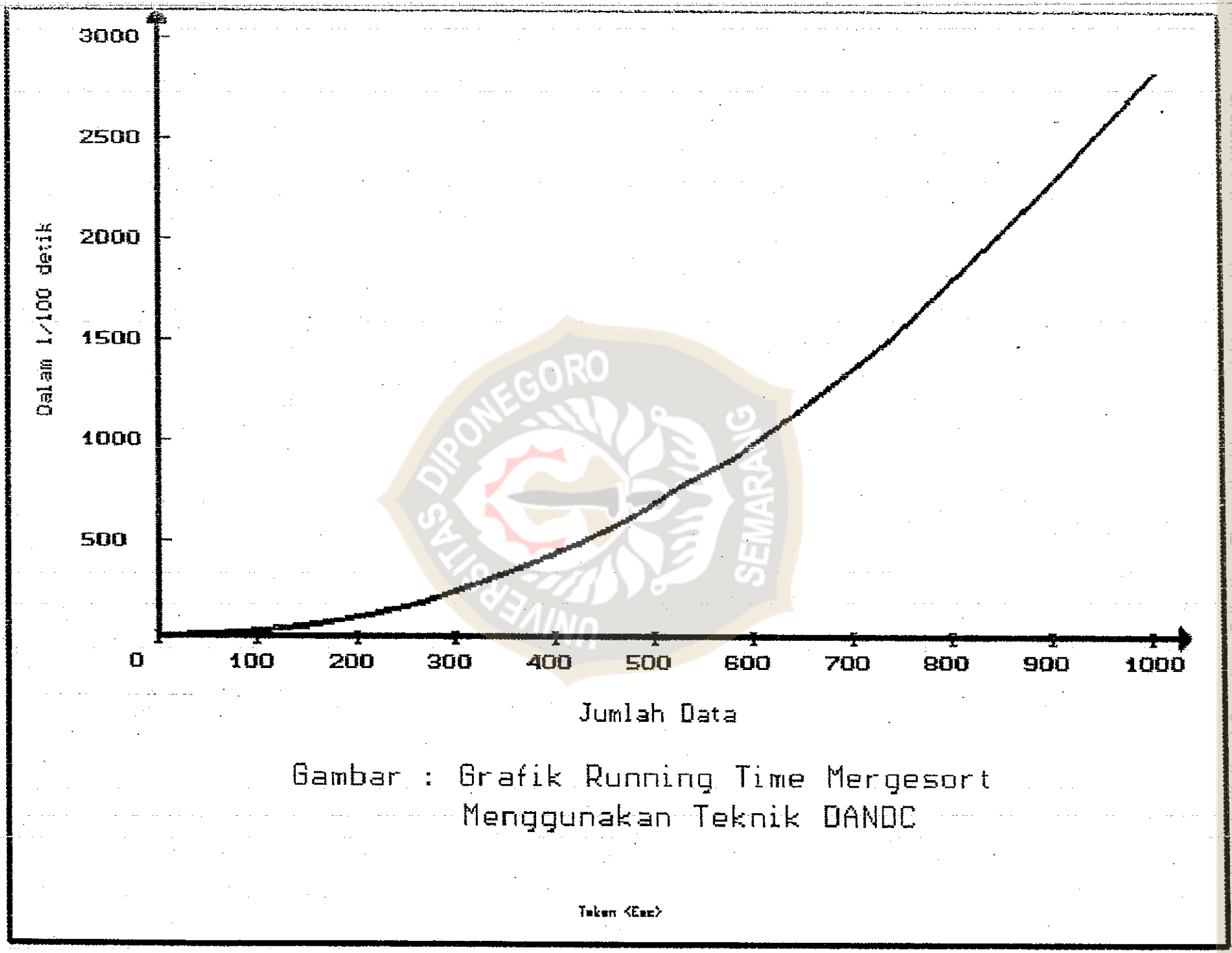
```
{ ** ----------<<<<<<        PROGRAM UTAMA        >>>>>>----------** }
begin
 textbackground(0) ; clrscr ;
 write('Random angka 0..n                 = ');readln(randm) ;
 write('Banyaknya data (maximal 2500)  = ');readln(jml_dt) ;
 write('Awal ploting data pada grafik     = ');readln(awl) ;
 write('Step ploting data                 = ');readln(step) ;
 y:=awl ;
 repeat
  randomize ;
  for z:=1 to y do s[z]:=random(randm) ;
  settime(0,0,0,0) ;
  for z := 1 to y do merge_sort(s,1,y,b) ;
  gettime(jam,menit,sec,secper100) ;
  waktu:=jam*60*60*100+menit*60*100+sec*100+secper100 ;
  writeln('Waktu dengan jumlah data ',y,' adalah ',waktu,' per 100 detik') ;
  y:=y+step ;
 until y>jml_dt ;
 writeln;writeln;writeln;
 writeln(' UNTUK MENAMPILKAN GRAFIK  <<<  Tekan sebarang tombol  >>>') ;
 repeat until keypressed ;
 inisialisasi ; batas ; layar ; sumbuxy ;
 skalasbx(awl,jml_dt) ;
 skalasby ;
 y:=awl ;
 repeat
  if y >= awl then
  begin
   randomize ;
   for z := 1 to y do s[z] := random(randm) ;
   settime(0,0,0,0) ;
   for z := 1 to y do merge_sort(s,1,y,b) ;
   gettime(jam,menit,sec,secper100) ;
   waktu := jam*60*60*100+menit*60*100+sec*100+secper100 ;
   sbx := (y-awl)*((560)/(jml_dt-awl)) ;
   if waktu <= 3000 then
    begin
     sby := wkt_max/3000*waktu ;
     setcolor(blue);setlinestyle(0,0,3) ;
     lineto(round(sbx)+80,(bts_bwh+10)-round(sby)) ;
    end
   else
    y := jml_dt ;
   y := y + step ;
  end ;
 until y > jml_dt ;
 tutup ;
end.
```

Random angka 0..n = 1000
Banyaknya data (maximal 2500) = 1000
Awal ploting data pada grafik = 0
Step ploting data = 10

Waktu dengan jumlah data 0	adalah	0	per 100 detik
Waktu dengan jumlah data 10	adalah	1	per 100 detik
Waktu dengan jumlah data 20	adalah	2	per 100 detik
Waktu dengan jumlah data 30	adalah	4	per 100 detik
Waktu dengan jumlah data 40	adalah	5	per 100 detik
Waktu dengan jumlah data 50	adalah	5	per 100 detik
Waktu dengan jumlah data 60	adalah	7	per 100 detik
Waktu dengan jumlah data 70	adalah	10	per 100 detik
Waktu dengan jumlah data 80	adalah	13	per 100 detik
Waktu dengan jumlah data 90	adalah	16	per 100 detik
Waktu dengan jumlah data 100	adalah	21	per 100 detik
Waktu dengan jumlah data 110	adalah	27	per 100 detik
Waktu dengan jumlah data 120	adalah	32	per 100 detik
Waktu dengan jumlah data 130	adalah	38	per 100 detik
Waktu dengan jumlah data 140	adalah	43	per 100 detik
Waktu dengan jumlah data 150	adalah	54	per 100 detik
Waktu dengan jumlah data 160	adalah	60	per 100 detik
Waktu dengan jumlah data 170	adalah	71	per 100 detik
Waktu dengan jumlah data 180	adalah	80	per 100 detik
Waktu dengan jumlah data 190	adalah	87	per 100 detik
Waktu dengan jumlah data 200	adalah	98	per 100 detik
Waktu dengan jumlah data 210	adalah	104	per 100 detik
Waktu dengan jumlah data 220	adalah	120	per 100 detik
Waktu dengan jumlah data 230	adalah	137	per 100 detik
Waktu dengan jumlah data 240	adalah	148	per 100 detik
Waktu dengan jumlah data 250	adalah	159	per 100 detik
Waktu dengan jumlah data 260	adalah	175	per 100 detik
Waktu dengan jumlah data 270	adalah	186	per 100 detik
Waktu dengan jumlah data 280	adalah	203	per 100 detik
Waktu dengan jumlah data 290	adalah	219	per 100 detik
Waktu dengan jumlah data 300	adalah	236	per 100 detik
Waktu dengan jumlah data 310	adalah	258	per 100 detik
Waktu dengan jumlah data 320	adalah	269	per 100 detik
Waktu dengan jumlah data 330	adalah	285	per 100 detik
Waktu dengan jumlah data 340	adalah	307	per 100 detik
Waktu dengan jumlah data 350	adalah	324	per 100 detik
Waktu dengan jumlah data 360	adalah	346	per 100 detik
Waktu dengan jumlah data 370	adalah	362	per 100 detik
Waktu dengan jumlah data 380	adalah	384	per 100 detik
Waktu dengan jumlah data 390	adalah	411	per 100 detik
Waktu dengan jumlah data 400	adalah	433	per 100 detik
Waktu dengan jumlah data 410	adalah	450	per 100 detik
Waktu dengan jumlah data 420	adalah	477	per 100 detik
Waktu dengan jumlah data 430	adalah	505	per 100 detik
Waktu dengan jumlah data 440	adalah	532	per 100 detik
Waktu dengan jumlah data 450	adalah	560	per 100 detik
Waktu dengan jumlah data 460	adalah	576	per 100 detik
Waktu dengan jumlah data 470	adalah	615	per 100 detik
Waktu dengan jumlah data 480	adalah	631	per 100 detik
Waktu dengan jumlah data 490	adalah	648	per 100 detik
Waktu dengan jumlah data 500	adalah	681	per 100 detik

Waktu dengan jumlah data 510	adalah	714	per 100 detik
Waktu dengan jumlah data 520	adalah	741	per 100 detik
Waktu dengan jumlah data 530	adalah	757	per 100 detik
Waktu dengan jumlah data 540	adalah	812	per 100 detik
Waktu dengan jumlah data 550	adalah	834	per 100 detik
Waktu dengan jumlah data 560	adalah	878	per 100 detik
Waktu dengan jumlah data 570	adalah	895	per 100 detik
Waktu dengan jumlah data 580	adalah	939	per 100 detik
Waktu dengan jumlah data 590	adalah	966	per 100 detik
Waktu dengan jumlah data 600	adalah	1010	per 100 detik
Waktu dengan jumlah data 610	adalah	1043	per 100 detik
Waktu dengan jumlah data 620	adalah	1087	per 100 detik
Waktu dengan jumlah data 630	adalah	1114	per 100 detik
Waktu dengan jumlah data 640	adalah	1147	per 100 detik
Waktu dengan jumlah data 650	adalah	1186	per 100 detik
Waktu dengan jumlah data 660	adalah	1230	per 100 detik
Waktu dengan jumlah data 670	adalah	1268	per 100 detik
Waktu dengan jumlah data 680	adalah	1307	per 100 detik
Waktu dengan jumlah data 690	adalah	1345	per 100 detik
Waktu dengan jumlah data 700	adalah	1395	per 100 detik
Waktu dengan jumlah data 710	adalah	1439	per 100 detik
Waktu dengan jumlah data 720	adalah	1477	per 100 detik
Waktu dengan jumlah data 730	adalah	1548	per 100 detik
Waktu dengan jumlah data 740	adalah	1603	per 100 detik
Waktu dengan jumlah data 750	adalah	1620	per 100 detik
Waktu dengan jumlah data 760	adalah	1647	per 100 detik
Waktu dengan jumlah data 770	adalah	1697	per 100 detik
Waktu dengan jumlah data 780	adalah	1757	per 100 detik
Waktu dengan jumlah data 790	adalah	1807	per 100 detik
Waktu dengan jumlah data 800	adalah	1872	per 100 detik
Waktu dengan jumlah data 810	adalah	1912	per 100 detik
Waktu dengan jumlah data 820	adalah	1944	per 100 detik
Waktu dengan jumlah data 830	adalah	1993	per 100 detik
Waktu dengan jumlah data 840	adalah	2032	per 100 detik
Waktu dengan jumlah data 850	adalah	2103	per 100 detik
Waktu dengan jumlah data 860	adalah	2136	per 100 detik
Waktu dengan jumlah data 870	adalah	2218	per 100 detik
Waktu dengan jumlah data 880	adalah	2229	per 100 detik
Waktu dengan jumlah data 890	adalah	2306	per 100 detik
Waktu dengan jumlah data 900	adalah	2372	per 100 detik
Waktu dengan jumlah data 910	adalah	2433	per 100 detik
Waktu dengan jumlah data 920	adalah	2455	per 100 detik
Waktu dengan jumlah data 930	adalah	2532	per 100 detik
Waktu dengan jumlah data 940	adalah	2554	per 100 detik
Waktu dengan jumlah data 950	adalah	2685	per 100 detik
Waktu dengan jumlah data 960	adalah	2713	per 100 detik
Waktu dengan jumlah data 970	adalah	2757	per 100 detik
Waktu dengan jumlah data 980	adalah	2834	per 100 detik
Waktu dengan jumlah data 990	adalah	2883	per 100 detik
Waktu dengan jumlah data 1000	adalah	2960	per 100 detik

UNTUK MENAMPILKAN GRAFIK <<< Tekan sebarang tombol >>>



Gambar : Grafik Running Time Mergesort
Menggunakan Teknik DANDC

Tekan <Esc>