ALGORITMA POSTMAN'S SORT UNTUK 4 DIGIT DENGAN BAHASA PEMROGRAMAN PASCAL

SKRIPSI

Oleh

FARID YUWONO

J2A 096 022

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS DIPONEGORO SEMARANG

2003

ALGORITMA POSTMAN'S SORT UNTUK 4 DIGIT DENGAN BAHASA PEMROGRAMAN PASCAL

S K R I P S I

Diajukan sebagai salah satu syarat memperoleh
Gelar Sarjana Strata Satu (S1) pada Jurusan Matematika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Diponegoro

Oleh

FARID YUWONO

J2A 096 022

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS DIPONEGORO SEMARANG
2003

HALAMAN PENGESAHAN

Lembar 1

JUDUL : ALGORITMA POSTMAN'S SORT UNTUK 4 DIGIT DENGAN BAHASA PEMROGRAMAN PASCAL

NAMA : FARID YUWONO

NIM : J2A 096 022

Telah dinyatakan lulus ujian sarjana pada tanggal 14 Agustus 2003

Semarang, 14 Agustus 2003

Panitia Penguji Ujian Sarjana

Jurusan Matematika

Ketua,

Drs. Kushartantya, MIKom
NIP. 130 805 062

Ketua Jurusan Matematika,

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

HALAMAN PENGESAHAN

Lembar 2

JUDUL : ALGORITMA POSTMAN'S SORT UNTUK 4 DIGIT DENGAN BAHASA PEMROGRAMAN PASCAL

NAMA : FARID YUWONO

NIM : J2A 096 022

Telah selesai dan layak mengikuti ujian sarjana

Semarang, 14 Agustus 2003

Pembimbing Utama,

Drs. Kushartantya, MIKom
NIP. 130 805 062

Pembimbing Anggota,

Drs. Putut Sri Wasito
NIP. 130 877 410

HALAMAN PERSEMBAHAN

Tugas akhir ini saya persembahkan untuk

: Alm. Bapak dan Ibu,
keluargaku tercinta.

KATA PENGANTAR

Puji syukur Penulis panjatkan ke hadirat Allah SWT karena dengan rahmat, hidayah dan karunia-NYA Penulis dapat menyelesaikan penyusunan Tugas Akhir dengan judul **"ALGORITMA POSTMAN'S SORT UNTUK 4 DIGIT DENGAN BAHASA PEMROGRAMAN PASCAL".**

Tugas Akhir ini disusun guna melengkapi syarat untuk mendapatkan gelar Sarjana Strata Satu pada Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Diponegoro.

Penulis menyadari bahwa penyusunan Tugas Akhir ini tidak dapat berjalan dengan baik dan lancar apabila tidak adanya dukungan dari berbagai pihak. Untuk itu, Penulis menyampaikan terima kasih kepada :

1. Bapak Drs. Bayu Surarso, MSc. PhD. sebagai Ketua Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Diponegoro.
2. Bapak Drs. Kushartantya, MIKom. sebagai Pembimbing Utama yang telah memberikan pengarahan kepada Penulis.
3. Bapak Drs. Putut Sri Wasito. sebagai Pembimbing Anggota yang telah sabar membimbing Penulis selama penyusunan skripsi ini.
4. Ibu Dra. Dwi Ispriyanti, MSi. sebagai dosen wali yang telah banyak membantu Penulis.

5. Para dosen dan staf Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Diponegoro yang memberikan dukungan dalam penyusunan skripsi ini.
6. Alm. Bapak dan Ibu serta kakak-kakakku tercinta yang selalu mendoakan dan mendukung keberhasilan ini.
7. Teman-teman angkatan 96 dan sahabat-sahabatku tersayang yang telah memberikan banyak bantuan hingga selesainya tulisan ini.

Semoga tulisan ini bermanfaat untuk pengembangan sains khususnya bidang komputer dan matematika serta berguna bagi pembaca.

Semarang, 14 Agusutus 2003

Penulis

ABSTRAK

Algoritma Postman's sort adalah suatu prosedur untuk mengurutkan data, yang pertama kali dicetuskan oleh Robert Ramey. Algoritma ini dimulai dengan mengurutkan bilangan pada digit terdepan, selanjutnya digit disebelah kanannya. Demikian seterusnya sehingga untuk m digit dan n data diperlukan m iterasi dengan kompleksitas waktu O(mn). Selanjutnya algoritma tersebut diimplementasikan dalam bentuk program dengan bahasa pemrograman Pascal.

ABSTRACT

The Postman's sort algorithm is a sorting procedure, which discovered by Robert Ramey. This algorithm is started by sorting the number on first digit, then move to right side of digit. So, there are m iteration for m digit and n data. And then, that algorithm is implemented in Pascal programming language.

DAFTAR ISI

HALAMAN JUDUL i

HALAMAN PENGESAHAN ii

HALAMAN PERSEMBAHAN iv

KATA PENGANTAR v

ABSTRAK vii

ABSTRACT viii

DAFTAR ISI ix

DAFTAR GAMBAR xi

DAFTAR TABEL xii

BAB I PENDAHULUAN 1

1.1. Latar Belakang 1

1.2. Permasalahan 1

1.3. Tujuan Penulisan 2

1.4. Metodologi 2

1.5. Pembatasan Masalah 2

1.6. Sistematika Penulisan 2

BAB II TEORI PENUNJANG 4

2.1. Dasar-dasar Algoritma 4

2.2. Pengurutan (Sorting) 5

2.3. Bahasa Pemrograman Pascal 8

2.3.1. Pernyataan Perulangan 9

2.3.1.1. Pernyataan for 10

2.3.1.2. Pernyataan repeat until 11

2.3.1.3. Pernyataan while 12

2.3.2. Pernyataan Penyeleksian Kondisi 13

2.3.2.1. Pernyataan if 13

2.3.2.2. Pernyataan case 14

2.3.3. Prosedur dan fungsi 15

2.3.3.1. Prosedur 16

2.3.3.2. Fungsi 16

2.3.4. Larik (Array) 17

2.4. Kompleksitas Algoritma 18

2.5. Waktu Tempuh (Running Time)..... 18

2.5.1. Notasi Big Oh 19

2.5.2. Analisa Algoritma 24

BAB III ANALISA ALGORITMA POSTMAN'S SORT 29

3.1. Metode Postman's Sort 29

3.2. Algoritma Postman's Sort 34

3.3. Analisa Algoritma Postman's Sort 53

BAB IV KESIMPULAN 64

DAFTAR PUSTAKA 65

LAMPIRAN 66

DAFTAR GAMBAR

Gambar 2.1. Hubungan masalah, algoritma dan solusi 4

Gambar 2.2. Struktur program Pascal 9

Gambar 3.1. Ilustrasi metode Postman's sort (1) 30

Gambar 3.2. Ilustrasi metode Postman's sort (2) 32

Gambar 3.3. Ilustrasi metode Postman's sort (3) 33

Gambar 3.4. Contoh metode Postman's sort (1) 41

Gambar 3.5. Contoh metode Postman's sort (2) 46

Gambar 3.6. Contoh metode Postman's sort (3) 52

Gambar 3.7. Grafik waktu eksekusi rata-rata 63

DAFTAR TABEL

Tabel 2.1. Contoh perhitungan kompleksitas waktu 26

Tabel 2.2. Contoh analisa kompleksitas waktu 27

Tabel 3.1. Perhitungan kompleksitas waktu algoritma Postman's sort 57

Tabel 3.2. Analisa kompleksitas waktu algoritma Postman's sort 61

Tabel 3.3. Waktu eksekusi 62

BAB I

PENDAHULUAN

1.1. Latar Belakang

Dalam kehidupan sehari-hari banyak sekali dijumpai aktivitas-aktivitas yang sangat membutuhkan adanya pengurutan data. Misal, ketika menggunakan kamus, dapat dibayangkan lama waktu yang diperlukan untuk menemukan suatu kata seandainya kata-kata pada kamus tersebut dalam keadaan acak (tidak terurut). Demikian pula pada saat membuka buku telepon untuk menemukan satu nama pelanggan telepon dari semua pelanggan. Dari contoh-contoh di atas terlihat sangat pentingnya proses pengurutan data saat melakukan aktivitas pencarian pada sekumpulan objek dalam jumlah besar.

Pengurutan data (*sorting*) secara umum dapat didefinisikan sebagai suatu proses untuk menyusun kembali himpunan objek (data) menggunakan aturan tertentu. Keuntungan dari data yang sudah dalam keadaan urut antara lain bahwa data lebih mudah untuk diolah kembali, misalnya dicari, dihapus, disisipi maupun digabungkan. Oleh karena itu proses pengurutan data sangat erat hubungannya dengan pengolahan data.

1.2. Permasalahan

Lingkup permasalahan yang akan dibahas dalam Tugas Akhir ini adalah bagaimana proses pengurutan data dengan metode Postman's sort dan analisa terhadap algoritma Postman's sort tersebut.

1.3. Tujuan penulisan

Tujuan dari penulisan Tugas Akhir ini adalah untuk mengetahui efektifitas algoritma Postman's sort.

1.4. Metodologi

Metode yang digunakan pada penulisan Tugas Akhir ini adalah metode referensi atau studi pustaka, dengan pencarian data melalui buku, literatur dan internet. Sedangkan proses visualisasinya menggunakan bahasa pemrograman Pascal.

1.5. Pembatasan masalah

Permasalahan akan dibatasi pada proses pengurutan data secara *ascending* pada bilangan integer positif maksimal 4 digit. Analisa algoritma yang dilakukan adalah analisa kompleksitas waktu dari algoritma Postman's sort menggunakan notasi O (Big oh).

1.6. Sistematika penulisan

BAB I PENDAHULUAN

1.1. Latar Belakang

1.2. Permasalahan

1.3. Tujuan Penulisan

1.4. Metodologi

1.5. Pembatasan Masalah

1.6. Sistemetika Penulisan

BAB II MATERI PENUNJANG

2.1. Dasar-dasar Algoritma

2.2. Pengurutan (Sorting)

2.3. Bahasa Pemrogram Pascal

2.4. Kompleksitas Algoritma

2.5. Waktu Tempuh (Running Time)

BAB III ANALISA ALGORITMA POSTMAN'S SORT

3.1. Metode Postman's Sort

3.2. Algoritma Postman's Sort

3.3. Analisa Algoritma Postman's Sort

BAB IV KESIMPULAN

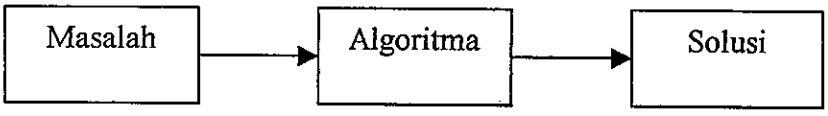
BAB II

TEORI PENUNJANG

2.1. Dasar – dasar Algoritma

Algoritma didefinisikan sebagai suatu metode yang terdiri dari serangkaian langkah terstruktur dan ditulis secara sistematis yang akan dikerjakan dalam menyelesaikan suatu masalah untuk mendapatkan solusi dengan bantuan komputer.

Hubungan antara masalah, algoritma, dan solusi dapat digambarkan sebagai berikut :



Gambar 2.1. Hubungan masalah, algoritma dan solusi

Proses dari masalah hingga terbentuk suatu algoritma disebut tahap pemecahan masalah. Proses dari algoritma hingga terbentuk suatu solusi disebut dengan tahap implementasi.

Algoritma pemrograman yang baik memiliki ciri – ciri sebagai berikut :

1. Memiliki metode yang tepat dalam memecahkan masalah.
2. Menghasilkan keluaran yang benar.
3. Ditulis secara sistemetis, sehingga dapat dipahami dan tidak menimbulkan arti ganda.
4. Operasi yang dibutuhkan terdefinisi dengan jelas.
5. Proses harus berakhir setelah sejumlah langkah dilakukan.

2.2. Pengurutan (Sorting)

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

Keuntungan dari data yang sudah dalam keadaan urut antara lain bahwa data lebih mudah untuk diolah kembali, misalnya dicari, dihapus, disisipi maupun digabungkan.

Data yang akan diurutkan tentunya bervariasi baik dalam hal banyak data maupun jenisnya, tetapi tidak ada algoritma yang terbaik untuk setiap situasi yang dihadapi. Bahkan tidak dapat ditentukan algoritma yang paling baik untuk situasi tertentu.

Ada beberapa faktor yang mempengaruhi efektifitas pengurutan data. Diantaranya adalah banyak dan jenis data, kapasitas pengingat, serta tempat penyimpanan data. Hal tersebut tentunya berpengaruh terhadap waktu yang diperlukan untuk pengurutan data. Oleh karena itu pemilihan algoritma sangat penting untuk menunjang efektifitas pengurutan data.

Metode pengurutan dapat diklasifikasikan menjadi 2 kategori, yaitu :

1. Pengurutan internal.

   Pengurutan internal merupakan metode pengurutan yang dilakukan di dalam memori utama komputer. Dalam pengurutan internal, metode yang digunakan harus ekonomis dalam penggunaan media penyimpan yang tersedia. Yang termasuk dalam pengurutan internal adalah pengurutan

larik, karena larik tersimpan dalam memori utama komputer.. Metode yang digunakan dalam pengurutan internal adalah :

- Sorting by Insertion (penyisipan).

  Prinsip dari metode ini adalah membandingkan data (x_i) dengan data sebelumnya, kemudian menyisipkan data (x_i) ke posisi yang tepat. Proses ini berlanjut sampai semua data terurutkan. Yang termasuk dalam metode ini adalah Straight Insertion Sort, Binary Insertion Sort dan Shell Sort.

- Sorting by Selection (seleksi).

  Prinsip dari metode ini adalah mencari data terkecil dari data pertama sampai terakhir, kemudian data terkecil tersebut ditukar dengan data pertama. Selanjutnya mencari data terkecil dari data kedua sampai data terakhir, kemudian data terkecil tersebut ditukar dengan data kedua. Proses ini berlanjut sampai semua data terurutkan. Yang termasuk dalam metode ini adalah Straight Selection Sort.

- Sorting by Exchange (penukaran).

  Prinsip dari metode ini berdasarkan pada konsep pembandingan dan penukaran sepasang data sampai semua data terurutkan. Yang termasuk dalam metode ini adalah Bubble Sort, Shaker Sort dan Quick Sort.

  Ketiga metode di atas disebut juga dengan metode langsung, karena proses penyisipan, seleksi atau penukaran dilakukan di dalam 1 larik.

- Sorting by Merging (penggabungan).

  Prinsip dari metode ini adalah membandingkan dan menggabungkan data dari dua buah vektor yang masing - masing telah terurutkan. Yang termasuk dalam metode ini adalah Merge Sort.

- Sorting by Distibution (distribusi).

  Prinsip dari metode ini menggunakan 2 buah larik. Larik pertama berisi data-data yang akan diurutkan dan larik kedua merupakan larik yang disediakan untuk pendistribusian data dari larik pertama. Selanjutnya data-data dari larik pertama didistribusikan ke dalam larik kedua sesuai dengan tempat yang telah disediakan. Yang termasuk dalam metode ini adalah Radix Sort dan Postman Sort.

2. Pengurutan eksternal.

   Pengurutan eksternal merupakan metode pengurutan yang datanya tersimpan dalam pangingat luar, misalnya cakram atau pita magnetis. Yang termasuk dalam pengurutan eksternal adalah pengurutan berkas. Prinsip dari pengurutan eksternal ini sama dengan Merge Sort. Data dengan jumlah N rekaman pada berkas dipecah menjadi x bagian (berkas), kemudian data pada masing-masing berkas diurutkan. Selanjutnya setelah x berkas diurutkan secara terepisah, dilakukan merging terhadap x berkas tersebut untuk mendapatkan berkas yang terdiri dari N rekaman yang sudah terurutkan.

2.3. Bahasa Pemrograman Pascal

Menurut jenisnya bahasa pemrograman pada komputer dibagi menjadi 2 kelompok, yaitu :

- Bahasa pemrograman tingkat rendah (*low level programming language*).
- Bahasa pemrograman tingkat tinggi (*high level programming language*).

Bahasa pemrograman tingkat rendah bersifat *machine dependent*, yang artinya bahwa bahasa tingkat rendah untuk suatu komputer akan berbeda dengan komputer lain. Yang termasuk bahasa pemrograman tingkat rendah adalah bahasa mesin.

Bahasa pemrograman tingkat tinggi adalah bahasa pemrograman yang dapat dipahami dan tidak berpengaruh pada komputer yang digunakan. Saat ini banyak dijumpai bahasa pemrograman tingkat tinggi dengan berbagai macam versi dan fasilitas yang disediakan, diantaranya adalah Pascal, BASIC, FORTRAN, C dan COBOL.

Bahasa pemrograman Pascal merupakan bahasa pemrograman dengan teknik pemrograman terstruktur. Pascal diambil dari nama seorang sarjana Perancis, Blaise Pascal. Pertama kali dikembangkan oleh Niklaus Wirth, seorang ahli ilmu komputer dari Swiss, pada tahun 1970.

Secara umum struktur program Pascal adalah sebagai berikut :

```
program BAGAN_PROGRAM;        {*** nama program ***}
uses .....        {*** deklarasi piranti ***}
label .....       {*** deklarasi label ***}
```

```
const .....        {*** deklarasi konstanta ***}
type .....         {*** deklarasi tipe data ***}
var .....          {*** deklarasi peubah ***}
procedure SATU;
procedure DUA;
function TAMBAH;
begin              {*** awal program utama ***}
.....
.....
< pernyataan – pernyataan dari program BAGAN_PROGRAM >
.....
.....
end.               {*** akhir program utama ***}
```

Gambar 2.2. Struktur program Pascal

2.3.1. Pernyataan Perulangan

Pernyataan perulangan (*repetitive pernyataan*) digunakan untuk melakukan proses berulang terhadap pernyataan tunggal maupun pernyataan majemuk. Yang termasuk pernyataan perulangan adalah pernyataan **for**, pernyataan **repeat...until** dan pernyataan **while.**

2.3.1.1. Pernyataan for

Proses berulang dalam pernyataan **for** langsung dikendalikan oleh suatu peubah yang disebut dengan peubah kendali (*control variable*).

Bentuk umum pernyataan **for** :

For *peubah* := *awal* **to** *akhir* **do** *pernyataan*;

dengan peubah : nama peubah kendali.

awal : nilai awal peubah kendali.

akhir : nilai akhir peubah kendali.

pernyataan : pernyataan yang akan diproses berulang.

Bentuk umum diatas dapat dijelaskan sebagai berikut. Pertama kali nilai *peubah* dibuat sama dengan *awal* dan *pernyataan* akan dikerjakan. Selanjutnya nilai *peubah* ditambah dengan 1, dan *pernyataan* dikerjakan lagi. Proses ini diulang sampai nilai *peubah* lebih dari *akhir*. Jika nilai awal lebih besar dari nilai *akhir*, maka *pernyataan* tidak dikerjakan.

Contoh pemakaian pernyataan **for** :

```
for i := 1 to 3 do
    writeln('Test', i);
```

Jika potongan program ini dijalankan, maka akan dihasilkan keluaran :

```
Test 1
Test 2
Test 3
```

Penulisan pernyataan **for** dapat divariasi menjadi :

for *peubah* := *awal* **downto** *akhir* **do** *pernyataan*;

Kata **downto** menunjukkan bahwa setelah *pernyataan* dikerjakan satu kali, maka nilai *peubah* dikurangi dengan 1, sampai akhirnya nilai *peubah* kurang dari nilai *akhir*. Dengan demikian kata **downto** harus digunakan jika nilai *awal* lebih besar dari nilai *akhir*.

Contoh :

```
for i := 3 downto 1 do
    writeln('Test', i);
```

Jika potongan program ini dijalankan, maka akan dihasilkan keluaran :

```
Test 3
Test 2
Test 1
```

2.3.1.2. Pernyataan repeat...until

Pernyataan **repeat** dipakai sebagai awal pernyataan dan **until** dipakai sebagai akhir pernyataan yang dikerjakan berulang sekaligus untuk mencek apakah proses berulang tersebut masih bisa diteruskan atau sudah harus berakhir.

Bentuk umum pernyataan **repeat...until** :

repeat *pernyataan* **until** *kondisi*;

dengan pernyataan : pernyataan yang akan dikerjakan berulang.

kondisi : syarat agar proses berulang dihentikan.

Contoh pemakaian pernyataan **repeat...until** :

```
mulai := 1;
repeat
    writeln('Test');
    mulai := mulai + 1
until mulai > 5;
```

Jika potongan program tersebut dijalankan, akan dihasilkan / dicetak kata **Test** sebanyak 5 kali.

2.3.1.3. Pernyataan while

Pernyataan **while** hampir sama dengan **repeat...until** dengan perbedaan bahwa dalam pernyataan **while** kondisi untuk melakukan proses berulang ditest di awal, sedangkan pada pernyataan **repeat...until** di bagian akhir.

Bentuk umum pernyataan **while** :

while *kondisi* **do** *pernyataan*;

dengan kondisi : syarat agar proses berulang berjalan.

pernyataan : pernyataan yang akan diproses berulang.

Contoh pemakaian pernyataan **while** :

```
mulai := 5;
while mulai > 0 do
    begin
    writeln('Test');
    mulai := mulai - 1;
    end;
```

Jika potongan program tersebut dijalankan, akan dihasilkan / dicetak kata **Test** sebanyak 5 kali.

2.3.2. Pernyataan Penyeleksian kondisi

Pernyataan penyeleksian kondisi digunakan untuk memilih bagian program yang akan dikerjakan sesuai dengan kondisi yang diberikan. Ada 2 buah pernyataan penyeleksian kondisi, yaitu pernyataan **if** dan pernyataan **case**.

2.3.2.1. Pernyataan if

Pernyataan **if** digunakan untuk mencek suatu kondisi dan menentukan apakah kondisi tersebut benar atau salah, kemudian melakukan suatu kegiatan sesuai dengan nilai kondisi tersebut.

Bentuk umum pernyataan **if** :

if *kondisi* **then** *pernyataan1* < **else** *pernyataan2* >;

dengan kondisi : kondisi yang ditest untuk menentukan apakah *pernyataan1* atau *pernyataan2* yang akan dikerjakan.

pernyataan1 : pernyataan yang akan dikerjakan jika *kondisi* bernilai benar.

pernyataan2 : pernyataan yang akan dikerjakan jika *kondisi* bernilai salah.

Contoh pemakaian pernyataan **if** :

```
if x = 100 then
    x := x /5
```

else

x := x –1;

2.3.2.2. Pernyataan case

Pernyataan **case** berisi ungkapan dan sederetan pernyataan yang masing–masing diawali dengan satu atau lebih konstanta atau dengan kata **else**.

Bentuk umum pernyataan **case** :

case *pemilih* **of**

Konstanta1 : pernyataan1;

Konstanta2 : pernyataan2;

.....

.....

< **else** : *pernyataann* ;>

end;

dengan pemilih : nama peubah sebagai pemilih;

konstanta1, konstanta2, ... : kemungkinan–kemungkinan nilai pemilih.

Pernyataan1, pernyataan2, ... : pernyataan yang akan dikerjakan sesuai dengan nilai pemilih.

Contoh pemakaian pernyataan **case** :

case *bilangan* **of**

1..10 : **writeln**('Antara 1 sampai 10');

11..100 : **writeln**('Antara 11 sampai 100');

```
        else     : writeln('Lebih besar dari 100')
    end;
```

2.3.3. Prosedur dan Fungsi

Prosedur dan fungsi merupakan sekelompok pernyataan yang seolah-olah terpisah dari program utama tetapi sesungguhnya merupakan bagian atau sub bagian dari program utama. Prosedur diaktifkan menggunakan pernyataan **procedure** (pemanggil prosedur), dan fungsi diaktifkan dengan suatu ungkapan yang hasilnya akan dikembalikan lagi sebagai nilai baru dari ungkapan tersebut (nama fungsi sekaligus berfungsi sebagai nama peubah).

Contoh prosedur untuk menghitung luas suatu persegi dengan sisi-sisi x dan y :

```
procedure LUAS;
var x, y, L : real;
begin
    L := x * y;
end;
```

Contoh fungsi untuk menghitung luas suatu persegi dengan sisi-sisi x dan y :

```
function LUAS;
var x, y, Luas : real;
begin
    Luas := x * y;
end;
```

2.3.3.1. Prosedur

Prosedur mempunyai struktur yang sama dengan struktur program, yaitu terdiri dari nama prosedur, deklarasi-deklarasi dan bagian utama dari prosedur itu sendiri. Deklarasi dalam prosedur disebut sebagai deklarasi lokal, sehingga hanya bisa digunakan dalam prosedur itu saja dan tidak dikenal di luar prosedur.

Bentuk umum deklarasi prosedur :

procedure *nama* <(*dafpar*)>;

dengan nama : nama prosedur.

dafpar : daftar parameter formal.

Contoh pemakaian deklarasi prosedur :

procedure BACA_MATRIX (Var Mat : Tabel; N : Integer);

2.3.3.2. Fungsi

Secara umum fungsi sama dengan prosedur, dengan perbedaan bahwa nama fungsi sekaligus berfungsi sebagai nama peubah, sehingga dalam deklarasi fungsi harus dinyatakan tipe datanya.

Bentuk umum deklarasi fungsi :

function *nama* <(*dafpar*)> : *tipe*;

dengan nama : nama fungsi.

dafpar : daftar parameter formal.

tipe : tipe data dari fungsi tersebut.

Dalam fungsi semua parameter formal harus berupa parameter nilai, tidak diperbolehkan ada parameter peubah.

Contoh pemakaian deklarasi fungsi untuk menghitung nilai rata-rata dari sekumpulan data yang diketahui :

```
function RATA ( Vektor : Larik; N : Integer);
var I : integer;
    R : real;
begin
        R := 0;
        for I := 1 to N do
                R := R + Vektor[I];
        RATA := R / N
end.
```

2.3.4. Larik (Array)

Larik (**array**) adalah tipe terstruktur yang mempunyai komponen dalam jumlah yang tetap dan setiap komponen mempunyai tipe data yang sama. Posisi masing-masing komponen dalam larik dinyatakan sebagai nomor index.

Bentuk umum dari deklarasi tipe larik :

Type *pengenal* = **array** [*tipe_index*] of *tipe*;

dengan pengenal : nama tipe data.

tipe_index : tipe data untuk nomor index.

tipe : tipe data komponen.

Parameter *tipe_index* menentukan banyaknya komponen dalam larik tersebut. Parameter ini adalah sembarang tipe ordinal kecuali **longint** dan subjangkauan dari **longint.**

Contoh pemakaian deklarasi tipe larik :

type Nilai = **array** [1..100] **of integer**;

Contoh ini menunjukkan bahwa *nilai* adalah tipe data yang berupa larik yang komponennya bertipe integer dan banyaknya 100 buah.

2.4. Kompleksitas algoritma

Salah satu alat yang digunakan untuk mengetahui efisiensi suatu algoritma adalah waktu yang diperlukan oleh komputer untuk menyelesaikan suatu permasalahan dengan menggunakan algoritma tersebut berdasarkan input yang diberikan.

Penyataan tentang hal tersebut di atas termasuk dalam kompleksitas perhitungan suatu algoritma. Kompleksitas waktu suatu algoritma adalah analisa waktu yang diinginkan untuk menyelesaikan suatu permasalahan. Analisa memory yang diinginkan termasuk dalam kompleksitas ruang suatu algoritma. Pertimbangan kompleksitas waktu dan ruang pada suatu algoritma merupakan suatu hal yang penting ketika suatu algoritma akan diimplementasikan.

2.5. Waktu Tempuh (Running Time)

Dengan diketahui kompleksitas waktu suatu algoritma maka efisiensi program berdasarkan waktu dapat dibedakan menurut *Running Time*-nya. Waktu

tempuh (*running time*) dari suatu algoritma adalah ukuran waktu untuk melaksanakan suatu program sehingga menghasilkan output pada suatu kompiler dan mesin eksekusi tertentu.

Dalam analisa algoritma, Running Time dinyatakan dalam simbol f(n). Running Time sebuah program tergantung beberapa faktor :

1. Input program.
2. Kualitas / kemampuan dari kompiler yang digunakan untuk kompilasi pada program.
3. Kemampuan dan kecepatan mesin yang digunakan untuk eksekusi program.
4. Kompleksitas waktu program.

Untuk menghitung waktu tempuh program, dikenal beberapa notasi diantaranya notasi O, notasi Θ, dan notasi Ω.

2.5.1. Notasi Big Oh

Definisi 2.1.

Jika f(n) adalah O(g(n)), maka terdapat dua buah konstanta bulat positif c dan m sedemikian hingga $f(n) \leq c.g(n)$, untuk $n \geq m$.

Penjelasan :

Untuk menunjukkan bahwa f(n) adalah O(g(n)), diperlukan dua buah konstanta c dan m sedemikian hingga $f(n) \leq c.g(n)$, untuk $n \geq m$. Konstanta c dan m yang memenuhi definisi tidak pernah tunggal. Jika konstanta c dan m ada, maka terdapat tidak terbatas konstanta c dan m yang lain, misalnya c_1 dan m_1, sedemikian hingga $c < c_1$ dan $m > m_1$ memenuhi definisi :

$f(n) \leq c.g(n) \leq c_1.g(n)$, untuk $n \geq m \geq m_1$.

Contoh :

Tunjukkan bahwa $f(n) = 5n + 8$ adalah $O(n)$.

Penyelesaian :

Menentukan dua buah konstanta c dan m sedemikian hingga $f(n) \leq c.g(n)$, untuk $n \geq m$. Jika $0 \leq 5n + 8 \leq 5n + 8n = 13n$, untuk $n \geq 1$, maka didapatkan konstanta $c = 13$ dan $m = 1$. Sehingga $f(n) \leq 13\ n$, untuk $n \geq 1$, dan terlihat bahwa f (n) adalah $O(n)$.

Teorema 2.1.

$f(n) = a_d\ n^d + a_{d-1}\ n^{d-1} + \ldots + a_1\ n + a_0$ adalah suatu fungsi polinomial dalam n dan derajat d, maka $f(n) = O(n^d)$.

Bukti :

Dengan merubah semua koefisien dari f(n) menjadi bilangan-bilangan positif maka nilai f(n) akan lebih besar untuk semua n bilangan integer positif. Jika ada $n^j \leq n^d$, untuk $j \leq d$, maka :

$$
\begin{aligned}
f(n) &= a_d\, n^d + a_{d-1}\, n^{d-1} + \ldots + a_1\, n + a_0 \\
&= a_d\, n^d + a_{d-1}\, n^{d-1} + \ldots + a_j\, n^j + \ldots + a_0 \\
&\leq |a_d|\ n^d + \ldots + |a_j|\ n^j + \ldots + |a_0| \\
&\leq |a_d|\ n^d + \ldots + |a_j|\ n^d + \ldots + |a_0|\ n^d \\
&= (|a_d| + \ldots + |a_j| + \ldots + |a_0|)\, n^d \\
&= c.\ n^d
\end{aligned}
$$

dengan $c = (|a_d| + \ldots + |a_j| + \ldots + |a_0|)$, sehingga f(n) adalah $O(n^d)$.

Contoh :

Tunjukkan bahwa $f(n) = 3n^2 - 4n + 5$ adalah $O(n^2)$.

Penyelesaian :

Berdasarkan teorema 2.1, f(n) merupakan fungsi polinomial dalam n dengan derajat 2, maka f(n) adalah $O(n^2)$.

$$\begin{aligned} f(n) &= 3n^2 - 4n + 5 \\ &\leq 3n^2 + |-4|n + 5 \\ &\leq 3n^2 + |-4|n^2 + 5n^2 \\ &= (3+4+5)\, n^2 \\ &= 12\, n^2 \end{aligned}$$

Jadi terlihat bahwa $f(n) = 3n^2 - 4n + 5$ adalah $O(n^2)$.

Teorema 2.2.

Jika f(n) adalah O(g(n)) maka a.f(n) = O(g(n)).

Bukti :

Berdasarkan definisi 2.1, jika f(n) = O(g(n)) maka $f(n) \leq c.g(n)$. Terdapat konstanta c_1, sedemikian hingga $c_1 = \frac{c}{a}$ maka :

$$\begin{aligned} f(n) &\leq c_1 \,.\, g(n) \\ &\leq \frac{c}{a}\, g(n) \end{aligned}$$

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

Contoh :

Tunjukkan bahwa $h(n) = 9n^2 - 12n + 15$ adalah $O(n^2)$.

Penyelesaian :

Dari contoh pada teorema 2.1, $f(n) = 3n^2 - 4n + 5$ adalah $O(n^2)$.

$h(n) = 9n^2 - 12n + 15 = 3(3n^2 - 4n + 5) = 3f(n)$. Maka menurut teorema 2.2, $3f(n)$ adalah $O(n^2)$.

Teorema 2.3.

Jika $f(n)$ adalah $O(g(n))$ dan $h(n)$ adalah $O(k(n))$, maka $f(n) + h(n) = O(g(n) + k(n))$.

Bukti :

$f(n)$ adalah $O(g(n))$ dan $h(n)$ adalah $O(k(n))$. Berdasar definisi 2.1, maka terdapat empat buah konstanta c_1 ,c_2 ,m_1 dan m_2, sehingga

$$f(n) \leq c_1 \,.\, g(n) \;,\; n \geq m_1$$

$$h(n) \leq c_2 \,.\, k(n) \;,\; n \geq m_2$$

$$\begin{aligned} f(n) + h(n) &= c_1 \,.\, g(n) + c_2 \,.\, g(n) \\ &\leq c \,.\, g(n) + c \,.\, g(n) \\ &= c\,(g(n) + k(n)) \\ &= O(g(n) + k(n)) \end{aligned}$$

Notasi $O(g(n) + k(n))$ dapat ditulis menjadi $O(\max(g(n) , k(n)))$, merupakan nilai maksimum dari $g(n)$ dan $k(n)$ yang disebut *maximum rule*.

Contoh :

$f(n) = 3n + 7$ adalah $O(n)$, dan $h(n) = 2n^2 - n + 8$ adalah $O(n^2)$. Tunjukkan bahwa $f(n) + h(n) = O(n^2)$.

Penyelesaian :

$$
\begin{aligned}
f(n) + h(n) &= 3n + 7 + 2n^2 - n + 8 \\
&= 2n^2 + 2n + 15 \\
&\leq 2n^2 + 2n^2 + 15n^2 \\
&= (2 + 2 + 15)n^2 \\
&= O(n^2)
\end{aligned}
$$

Teorema 2.4.

Jika f(n) adalah O(g(n)) dan h(n) adalah O(k(n)), maka $f(n) \cdot h(n) = O(g(n) \cdot k(n))$.

Bukti :

f(n) adalah O(g(n)) dan h(n) adalah O(k(n)). Berdasar definisi 2.1, maka terdapat empat buah konstanta c_1 ,c_2 ,m_1 dan m_2, sehingga

$f(n) \leq c_1 \cdot g(n)$, $n \geq m_1$

$h(n) \leq c_2 \cdot k(n)$, $n \geq m_2$

$$
\begin{aligned}
f(n) \cdot h(n) &= c_1 \cdot g(n) \cdot c_2 \cdot g(n) \\
&\leq c \cdot g(n) \cdot c \cdot g(n) \\
&= c\,(g(n) \cdot k(n)) \\
&= O(g(n) \cdot k(n))
\end{aligned}
$$

Contoh :

$f(n) = 2n^2 - 3$ adalah $O(n^2)$ dan $h(n) = n + 1$ adalah $O(n)$. Tunjukkan bahwa $f(n) \cdot h(n) = O(n^3)$

Penyelesaian :

$$
\begin{aligned}
f(n) \,.\, h(n) &= 2n^3 + 2n^2 - 3n - 3 \\
&\leq 2n^3 + 2n^2 + |-3|\,n + |-3| \\
&\leq 2n^3 + 2n^2 + 3n^3 + 3n^3 \\
&= (2+2+3+3)n^3 \\
&= 10\,n^3 \\
&= O(n^3)
\end{aligned}
$$

Dari definisi dan teorema-teorema yang telah dijelaskan, dapat dibuat suatu rumusan umum, yaitu $n^i = O(n^j)$, untuk $i \leq j$.

2.5.2. Analisa Algoritma

Jika ditentukan beberapa algoritma yang berbeda untuk menyelesaikan suatu 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 suatu algoritma.

Secara umum running time pada suatu pernyataan / kelompok pernyataan terparameterisasi oleh ukuran input / beberapa variabel. Parameter untuk running time suatu program adalah n (ukuran input). Adapun aturan umum untuk menganalisa suatu algoritma adalah :

1. Jika $f(n) = O(g(n))$ dan $h(n) = O(k(n))$, maka
   a. $f(n) + h(n) = O(\max(g(n), k(n))$.
   b. $f(n) \,.\, h(n) = O(g(n) \,.\, k(n))$.

2. Running Time suatu loop adalah running time dari besarnya jumlah iterasi pernyataan / kelompok pernyataan di dalam loop.
3. Total running time di dalam kelompok loop tersarang adalah running time dari pernyataan / kelompok pernyataan yang merupakan hasil kali dari besarnya keseluruhan loop.
4. Running Time setiap *assigment* (tugas), *read* (baca), dan *write* (tulis) adalah O(1).
5. Running Time pada barisan pernyataan ditentukan dengan aturan penjumlahan yang merupakan running time terbesar dari beberapa barisan pernyataan.
6. Running Time pada pernyataan *if* adalah harga pada kondisi pernyataan yang dieksekusi ditambah waktu untuk mengevaluasi kondisi. Waktu menghitung kondisi secara normal adalah O(1). Running time untuk *if..then..else* adalah waktu menghitung kondisi ditambah waktu terbesar yang dibutuhkan eksekusi pernyataan jika kondisi *false* (salah).
7. Waktu eksekusi adalah jumlah semua waktu sekitar loop, waktu eksekusi sekumpulan pernyataan dan waktu mengevaluasi kondisi untuk penghentian.

Metode yang digunakan untuk mencari running time dari suatu program adalah dengan membuat tabel dalam daftar jumlah total dari langkah-langkah yang telah dilaksanakan oleh setiap pernyataan. Notasi *cost* merupakan jumlah langkah dari suatu pernyataan setiap eksekusi. Sedangkan *times* adalah jumlah total waktu setiap pernyataan yang telah dieksekusi. Dengan mengkombinasikan keduanya, akan diperoleh perhitungan total setiap pernyataan. Selanjutnya dengan

penjumlahan total tiap pernyataan akan diperoleh running time dari suatu program.

Contoh :

Prosedur penjumlahan dua buah matrik A dan B ordo m x n, dengan hasil penjumlahan adalah matrik C.

```
procedure JUMLAH(var A, B, C : matrik; m, n : integer);
1. var i, j : integer;
2. begin
3.     for i := 1 to m do
4.         for j := 1 to n do
5.         C[i,j] := A[i,j] + B[i,j];
6. end;
```

Prosedur jumlah tersebut akan digunakan untuk mencari running time dari penjumlahan 2 buah matrik.

Line	Cost	Times	Total Steps
1	0	0	0
2	0	0	0
3	1	m + 1	m + 1
4	1	m (n + 1)	mn + m
5	1	mn	mn
6	0	1	0
Total number of steps			2mn + 2m + 1

Tabel 2.1. Contoh perhitungan kompleksitas waktu

Times pada baris ke 3 adalah m + 1 bukan m, hal ini karena i membutuhkan penambahan ke atas menjadi m + 1 sebelum loop diakhiri. Begitu juga *times* pada baris ke 4 yaitu m (n +1). Selanjutnya akan dicari analisa kompleksitas waktu untuk contoh program diatas.

Line	Cost	Times	Total Steps
1	0	0	O(0)
2	0	0	O(0)
3	1	m + 1	O(m)
4	1	mn + m	O(mn)
5	1	Mn	O(mn)
6	0	1	O(1)
f(m,n) = O(max(g(m,n) = O(mn)			

Tabel 2.2. Contoh analisa kompleksitas waktu

Berdasarkan teorema-teorema sub bab 2.5.1, penulisan tabel diatas dapat dijelaskan sebagai berikut :

- Pada line 1, 2 dan 6 mempunyai *total steps* 0, sehingga masing-masing mempunyai kompleksitas waktu O(0).
- Pada line 3 mempunyai *total steps* m + 1, sehingga

  $f(m) = m + 1$

  $0 \leq m + 1 \leq m + m = 2m$

  $f(m) \leq 2m$, maka $f(m) = O(m)$.

- Pada line 4 mempunyai *total steps* mn + m, sehingga

  $f(m,n) = mn + m$

  $0 \leq mn + m \leq mn + mn = 2mn$

  $f(m,n) \leq 2mn$, maka $f(m,n) = O(mn)$.

- Pada line 5 mempunyai *total steps* mn, sehingga

  $f(m,n) = mn$, maka $f(m,n) = O(mn)$.

Sehingga $f(m,n) = 2mn + 2m + 1$

$0 \leq 2mn + 2m + 1 \leq 2mn + 2mn + mn = 5mn$

$f(m,n) \leq 5mn$, maka $f(m,n) = O(mn)$.

BAB III

ANALISA ALGORITMA POSTMAN'S SORT

3.1. Metode Postman's sort

Dasar dari metode Postman's sort ini adalah aktifitas pegawai pos pada saat menerima surat sampai mendistribusikannya hingga ke tempat tujuan. Saat pegawai pos menerima surat dalam jumlah banyak, maka surat-surat tersebut dipisahkan berdasarkan negara dan dikirimkan ke negara tujuan. Kemudian surat-surat tersebut dipisahkan lagi berdasarkan kota dan dikirimkan ke kota tujuan. Proses ini berlanjut hingga masing-masing surat sampai ke alamat tujuan.

Berdasarkan metode tersebut diatas, dapat dilakukan pengurutan terhadap bilangan. Dengan menggunakan basis bilangan desimal, bilangan-bilangan yang akan diurutkan dipecah menjadi 10 kelompok berdasarkan digit signifikan terbesar (DSB)nya, dengan menganggap bahwa bilangan-bilangan tersebut mempunyai cacah digit yang sama, yaitu dengan menambahkan sejumlah digit 0 pada posisi di sebelah kiri DSB jika diperlukan. Kelompok-kelompok ini diberi nama dengan kelompok "0", "1", "2", dan seterusnya sampai kelompok "9", sehingga bilangan-bilangan yang ada di dalam kelompok 0 merupakan bilangan-bilangan yang DSBnya 0, dan seterusnya sehingga bilangan-bilangan yang ada di dalam kelompok "9" merupakan bilangan-bilangan yang DSBnya 9. Setelah semua bilangan dikelompokkan, maka dapat dilihat bahwa semua bilangan dalam kelompok "0" nilainya selalu lebih kecil dari semua bilangan yang ada di kelompok "1". Semua bilangan dalam kelompok "1" nilainya selalu lebih kecil

dari semua bilangan yang ada di kelompok "2", dan seterusnya. Dengan cara yang sama, masing-masing kelompok dipecah lagi menjadi 10 kelompok sesuai dengan ketentuan di atas sampai digit signifikan terkecil (DSK) nya terurutkan. Jika metode tersebut dilakukan, maka semua bilangan akan terurutkan secara *ascending.*

Untuk lebih memperjelas metode Postman's Sort ini, perhatikan contoh berikut:

Terdapat data di bawah ini yang akan diurutkan secara *ascending* :

350	472	416	161	729	167	475	333

Iterasi 1 :

Data	Kelompok									
	0	1	2	3	4	5	6	7	8	9
350				350						
472					472					
416					416					
161		161								
729								729		
167		167								
475					475					
333				333						
Hasil	161	167	350	333	472	416	475	729		

Gambar 3.1. Ilustrasi metode Postman's sort (1)

Iterasi 2 :

Data	Kelompok									
	0	1	2	3	4	5	6	7	8	9
161							161			
167							167			
Hasil	161	167								

Gambar 3.2. (i)

Data	Kelompok									
	0	1	2	3	4	5	6	7	8	9
350						350				
333				333						
Hasil	161	167	333	350						

Gambar 3.2. (ii)

Data	Kelompok									
	0	1	2	3	4	5	6	7	8	9
472								472		
416		416								
475								475		
Hasil	161	167	333	350	416	472	475			

Gambar 3.2. (iii)

Data	Kelompok									
	0	1	2	3	4	5	6	7	8	9
729			729							
Hasil	161	167	333	350	416	472	475	729		

Gambar 3.2. (iv)

Gambar 3.2. (i),(ii),(iii),(iv). Ilustrasi metode Postman's sort (2)

Iterasi 3 :

Data	Kelompok									
	0	1	2	3	4	5	6	7	8	9
161		161								
167								167		
Hasil	161	167								

Gambar 3.3. (i)

Data	Kelompok									
	0	1	2	3	4	5	6	7	8	9
333				333						
Hasil	161	167	333							

Gambar 3.3. (ii)

Data	Kelompok									
	0	1	2	3	4	5	6	7	8	9
350	350									
Hasil	161	167	333	350						

Gambar 3.3. (iii)

Data	Kelompok									
	0	1	2	3	4	5	6	7	8	9
416		416								
Hasil	161	167	333	350	416					

Gambar 3.3. (iv)

Data	Kelompok									
	0	1	2	3	4	5	6	7	8	9
472			472							
475						475				
Hasil	161	167	333	350	416	472	475			

Gambar 3.3. (v)

Data	Kelompok									
	0	1	2	3	4	5	6	7	8	9
729										729
Hasil	161	167	333	350	416	472	475	729		

Gambar 3.3. (vi)

Gambar 3.3. (i),(ii),(iii),(iv),(v),(vi). Ilustrasi metode Postman's sort (3)

Gambar 3.1. sampai 3.3, menjelaskan proses pengurutan menggunakan metode Postman's sort. Gambar 3.1. menunjukkan proses pemisahan data berdasarkan digit ratusan. Setelah proses ini diselesaikan, akan diperoleh urutan bilangan :

161	167	350	333	472	416	475	729

Gambar 3.2.(i),(ii),(iii),(iv), menunjukkan proses pemisahan data pada kelompoknya berdasarkan digit puluhan. Setelah proses ini diselesaikan, akan diperoleh urutan data :

161	167	333	350	416	472	475	729

Gambar 3.3. (i),(ii),(iii),(iv),(v),(vi) menunjukkan proses pemisahan data pada kelompoknya berdasar digit satuan. Proses ini merupakan akhir dari pengurutan sehingga diperoleh urutan data :

161	167	333	350	416	472	475	729

3.2. Algoritma Postman's sort

Metode postman's sort diimplementasikan menggunakan **array**. Peubah yang dibutuhkan adalah A, B dan C, yang mempunyai struktur **array** :

```
type  larik = array [1..15600] of integer;
      baris = array [0..9] of integer;
var A, C    : larik;
    B       : baris;
    M, N    : integer;
```

A merupakan bilangan yang terletak pada digit satuan, puluhan, ratusan dan seterusnya. C merupakan data yang akan diurutkan ataupun data yang telah diurutkan pada setiap iterasi. Sedangkan B merupakan peubah bantu dalam proses pengurutan.

Setelah peubah A, B dan C ditentukan, diperlukan iterasi sebanyak cacah digit terbesar (M) dari masing-masing data C[1..N]. Dalam setiap iterasi, semua data dipecah menjadi 10 kelompok (B[0..9]) dimulai dari digit terbesarnya (A[1..N]), misal dimulai dari digit ratusan, puluhan dan satuan).

Dengan menggunakan pseudocode, algoritma postman's sort adalah sebagai berikut :

Algoritma Postman's sort :

Diketahui :

M : cacah digit terbesar.

N : jumlah data.

Peubah :

A : bilangan pada digit satuan, puluhan dan seterusnya

U : bilangan yang terletak sebelum A.

C : bilangan yang akan diurutkan maupun hasil pengurutan.

B, D : peubah bantu.

g, i, j, k, p, s, t, x, pangkat : pencacah iterasi

Fungsi :

inc(x) : harga x bertambah 1.

dec(x) : harga x berkurang 1.

x div y : hasil pembagian bulat x oleh y.

x mod 10 : sisa pembagian x oleh 10.

Di dalam procedure Postman's sort terdapat procedur Inisialisasi yang merupakan procedure untuk menentukan nilai dari peubah x dan D[x]. X adalah jumlah kelompok pemecahan dan D[x] adalah banyak data pada tiap kelompok. Nilai dari peubah x dan D[x] ini berfungsi sebagai pencacah iterasi pada procedure Postman'a sort.

Procedure Inisialisasi

```
x := 1
    for i = 1 to n do
        pangkat:=1000
        repeat
            for p = 1 to k-1 do
            pangkat = pangkat div 10
        until pangkat > 0
        U = C[i] div (pangkat*10)
        if U = W then
            inc(D[x])
        else
            inc(x)
            inc(D[x])
            W = U
```

```
15.             end if
16.         next i
```

Procedure Postman's Sort (C,N)

Langkah :

```
        {* awal iterasi *}
1.      for k = 1 to m do
2.          Inisialisasi                          {* procedure Inisialisasi *}
3.          g = 0
            {* iterasi pada tiap kelompok *}
4.          for i = 1 to x do
5.              s = D[i]
6.              for j := (g+1) to (g+s) do
7.                  inc(g)
8.                  A[j] = (C[j] div pangkat) mod 10      {* digit pada posisi
                                                  satuan, puluhan, dan seterusnya *}
9.                  V[g] = C[j]
10.                 B[A[j]] = B[A[j]] + 1          {* jumlah A[j] pada tiap
                                                          kelompok *}
11.             next j
                {* pesan tempat *}
12.             for  t = 0 to 8 do
13.                 B[t+1] = B[t] + B[t+1]
                {* menempatkan data pada posisinya *}
```

```
            for t = g downto (g-s+1) do
                C[B[A[t]] + g-s] = V[t]
                B[A[t]]) = B[A[t]] - 1
            next t
            for t = 0 to 9 do
                B[t] = 0
        next i
        for t = 1 to x do
            D[t] = 0
    next k
    end.
```

Untuk memperjelas algoritma Postman's sort diatas, dapat dilihat pada contoh berikut :

Terdapat data di bawah ini yang akan diurutkan secara *ascending* :

350 472 416 161 729 167 475 333

Data tersebut merupakan C[1..8], sehingga :

C[1] : 350

C[2] : 472

C[3] : 416

C[4] : 161

C[5] : 729

C[6] : 167

C[7] : 475

C[8] : 333

Berdasarkan langkah-langkah pada algoritma Postman's sort, maka proses pengurutan dari data tersebut memerlukan 3 iterasi (sesuai dengan cacah digit terbesarnya). Dengan mengerjakan operasi-operasi yang bekerja pada peubah A[1..N], B[0..9] dan C[1..N], maka dapat digambarkan langkah-langkah pada setiap iterasi seperti berikut :

3	5	0
4	7	2
4	1	6
1	6	1
7	2	9
1	6	7
4	7	5
3	3	3

Iterasi 1:

3	5	0
4	7	2
4	1	6
1	6	1
7	2	9
1	6	7
4	7	5
3	3	3

A :

3	4	4	1	7	1	4	3

B :

0	0	0	0	0	0	0	0	0	0

	5	0
	7	2
	1	6
	6	1
	2	9
	6	7
	7	5
	3	3

A:

3	4	4	1	7	1	4	3

B:

0	2	0	2	3	0	0	1	0	0

	5	0
	7	2
	1	6
	6	1
	2	9
	6	7
	7	5
	3	3

A:

3	4	4	1	7	1	4	3

B:

0	2	2	4	7	7	7	8	8	8

	5	0
	7	2
	1	6
	6	1
	2	9
	6	7
	7	5
	3	3

A:

B:

0	0	2	2	7	7	7	5	8	8

C:

1	1	3	3	4	4	4	7

1	6	1
1	6	7
3	5	0
3	3	3
4	7	2
4	1	6
4	7	5
7	2	9

Gambar 3.4. Contoh metode Postman's sort iterasi 1

Iterasi 2

1	6	1
1	6	7
3	5	0
3	3	3
4	7	2
4	1	6
4	7	5
7	2	9

Iterasi 2.1 :

1		1
1		7
3	5	0
3	3	3
4	7	2
4	1	6
4	7	5
7	2	9

A :

6	6

B :

0	0	0	0	0	0	2	0	0	0

1		1
1		7
3	5	0
3	3	3
4	7	2
4	1	6
4	7	5
7	2	9

A :

6	6

B :

0	0	0	0	0	0	2	2	2	2

1		1
1		7
3	5	0
3	3	3
4	7	2
4	1	6
4	7	5
7	2	9

A :

B :

0	0.	0	0	0	0	0	2	2	2

C :

6	6						

Iterasi 2.2 :

1		1
1		7
3		0
3		3
4	7	2
4	1	6
4	7	5
7	2	9

A :

5	3

B :

0	0	0	1	0	1	0	0	0	0

C :

6	6						

1		1
1		7
3		0
3		3
4	7	2
4	1	6
4	7	5
7	2	9

A :

5	3

B :

0	0	0	1	1	2	2	2	2	2

C :

6	6						

1		1
1		7
3		0
3		3
4	7	2
4	1	6
4	7	5
7	2	9

A :

B :

0	0	0	0	1	1	2	2	2	2

C :

6	6	3	5				

Iterasi 2.3 :

1		1
1		7
3		0
3		3
4		2
4		6
4		5
7	2	9

A :

7	1	7

B :

0	1	0	0	0	0	0	2	0	0

C :

6	6	3	5				

1		1
1		7
3		0
3		3
4		2
4		6
4		5
7	2	9

A :

7	1	7

B :

0	1	1	1	1	1	1	3	3	3

C :

6	6	3	5				

1		1
1		7
3		0
3		3
4		2
4		6
4		5
7	2	9

A :

B :

0	0	1	1	1	1	1	1	3	3

C :

6	6	3	5	1	7	7	

Iterasi 2.4 :

1		1
1		7
3		0
3		3
4		2
4		6
4		5
7		9

A :

2	

B :

0	0	1	0	0	0	0	0	0	0

C :

6	6	3	5	1	7	7	

1		1
1		7
3		0
3		3
4		2
4		6
4		5
7		9

A :

2	

B :

0	0	1	1	1	1	1	1	1	1

C :

6	6	3	5	1	7	7	

1		1
1		7
3		0
3		3
4		2
4		6
4		5
7		9

A :

B :

0	0	0	1	1	1	1	1	1	1

C :

6	6	3	5	1	7	7	2

1	6	1
1	6	7
3	3	3
3	5	0
4	1	6
4	7	2
4	7	5
7	2	9

Gambar 3.5. Contoh metode Postman's sort iterasi 2

Iterasi 3 :

1	6	1
1	6	7
3	3	3
3	5	0
4	1	6
4	7	2
4	7	5
7	2	9

Iterasi 3.1 :

1	6	
1	6	
3	3	3
3	5	0
4	1	6
4	7	2
4	7	5
7	2	9

A :

1	7

B :

0	1	0	0	0	0	0	1	0	0

1	6	
1	6	
3	3	3
3	5	0
4	1	6
4	7	2
4	7	5
7	2	9

A :

1	7

B :

0	1	1	1	1	1	1	2	2	2

1	6	
1	6	
3	3	3
3	5	0
4	1	6
4	7	2
4	7	5
7	2	9

A :

B :

0	0	1	1	1	1	1	1	2	2

C :

1	7						

Iterasi 3.2 :

1	6	
1	6	
3	3	
3	5	0
4	1	6
4	7	2
4	7	5
7	2	9

A :

3

B :

0	0	0	1	0	0	0	0	0	0

C :

1	7						

1	6	
1	6	
3	3	
3	5	0
4	1	6
4	7	2
4	7	5
7	2	9

A :

3

B :

0	0	0	1	1	1	1	1	1	1

C :

1	7						

1	6	
1	6	
3	3	
3	5	0
4	1	6
4	7	2
4	7	5
7	2	9

A :

B :

0	0	0	0	1	1	1	1	1	1

C :

1	7	3					

Iterasi 3.3 :

1	6	
1	6	
3	3	
3	5	
4	1	6
4	7	2
4	7	5
7	2	9

A :

0	

B :

1	0	0	0	0	0	0	0	0	0

C :

1	7	3					

1	6	
1	6	
3	3	
3	5	
4	1	6
4	7	2
4	7	5
7	2	9

A :

0	

B :

1	1	1	1	1	1	1	1	1	1

C :

1	7	3					

1	6	
1	6	
3	3	
3	5	
4	1	6
4	7	2
4	7	5
7	2	9

A :

B :

0	1	1	1	1	1	1	1	1	1

C :

1	7	3	0				

Iterasi 3.4 :

1	6	
1	6	
3	3	
3	5	
4	1	
4	7	2
4	7	5
7	2	9

A :

	6

B :

0	0	0	0	0	0	1	0	0	0

C :

1	7	3	0				

1	6	
1	6	
3	3	
3	5	
4	1	
4	7	2
4	7	5
7	2	9

A :

	6

B :

0	0	0	0	0	0	1	1	1	1

C :

1	7	3	0				

1	6	
1	6	
3	3	
3	5	
4	1	
4	7	2
4	7	5
7	2	9

A :

B :

0	0	0	0	0	0	0	1	1	1

C :

1	7	3	0	6			

Iterasi 3.5 :

1	6	
1	6	
3	3	
3	5	
4	1	
4	7	
4	7	
7	2	9

A :

2	5

B :

0	0	1	0	0	1	0	0	0	0

C :

1	7	3	0	6			

1	6	
1	6	
3	3	
3	5	
4	1	
4	7	
4	7	
7	2	9

A :

2	5

B :

0	0	1	1	1	2	2	2	2	2

C :

1	7	3	0	6			

1	6	
1	6	
3	3	
3	5	
4	1	
4	7	
4	7	
7	2	9

A :

2	5

B :

0	0	0	1	1	1	2	2	2	2

C :

1	7	3	0	6	2	5	

Iterasi 3.6 :

1	6	
1	6	
3	3	
3	5	
4	1	
4	7	
4	7	
7	2	

A :

	9

B :

0	0	0	0	0	0	0	0	0	1

C :

1	7	3	0	6	2	5	

1	6	
1	6	
3	3	
3	5	
4	1	
4	7	
4	7	
7	2	

A :

	9

B :

0	0	0	0	0	0	0	0	0	1

C :

1	7	3	0	6	2	5	

1	6	
1	6	
3	3	
3	5	
4	1	
4	7	
4	7	
7	2	

A :

B :

0	0	0	0	0	0	0	0	0	0

C :

1	7	3	0	6	2	5	9

Gambar 3.6. Contoh metode Postman's sort iterasi 3

1	6	1
1	6	7
3	3	3
3	5	0
4	1	6
4	7	2
4	7	5
7	2	9

Setelah setiap langkah dilakukan sebanyak 3 iterasi, maka didapatkan data yang telah urut secara *ascending* :

161	167	333	350	416	472	475	729

3.2. Analisa Algoritma Postman's Sort

Secara garis besar, algoritma Postman's sort adalah sebagai berikut :

```
Procedure Postmans_Sort ( C[1..N] );
    0.  Baca data random yang akan diurutkan;
    1.  for k = 1 to m do
    2.  begin
    3.      Inisialisasi;
    4.      g = 0;
    5.      for i = 1 to x do
    6.      begin
    7.          s = D[i];
```

```
8.          for j := (g+1) to (g+s) do
9.          begin
10.             inc(g);
11.             A[j] = (C[j] div pangkat) mod ;
12.             V[g] = C[j];
13.             B[A[j]] = B[A[j]] + 1;
14.         end;
15.         for  t = 0 to 8 do
16.             B[t+1] = B[t] + B[t+1];
17.         for t = g downto (g-s+1) do
18.         begin
19.             C[B[A[t]] + g-s] = V[t];
20.             B[A[t]]) = B[A[t]] – 1;
21.         end;
22.         for t = 0 to 9 do
23.             B[t] = 0;
24.     end;
25. end;
```

Procedure Postman's sort menggambarkan cara algoritma tesebut bekerja untuk pengurutan data C[1..N]. Dengan menggunakan procedure tersebut maka algoritma Postman's sort dapat dijelaskan sebagai berikut :

- Baris 0 merupakan proses input data C[1..N] dari program utama.

- Baris 1 adalah awal dari iterasi 1 sampai m. Baris ini merupakan pernyataan perulangan for yang prosesnya mencakup baris 2 sampai 25. Running time dari pernyataan perulangan ini adalah harga dari jumlah iterasi tiap blok pernyataan yang ada di dalamnya.
- Baris 2, 6, 9, 18, begin merupakan awal dari pernyataan perulangan for.
- Baris 3 merupakan procedur Inisialisasi.
- Baris 4 merupakan pernyataan pemberian.
- Baris 5 merupakan pernyataan perulangan for yang prosesnya mencakup baris 6 sampai 24. Running time dari pernyataan perulangan ini adalah harga dari jumlah iterasi tiap blok pernyataan yang ada di dalamnya.
- Baris 7 merupakan pernyataan pemberian.
- Baris 8 merupakan pernyataan perulangan for yang prosesnya mencakup baris 9 sampai 14. Running time dari pernyataan perulangan ini adalah harga dari jumlah iterasi tiap blok pernyataan yang ada di dalamnya.
- Baris 10 sampai 13 merupakan pernyataan pemberian.
- Baris 14, 21, 24, 25, end merupakan akhir dari pernyataan perulangan for.
- Baris 15 merupakan pernyataan perulangan for yang prosesnya mencakup baris 16. Running time dari pernyataan perulangan ini adalah harga dari jumlah iterasi tiap blok pernyataan yang ada di dalamnya.
- Baris 17 merupakan pernyataan perulangan for yang prosesnya mencakup baris 18 sampai 21. Running time dari pernyataan perulangan

ini adalah harga dari jumlah iterasi tiap blok pernyataan yang ada di dalamnya.

- Baris 19 dan 20 merupakan pernyataan pemberian.
- Baris 22 merupakan pernyataan perulangan for yang prosesnya mencakup baris 23. Running time dari pernyataan perulangan ini adalah harga dari jumlah iterasi pernyataan yang ada di dalamnya.

Dari penjelasan di atas dapat dihitung Running time dari algoritma Postman's sort sebagai berikut :

Line	Cost	Times	Total Steps
0	0	-	-
1	c_1	m + 1	c_1 (m + 1)
2	0	1	0
3	c_2	m (n + 1)	c_2 m (n + 1)
4	1	m	m
5	c_3	m (x + 1)	c_3 m (x + 1)
6	0	1	0
7	1	m x	m x
8	c_4	m x (s + 1)	c_4 m x (s + 1)
9	0	1	0
10 - 13	4	m x s	4 m x s
14	0	1	0
15	1	9 m x s	9 m x s
16	1	8 m x s	8 m x s
17	c_5	m x (s + 1)	c_5 m x (s + 1)
18	0	1	0
19-20	2	m x s	2 m x s

21	0	1	0
22	1	10 m x s	10 m x s
23	0	1	0
24	0	1	0

Tabel 3.1 Perhitungan kompleksitas waktu algoritma Postman's sort

Berdasarkan teorema-teorema pada bab II, maka *total steps* pada tabel diatas dapat dinyatakan sebagai fungsi f(m,n), sehingga dapat ditentukan running timenya :

- Baris 2, 6, 9, 14, 21, 18, 24, 25 :

  $f(m,n) = 0$

  $f(m,n) = O(0)$.

- Baris 1 :

  $f(m) = c_1 (m + 1)$

  $= c_1 m + c_1$

  $0 \leq c_1 m + c_1 \leq c_1 m + c_1 m = 2 c_1 m$

  $f(m,n) \leq 2 c_1 m = O(m)$.

- Baris 3 :

  $f(m,n) = c_2 m (n + 1)$

  $= c_2 m n + c_2 m$

  $0 \leq c_2 m n + c_2 m \leq c_2 m n + c_2 m n = 2 c_2 m n$

  $f(m,n) \leq 2 c_2 m n = O(mn)$.

- Baris 4 :

  f (m) = m

  f (m) = O(m).

Pada sub bab 3.2 diketahui bahwa x adalah jumlah kelompok pemecahan dan D[x] atau s adalah banyak data pada tiap kelompok. Misal untuk jumlah data N = 10, pada iterasi k = 1 akan dihasilkan nilai x = 1 dan D[1] = 10. Untuk iterasi k = 2 dan seterusnya akan dihasilkan nilai :

$1 \leq x \leq 10$ dan D[1] + D[2] + ... + D[x] = 10.

Maka untuk :

```
for i := 1 to x do
    for j := 1 to D[I] do
        begin
        .....
        {* pernyataan – pernyataan *}
        .....
        end;
```

jumlah iterasi pernyataan – pernyataan di dalam loop adalah

D[1].1 + D[2].1 + ... + D[x].1 = N kali.

Sehingga untuk baris 5 sampai 22, pernyataan – pernyataan yang ada di dalam loop for i := 1 to x akan melakukan iterasi sebanyak N kali.

- Baris 5 :

$f(m) = c_3\, m\,(x+1)$

$= c_3\,(x+1)\,m$

$= c\,m$

$f(m) \leq c\,m = O(m).$

- Baris 7 :

$f(m) = x\,m$

$= c\,m$

$f(m) \leq c\,m = O(m).$

- Baris 8 :

$f(m,n) = c_4\, m\, x\,(s+1)$

$= c_4\, m\, x\ s + c_4\, m\, x$

$= c\,m\,n + c\,m$

$0 \leq c\,m\,n + c\,m \leq c\,m\,n + c\,m\,n = 2\,c\,m\,n$

$f(m,n) \leq 2\,c\,m\,n = O(mn).$

- Baris 10 - 13 :

$f(m,n) = 4\,m\,x\,s$

$= 4\,m\,n$

$f(m,n) \leq c\,m\,n = O(mn).$

- Baris 15 :

$f(m,n) = 9\,m\,x\,s$

$= 9\,m\,n$

$f(m,n) \leq c\,m\,n = O(mn).$

- Baris 16 :

  $f(m,n) = 8\,m\,x\,s$

  $= c\,m\,n$

  $f(m,n) \leq c\,m\,n = O(mn)$.

- Baris 17 :

  $f(m,n) = c_5\,m\,x\,(s+1)$

  $= c_5\,m\,x\,s + c_5\,m\,x$

  $= c\,m\,n + c\,m$

  $0 \leq c\,m\,n + c\,m \leq c\,m\,n + c\,m\,n = 2\,c\,m\,n$

  $f(m,n) \leq 2\,c\,m\,n = O(mn)$.

- Baris 19 - 20 :

  $f(m,n) = 2\,m\,x\,s$

  $= 2\,m\,n$

  $f(m,n) \leq c\,m\,n = O(mn)$.

- Baris 22 :

  $f(m,n) = 10\,m\,x\,s$

  $= 10\,m\,n$

  $f(m,n) \leq c\,m\,n = O(mn)$.

Jadi total running time dari prosedur Postman's sort adalah :

$f(m,n) = 4m + 8mn$

$0 \leq 4m + 8mn \leq 4mn + 8mn = 12\,mn$

$f(m,n) \leq 12\,mn = O(mn)$

Line	Cost	Times	Total Steps
0	0	-	-
1	c_1	m + 1	O(m)
2	0	1	O(0)
3	c_2	m (n + 1)	O(mn)
4	1	m	O(m)
5	c_3	m (x + 1)	O(m)
6	0	1	O(0)
7	1	m x	O(m)
8	c_4	m x (s + 1)	O(mn)
9	0	1	O(0)
10 - 13	4	m x s	O(mn)
14	0	1	O(0)
15	1	9 m x s	O(mn)
16	1	8 m x s	O(mn)
17	c_5	m x (s + 1)	O(mn)
18	0	1	O(0)
19-20	2	m x s	O(mn)
21	0	1	O(0)
22	1	10 m x s	O(mn)
23	0	1	O(0)
24	0	1	O(0)
F(m,n) = O(max(f(m,n))) = O(mn)			

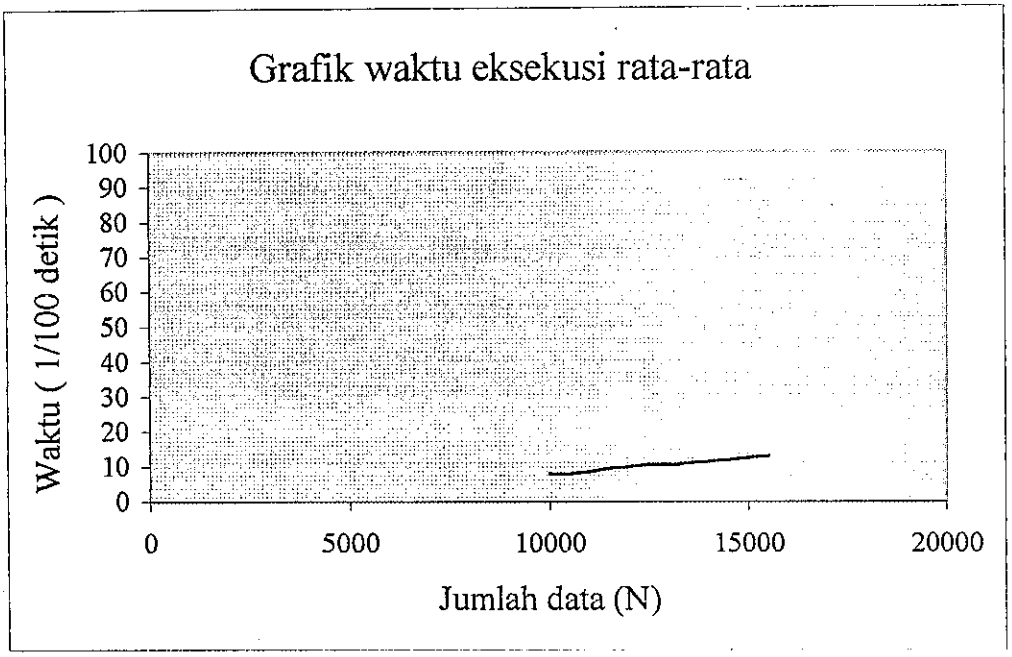
Tabel 3.2 Analisa kompleksitas waktu algoritma Postman's sort

Waktu yang diperlukan oleh metode Postman's sort ini tergantung pada m dan n. Loop terluar for k := 1 to m bergerak sebesar m waktu (sekali untuk masing-masing digit) dan loop di dalamnya bergerak sebesar n waktu (sekali untuk masing-masing elemen).

Dengan melakukan eksekusi program menggunakan algoritma Postman's Sort, untuk setiap jumlah data yang ditentukan, diperlukan waktu tempuh (*running time*) rata-rata sebagai berikut :

Tabel 3.3. Waktu eksekusi

JUMLAH DATA	WAKTU EKSEKUSI (1/100 detik)										WAKTU RATA-RATA (1/100 detik)
	1	2	3	4	5	6	7	8	9	10	
10000	5	10	10	5	10	10	5	10	10	5	8.00
10500	10	10	5	10	5	10	5	5	10	10	8.00
11000	10	5	10	10	10	10	5	5	10	10	8.50
11500	10	10	10	10	10	5	10	10	10	10	9.50
12000	10	10	10	10	10	10	10	10	10	10	10.00
12500	10	10	10	10	10	10	10	10	10	15	10.50
13000	10	10	10	10	10	15	10	10	10	10	10.50
13500	10	15	10	10	10	10	10	15	10	10	11.00
14000	15	10	10	15	10	10	10	10	10	15	11.50
14500	15	15	10	15	10	10	15	10	10	10	12.00
15000	15	10	15	15	10	10	10	10	15	15	12.50
15500	10	15	15	10	15	15	10	15	10	15	13.00



Gambar 3.7. Grafik waktu eksekusi rata-rata

BAB IV

KESIMPULAN

Dari pembahasan pada Tugas Akhir ini dapat diambil kesimpulan sebagai berikut :

1. Waktu tempuh (running time) dari algoritma Postman sort tergantung pada jumlah iterasi m dan banyaknya data n, dengan perolehan kompleksitas waktu dalam notasi O = O(mn).
2. Dari hasil uji coba, untuk jumlah data N > 10000 terlihat bahwa grafik yang terbentuk mendekati linier.

DAFTAR PUSTAKA

Albertson, Michael O : *Discrete Mathematics With Algorithms*, Smith College, New York, 1988

Baase, Sara : *Computer Algorithms*, Massachusetts, Addison – wesley, 1989

Knuth, Donald E : *The art of computer programing, vol 3 : Sorting and Searching*, Menlo Park, Cal., Addison – Wesley, 1973

Ramey, Robert : *Software Development ,The Postman's Sort*, www.rrsd.com

Ramey, Robert : *The Postman's Sort*, www.iitg.ernet.in

Santosa, P. I. : *Struktur Data Menggunakan Turbo Pascal 6.0*, Andi Offset Yogyakarta, 1992

Wirth, Niklaus : *Algorithms + Data Structures = Programs*, Englewood Cliffs, N. J., Prentice – Hall, 1976