BAB II

DASAR TEORI

2.1 Pengurutan dengan Metode Quicksort

2.1.1 Pengurutan (Sorting)

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

2.1.1.1 Pengurutan Data Secara Urut Naik

Pengurutan data secara urut naik (ascending) adalah pengurutan data dari data yang nilainya paling kecil sampai dengan data yang nilainya paling besar.

Contoh 2.1 :

45 34 67 23 89 12 56

Akan diurutkan data secara urut naik, hasilnya :

12 23 34 45 56 67 89

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

2.1.1.2 Pengurutan Data Secara Urut Turun

Yang dimaksud pengurutan data secara urut turun (descending) adalah pengurutan data dari data yang nilainya paling besar sampai dengan data yang nilainya paling kecil.

Contoh 2.2 :

29 78 57 25 8 48

Akan diurutkan data secara urut turun, hasilnya :

78 57 48 29 25 8

Data yang diurutkan sangat bervariasi, baik dalam hal jumlah data maupun jenis data yang akan diurutkan. Sayangnya, tidak ada satu pun algoritma terbaik untuk untuk setiap situasi yang dihadapi. Bahkan terkadang cukup sulit untuk menentukan algoritma mana yang paling baik untuk situasi tertentu. Dikarenakan adanya beberapa faktor yang mempengaruhi efektifitas algoritma pengurutan. Beberapa faktor yang mempengaruhi efektifitas suatu algoritma pengurutan antara lain : jumlah data yang akan diurutkan, kapasitas pengingat apakah mampu menyimpan semua data , dan tempat penyimpanan data. Contoh tempat penyimpanan data adalah piringan (disk), disket, dan hard disk.

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

Dengan alasan-alasan di atas sejumlah metode pengurutan bisa diklasifikasikan menjadi dua kategori, yaitu pengurutan larik (array) dan pengurutan berkas (sequential access file).

2.1.2 Pengurutan Larik

Pengurutan larik disebut juga sebagai internal sorting. Dikatakan internal karena larik tersebut tersimpan dalam memori utama komputer yang mempunyai kecepatan tinggi. Contoh sederhana dalam internal sorting adalah pengurutan sejumlah kartu yang telah diberi nomor. Penyusunan kartu sebagai sebuah larik dapat dibayangkan bahwa semua kartu dapat terlihat semua sehingga semua kartu dapat dilihat dengan jelas nomornya. Dalam internal sorting sendiri terdapat tiga macam pengurutan yaitu :

1. Sorting by insertion (pengurutan dengan penyisipan)
2. Sorting by selection (pengurutan dengan penyeleksian)
3. Sorting by exchange (pengurutan dengan penukaran)

2.1.3 Pengurutan Berkas

Pengurutan berkas disebut juga sebagai eksternal sorting. Dikatakan berkas karena biasanya tersimpan dalam pengingat luar (tambahan), misalnya piringan atau pita magnetis.

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

Penyusunan kartu yang agak berbeda dengan internal sorting di atas adalah sebuah contoh eksternal sorting. Yaitu semua kartu ditumpuk sehingga hanya kartu bagian atas saja yang dapat dilihat nomornya. Batasan ini akan membawa konsekuensi mendalam dalam pemilihan metode yang akan digunakan, tetapi mungkin tidak bisa dihindarkan karena jumlah kartu cukup banyak sehingga meja tidak akan cukup luas untuk meletakkan seluruh kartu diatasnya.

Pengurutan yang termasuk dalam eksternal sorting antara lain mergesort dan polyphasesort.

2.1.4 Metode Quicksort

Metode Quicksort termasuk dalam pengurutan dengan cara penukaran. Metode ini diperkenalkan oleh C.A.R Hoare. Unjuk kerja metode ini sangat cepat (lebih cepat dari metode-metode pengurutan yang lain) sehingga oleh penemunya disebut Quicksort.

Tabel 2.1 menunjukkan waktu yang diperlukan oleh berbagai metode pengurutan menggunakan sistem Pascal pada komputer CDC 6400. Tiga kolom yang terlihat menunjukkan waktu yang diperlukan untuk mengurutkan larik yang sudah terurutkan, larik dengan urutan acak, dan larik dalam keadaan urutan terbalik.

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

Metode	Terurutkan		Acak		Urut terbalik	
	N = 256	N = 512	N= 256	N= 512	N=256	N=512
Penyisipan langsung	12	23	366	1444	704	2836
Penyisipan biner	56	125	373	1327	662	2490
Pemilihan langsung	489	1907	509	1956	695	2675
Bublesort	540	2165	1026	4054	1492	5931
Shakesort	5	9	961	3642	1619	6520
Shellsort	58	116	127	349	157	492
Heapsort	116	253	110	241	104	226
Mergesort	99	234	102	242	99	232
Quicksort	31	69	60	146	37	79

Tabel 2.1 *Waktu eksekusi berbagai metode pengurutan (dalam mili detik)*

Quicksort didasarkan pada fakta bahwa penukaran sebaiknya dilakukan atas dua data yang berjarak cukup jauh. Dimisalkan terdapat N elemen dalam keadaan urut turun. Adalah mungkin untuk mengurutkan N elemen tersebut dengan N/2 kali, yakni dengan pertama kali menukarkan elemen paling kiri dengan paling kanan, kemudian secara bertahap menuju ke elemen yang ada di tengah. Hal ini bisa dilakukan jika sudah jelas bahwa urutannya adalah urut turun atau urut naik.

Secara garis besar metode ini dijelaskan sebagai berikut. Misalnya akan diurutkan larik A yang mempunyai N elemen. Dipilih sembarang elemen dari larik tersebut, misalnya X. Kemudian semua elemen tersebut disusun dengan

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

menempatkan X pada posisi J sedemikian rupa sehingga elemen pertama sampai ke J-1 mempunyai nilai lebih kecil dari X dan elemen ke J+1 sampai ke N mempunyai nilai lebih besar dari X. Dengan demikian sekarang terdapat tiga buah bagian, bagian pertama terdiri dari elemen yang nilainya lebih kecil dari X, bagian kedua terdiri elemennya X, dan bagian ketiga adalah elemen yang nilainya lebih besar dari X.

Pada langkah berikutnya, proses di atas diulang pada bagian pertama dan bagian ketiga sehingga larik tersebut menjadi tujuh subbagian. Proses di atas diulang kembali terus-menerus sehingga seluruh subbagian mempunyai satu elemen. Untuk lebih jelasnya, perhatikan contoh berikut (diambil elemen pertama dan ditempatkan pada posisinya). Misalkan terdapat larik yang elemennya adalah

29	60	15	36	78	40	11	29	19

Gambar 2.1 Larik untuk ilustrasi Metode Quicksort.

15 11 29 19	29	60 36 78 40
bagian pertama	bagian kedua	bagian ketiga

Gambar 2.2. Keadaan larik setelah satu kali proses.

Diambil 29 sebagai pivot. Pivot ini akan diletakkan pada suatu posisi sehingga akan membagi larik tersebut menjadi tiga bagian, dimana bagian pertama terdiri dari elemen yang nilainya selalu lebih kecil atau sama dengan

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

29, bagian kedua terdiri dari pivot tersebut yaitu 29, dan bagian ketiga terdiri dari elemen yang nilainya lebih besar dari 29.

Kemudian pada bagian pertama dan bagian ketiga diproses seperti di atas sampai seluruh subbagian mempunyai satu elemen dimana kondisi ini merupakan kondisi yang sudah terurutkan. Gambar 2.3 menyajikan proses secara lengkap metode Quicksort untuk mengurutkan larik mulai dari gambar 2.1.

29 60 15 36 78 40 11 29 11

15 11 29 19 29 60 36 78 40

11 15 29 19 29 36 40 60 78

11 15 19 29 29 36 40 60 78

11 15 19 29 29 36 40 60 78

Gambar 2.3. Ilustrasi metode Quicksort

Gambar 2.3 di atas menunjukkan proses pembagian dari larik menjadi bagian-bagian, bagian menjadi subbagian, dan seterusnya. Gambar 2.4 memperlihatkan isi sub-subbagian setelah proses pengurutan selesai dilaksanakan.

11 15 19 29 29 36 40 60 78

Gambar 2.4. Hasil akhir dari metode Quicksort.

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

Dari ilustrasi di atas dapat dilihat bahwa metode quicksort dapat diimplementasikan menggunakan dua cara. Cara pertama adalah dengan cara non rekursif dan cara kedua rekursif. Pada kedua cara di atas, persoalan utama yang perlu diperhatikan adalah bagaimana meletakkan suatu pivot pada posisinya yang tepat sehingga memenuhi ketentuan di atas dan bagaimana menyimpan batas-batas bagian pada saat larik dibagi menjadi tiga bagian, kemudian menjadi tujuh subbagian, dan seterusnya. Yang akan dijelaskan disini adalah penempatan pivot pada posisinya yang tepat.

Contoh 2.3:

37	58	6	18	50	28	121	17	120
I=1								J=9

Pertama kali ditentukan I=1 (untuk bergerak dari kiri ke kanan) dan J=N untuk bergerak dari kanan ke kiri. Kedua perubah ini diperlukan untuk mencatat posisi terkahir dimana proses harus berhenti. Dalam hal ini proses akan dihentikan jika nilai I lebih besar atau sama dengan J. Sebagai contoh, akan ditempatkan elemen pertama , 37, pada posisinya yang tepat dengan bergerak dari dua arah dari kiri ke kanan, dan dari kanan ke kiri secara bergantian. Kemudian dilakukan pembandingan elemen antara 37 dengan elemen yang lebih kecil dari 37, yang didapat dari elemen terakhir yang bergerak dari kanan ke kiri (J-1). Dan dari proses tersebut diperoleh elemen 17. Selanjutnya kedua elemen ini ditukarkan. Sehingga diperoleh isi larik yang baru yaitu :

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

17	58	6	18	50	28	121	37	120
I=1							J=8	

Dimulai dari 17, selanjutnya dilakukan pembandingan antara 37 dengan elemen yang nilainya lebih besar dari 37, yang didapat dari elemen pertama yang bergerak dari kiri ke kanan (I+1). Dan dari proses tersebut diperoleh elemen 58. Kemudian kedua elemen ini ditukarkan, sehingga diperoleh isi larik yang baru, yaitu :

17	37	6	18	50	28	121	58	120
	I=2						J=8	

Berikutnya bergerak dari kanan ke kiri dimulai dari 58. Selanjutnya dilakukan pembandingan antara 37 dengan elemen pada J=8. Proses ini akan terus berjalan ke kiri (J-1) hingga didapat elemen yang lebih kecil dari 37. Proses ini kemudian berhenti pada elemen 28. Selanutnya kedua elemen ini ditukarkan, sehingga diperoleh isi larik yang baru, yaitu :

17	28	6	18	50	37	121	58	120
	I=2				J=6			

Berikutnya dilakukan pembandingan antara 37 dengan elemen pada I=2. Proses pembandingan akan terus berjalan ke kanan (I+1) hingga didapat elemen yang lebih besar dari 37. Proses ini kemudian berhenti pada elemen 50. Selanjutnya kedua elemen ini ditukarkan, sehingga diperoleh isi larik yang baru, yaitu :

17	28	6	18	37	50	121	58	120
				I=5	J=6			

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

Sekarang dimulai dari 50 (J=6) dilakukan pembandingan antara 37 dengan 50. Proses pembandingan akan terus berjalan ke kiri (J-1) hingga didapat elemen yang nilainya lebih kecil dari 37. Pada langkah ini ternyata nilai I dan J adalah sama. Hal ini berarti proses penempatan pivot yang bernilai 37 telah selesai. Disini bisa diketahui bahwa semua elemen yang terletak di sebelah kiri (bagian pertama), nilainya selalu lebih kecil dari 37, dan semua elemen yang terletak di sebelah kanan (bagian ketiga), nilainya selalu lebih besar dari 37, seperti terlihat di bawah ini.

17	28	6	18	37	50	121	58	120
bagian kiri					bagian kanan			

Hal seperti ini kemudian diulang terus-menerus pada dua bagian tersebut. Proses pengulangan berhenti jika setiap bagian hanya terdapat satu elemen.

Implementasi metode ini secara rekursif atau non rekursif akan dijelaskan berikut ini.

2.1.4.1 Metode Quicksort secara Rekursif

Rekursif didefinisikan sebagai suatu proses dari suatu objek yang dapat memanggil dirinya sendiri secara berulang-ulang. Hanya syarat atau kondisi tertentu yang dapat menyebabkan proses tersebut akan berhenti.

Proses rekusif untuk beberapa kasus merupakan algoritma yang baik dan dapat membuat pemecahan masalah lebih mudah. Akan tetapi sebagai imbalannya proses rekursif ini harus dibayar mahal dengan memori yang

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

banyak digunakan, dikarenakan setiap kali suatu sub program dipanggil, maka diperlukan sejumlah tambahan memori.

Metode Quicksort secara rekursif pun demikian. Di dalam metode Quicksort terjadi pemanggiilan terhadap dirinya sendiri secara terus-menerus. Dengan menggunakan cara rekursif terdapat tiga langkah yang harus dilakukan. Yang pertama adalah penempatan pivot pada posisinya yang tepat. Kemudian yang kedua adalah mengurutkan bagian pertama. Dan yang terakhir adalah mengurutkan bagian ketiga. Untuk mengurutkan bagian pertama atau bagian ketiga, ketiga langkah ini diulang kembali dengan menentukan batas-batas bagian yang akan diproses. Batas akhir proses rekursif ditentukan apabila batas kiri bagian sama dengan atau lebih besar dari batas kanan.

Metode Quicksort secara rekursif secara sederhana dapat dituliskan sebagai berikut :

```
Procedure Quicksort (Var A : Larik ; Awal, Akhir : Integer) ;
Var I,J : Integer ;
Begin
If Awal < Akhir then
  Begin
   ATUR ;                          {menempatkan pivot pada posisi J}
   QUICKSORT(A,Awal,J-1) ;         {memproses bagian pertama}
   QUICKSORT(A,J+1,Akhir) ;        {memproses bagian ketiga}
  End ;
```


This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

End ;

Prosedur di atas dalam program utama dipanggil dengan :

QUICKSORT(A,1,N) ;

Dengan A adalah larik yang akan diurutkan dan N adalah banyaknya data.

2.1.4.2 Metode Quicksort secara Non Rekusif

Implementasi secara non rekusif memerlukan dua buah tumpukan yang digunakan untuk menyimpan batas-batas bagian. Tumpukan pertama mencatat batas-kiri bagian dan tumpukan kedua mencatas batas-kanan bagian. Masing-masing elemen pada kedua tumpukan ini harus bertipe integer karena mencatat subskrip suatu elemen. Dalam prosedur yang akan disajikan hanya digunakan sebuah tumpukan, tetapi elemennya bertipe record yang terdiri dari dua buah medan, yaitu Kiri (untuk mencatat batas kiri), dan Kanan (untuk mencatat batas kanan). Dalam hal ini tumpukan dideklarasikan sebagai array.

Dengan mengacu pada contoh di atas, sebelum metode ini dijalankan , medan kiri diatur sama dengan satu, dan medan kanan sama dengan N. Pada contoh 2.3, N = 9. Sebuah elemen yang bernilai 37 dibawa pada posisinya yang tepat maka isi tumpukan akan berubah seperti diperlihatkan pada gambar 2.5.

Sesuai dengan sifat tumpukan, maka bagian paling atas akan dikerjakan lebih dahulu. Sehingga, dengan memperhatikan Gambar 2.5b, maka bagian atas (dengan batas kiri = 6 dan batas kanan = 9) akan dikerjakan lebih dahulu.

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

diharapkan program yang tersusun akan bisa dijalankan dengan baik dan memberikan hasil seperti yang diharapkan.

Struktur program Pascal memiliki bagian-bagian seperti pada gambar 6. Program bisa diawali dengan nama program (diawali dengan kata baku/reserved word program), diikuti deklarasi piranti (diawali dengan kata uses), deklarasi label statement (diawali dengan kata baku label), deklarasi konstanta (diawali dengan kata baku const), deklarasi tipe data (diawali dengan kata baku type) dan deklarasi perubah dari program utama (diawali dengan kata baku var); prosedur-prosedur, fungsi-fungsi dan akhirnya bagian program utama sendiri.

```
Program ..... ;                          { Kepala Program }
Uses .......                             { Deklarasi Piranti }
Label .......                            { Deklarasi Label }
Const ......                             { Deklarasi Konstanta }
Type ......                              { Deklarasi Tipe Data }
Var ......                               { Deklarasi Perubah }
Procedure ..... ;
Function ..... ;
Begin                                    { Awal Program Utama }
.
.< Statemen-statemen dari program Bagan_program >
..
End.                                     { Akhir Program Utama }
```

Gambar 2.6. Struktur Program Pascal

Secara umum struktur program Pascal dibagi menjadi tiga bagian yaitu :

1. Kepala program, yang terdiri dari kata program dan judul program.

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

2. Bagian deklarasi, yang terdiri dari deklarasi piranti, deklarasi label, deklarasi konstanta, deklarasi tipe data, dan deklarasi perubah.
3. Bagian statement, yang dimulai dengan kata begin dan diakhiri kata end.

Dalam hal ini akan dijelaskan prosedur dan fungsi sebagai bagian penting yang akan digunakan dalam pembuatan program.

2.2.2 Prosedur

Prosedur mempunyai struktur yang sama dengan program, yaitu nama prosedur, deklarasi-deklarasi dan bagian utama prosedut itu sendiri.

Bentuk umum prosedur adalah :

Procedure nama(parameter formal) ;

Dengan nama : nama prosedur

Parameter formal : daftar parameter formal

Parameter formal terdapat dua macam, yaitu parameter nilai dan parameter perubah. Parameter nilai adalah parameter yang tidak diawali dengan kata baku var. Parameter perubah adalah parameter yang diawali dengan kata baku var. Parameter nilai mempunyai sifat bahwa meskipun nilai parameter tersebut di dalam prosedur telah mengalami perubahan, namun perubahannya tidak akan dibawa ke dalam program utama. Sedangkan parameter perubah memiliki sifat kebalikannya. Yaitu jika di dalam prosedur nilai parameter

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

diharapkan program yang tersusun akan bisa dijalankan dengan baik dan memberikan hasil seperti yang diharapkan.

Struktur program Pascal memiliki bagian-bagian seperti pada gambar 6. Program bisa diawali dengan nama program (diawali dengan kata baku/reserved word program), diikuti deklarasi piranti (diawali dengan kata uses), deklarasi label statement (diawali dengan kata baku label), deklarasi konstanta (diawali dengan kata baku const), deklarasi tipe data (diawali dengan kata baku type) dan deklarasi perubah dari program utama (diawali dengan kata baku var); prosedur-prosedur, fungsi-fungsi dan akhirnya bagian program utama sendiri.

```
Program ..... ;                          { Kepala Program }
Uses .......                             { Deklarasi Piranti }
Label .......                            { Deklarasi Label }
Const ......                             { Deklarasi Konstanta }
Type ......                              { Deklarasi Tipe Data }
Var ......                               { Deklarasi Perubah }
Procedure ..... ;
Function ..... ;
Begin                                    { Awal Program Utama }
.
.< Statemen-statemen dari program Bagan_program >
..
End.                                     { Akhir Program Utama }
```

Gambar 2.6. Struktur Program Pascal

Secara umum struktur program Pascal dibagi menjadi tiga bagian yaitu :

1. Kepala program, yang terdiri dari kata program dan judul program.

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

mengalami perubahan, maka perubahnnya akan dibawa ke dalam program utama

.

Contoh 2.4 :

Procedure NILAI(Var Angka1,Angka2 :Integer) ;

Prosedur di atas dipanggil dengan :

NILAI(Nilai1,Nilai2) ;

2.2.3 Fungsi

Pada dasarnya fungsi hampir sama dengan prosedur. Perbedaannya adalah nama fungsi berfungsi sekaligus sebagai suatu perubah dan dalam deklarasi fungsi harus dinyatakan tipe datanya.

Bentuk umum fungsi adalah :

Function nama<(dafpar)> : tipe ;

Dengan nama : nama fungsi

dafpar : daftar parameter formal

tipe : tipe data dari fungsi tersebut

Semua parameter formal dalam fungsi harus berupa parameter nilai.

Contoh 2.5 :

Function JUMLAH(A : Larik; N : Integer) : integer ;

Dalam program utama fungsi tersebut dapat dipanggil dengan :

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

```
Writeln('Nilai total = ',JUMLAH(Barisan,Maks) ;
```

2.3 Running Time Program

Dalam memecahkan sebuah permasalahan, tersedia beberapa pilihan algoritma, dimana banyak kesulitan untuk memilihnya. Ada dua macam algoritma yang dapat menjadi dasar untuk pemilihan yaitu :

1. Algoritma yang mudah dipahami bagi pemakainya. Algoritma ini biasanya tidak mampu membuat komputer bekerja dengan cepat.
2. Algoritma yang dapat membuat komputer bekerja seefisien dan secepat mungkin. Algoritma ini biasanya adalah algoritma komplek yang sulit dibuat dan dipahami.

Jika dibutuhkan sebuah program yang hanya digunakan sekali atau beberapa kali saja, maka pilihan pertama lebih baik. Karena disini tidak terlalu dibutuhkan kecepatan dari proses program tersebut. Sebaliknya, jika dibutuhkan sebuah program yang digunakan sesering mungkin, maka pilihan kedua lebih baik. Kecepatan dari program tersebut mutlak dibutuhkan. Dengan semakin tinggi kecepatan maka semakin sedikit pula waktu yang dibutuhkan untuk menyelesaikan proses. Dan waktu yang dibutuhkan oleh komputer untuk menyelesaikan proses ini disebut running time, dinotasikan dengan T(n).

Ada beberapa faktor yang mempengaruhi running time dari sebuah program. Yaitu :

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

1. Input dari program tersebut.
2. Kecepatan bahasa pemrograman yang digunakan untuk menciptakan program tersebut.
3. Kecepatan komputer tersebut sendiri untuk mengeksekusi program.
4. Kompleksitas dari algoritma itu sendiri.
   - Pembandingan antara elemen yang satu terhadap yang lain
   - Jumlah penukaran antara elemen yang satu dengan yang lain

2.3.1 Input dari Program

Yang dimaksud input dari program adalah ukuran/panjang (n) dari input tersebut. Misalnya terdapat data 6,8,1,4,2,8,5. Data tersebut memiliki n=7. Data tersebut akan diurutkan secara urut turun. Sehingga menjadi 8,8,6,5,4,2,1. Kemudian terdapat data lain dengan n=10. Yaitu 23,11,34,22,45,12,44,56,34,37. Bila data tersebut diurutkan secara urut turun akan menjadi 56,45,44,37,34,34,23,22,12,11. Data dengan n=10 tentunya akan menghasilkan running time yang lebih lama dibandingkan dengan n=7.

2.3.2 Compiler dan CPU yang Digunakan

Faktor kedua dan ketiga menyatakan bahwa running time juga bergantung pada compiler yang digunakan untuk mengkompile dan CPU yang

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

digunakan untuk mengeksekusi. Semakin baik dan cepat bahasa pemrograman dan komputer yang digunakan akan semakin sedikit pula running time yang dihasilkan.

2.3.3 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..

Tabel 2.2 menunjukkan beberapa terminologi untuk mendiskripsikan kompleksitas waktu algoritma.

Kompleksitas	**Terminologi**
O(1)	Kompleksitas konstanta
O(log n)	Kompleksitas logaritma
O(n)	Kompleksitas linier
O(n log n)	Kompleksitas n log n
$O(n^b)$	Kompleksitas polinomial
$O(b^n)$	Kompleksitas eksponensial
O(n!)	Kompleksitas faktorial

Tabel 2.2 Terminologi kompleksitas algoritma.

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

Dalam menentukan analisa algoritma nanti, harus diketahui tiga hal berikut ini, yaitu :

1. Worst case running time yaitu suatu keadaan yang merupakan nilai maksimum dari fungsi T(n) untuk setiap input yang mungkin atau keadaan yang terburuk dari proses dalam suatu algoritma, sehingga waktu yang ditempuh adalah waktu yang maksimum.
2. Average case running time merupakan keadaan dari waktu tempuh yang ekivalen dengan nilai ekspetasi dari fungsi T(n) untuk setiap input data yang mungkin. Nilai ekspetsi dari fungsi T(n) = E, yang didefinisikan sebagai :

   $E = n_1p_1 + n_2p_2 + \ldots + n_kp_k$

   Dengan :

   n_1, n_2, n_k merupakan nilai yang muncul.

   p_1, p_2, p_k merupakan probabilitas dari setiap nilai (n) yang muncul.
3. Best case running time yaitu keadaan yang merupakan nilai minimum dari fungsi T(n) untuk setiap input yang mungkin atau keadaan yang terbaik dari proses suatu algolritma sehingga waktu yang ditempuh adalah waktu yang minimum

Dalam menentukan running time dari 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.

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

2. Worst case running time dari suatu algoritma merupakan suatu batas atas dari suatu running time. Dengan demikian akan memberikan jaminan 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.

2.4 Penghitungan Running Time Program dengan Fungsi Pertumbuhan

2.4.1 Notasi Big-Oh

Definisi :

T(n) adalah O(f(n)), jika terdapat konstanta positif c dan n_0 sedemikian sehingga $T(n) \leq c(f(n))$, untuk setiap $n \geq n_0$.

Contoh 2.6:

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

Penyelesaian :

Dicoba untuk $n_0 = 0$, maka berdasarkan definisi dapat diambil $n = 0$.

Sehingga berlaku :

$T(n) \leq c(f(n))$

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

$(n + 1)^2 \leq c.n^2$

$(0+1)^2 \leq c.0^2$

$1 \leq 0$

Percobaan salah, maka untuk konstanta $n_0 = 0$ tidak dapat diambil.

Sekarang akan dicoba untuk $n_0 = 1$, maka berdasarkan definisi dapat diambil $n = 1$

Sehingga berlaku :

$T(n) \leq c(f(n))$

$(n + 1)^2 \leq c.n^2$

$(1+1)^2 \leq c.1^2$

$4 \leq c$, diambil $c = 4$.

Dengan demikian dapat diambil $n_0 = 1$ dan $c = 4$, supaya berlaku $(n+1)^2 \leq 4n^2$, untuk setiap $n \geq 1$.

Contoh 2.7

Buktikan bahwa $T(n) = 3^n$ adalah bukan $O(2^n)$.

Penyelesaian :

Andaikan terdapat konstanta positif n_0 dan c sedemikan sehingga untuk setiap $n \geq n_0$, berlaku $3^n \leq c2^n$. Sehingga dapat diartikan bahwa $c \geq (3/2)^n$ untuk sembarang $n \geq n_0$. Tetapi $(3/2)^n$ akan semakin besar sebagaimana n yang semakin besar. Jadi tidak ada konstanta positif c yang dapat memenuhi $c \geq (3/2)^n$ untuk setiap n. Sehingga $T(n) = 3^n$ bukan $O(2^n)$.

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

2.4.2 Penjumlahan dan Perkalian dalam Notasi Big-Oh

Sebelum menghitung running time, maka harus terlebih dahulu diketahui cara penjumlahan dan penggandaan dalam notasi "big-oh".

Teorema 2.1

Andaikan bahwa $T_1(n)$ dan $T_2(n)$ adalah running time dari dua program P_1 dan P_2. $T_1(n)$ adalah $O(f(n))$ dan $T_2(n)$ adalah $O(g(n))$. Maka running time dari P_1 ditambah dengan P_2, $T_1(n) + T_2(n)$, adalah $O(\max(f(n),g(n)))$.

Bukti :

Ambil beberapa konstanta c_1,c_2,n_1,n_2

Jika $n \geq n_1$ maka $T_1(n) \leq c_1f(n)$

Jika $n \geq n_2$ maka $T_2(n) \leq c_2g(n)$.

Misalkan $n_0 = \max(n1,n2)$.

Jika $n \geq n_0$, maka berlaku $T_1(n) + T_2(n) \leq c_1f(n) + c_2g(n)$.

Dapat disimpulkan :

Jika $n \geq n_0$, maka $T_1(n) + T_2(n) \leq (c_1+c_2)\max(f(n),g(n))$.

Maka kombinasi dari running time $T_1(n) + T_2(n)$ adalah $O(\max(f(n),g(n)))$.

Teorema 2.1 di atas disebut aturan penjumlahan (rule of sum).

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

Contoh 2.8:

Aturan penjumlahan di atas dapat digunakan untuk menghitung running time yang merupakan sebuah barisan dari langkah-langkah program, dimana setiap langkah mungkin terdiri dari looping dan cabang-cabang. . Andaikan kita mempunyai 3 langkah yang running timenya adalah $O(n^2)$, $O(n^3)$, dan O(nlogn). Kemudian running time dari dua langkah pertama adalah $O(\max(n^2,n^3))$ dimana akan menjadi $O(n^3)$. Dan running time dari ketiga-tiganya adalah $O(\max(n^3,n\log n))$ dimana akan menjadi $O(n^3)$.

Teorema 2.2

Andaikan bahwa $T_1(n)$ dan $T_2(n)$ adalah running time dari dua program P_1 dan P_2. $T_1(n)$ adalah $O(f(n))$ dan $T_2(n)$ adalah $O(g(n))$. Maka running time dari P_1 dikalikan dengan P_2, $T_1(n) . T_2(n)$, adalah $O(\max(f(n),g(n)))$.

Bukti :

Ambil beberapa konstanta c_1,c_2,n_1,n_2

Jika $n \geq n_1$ maka $T_1(n) \leq c_1 f(n)$

Jika $n \geq n_2$ maka $T_2(n) \leq c_2 g(n)$.

Misalkan $n_0 = \max(n1,n2)$.

Jika $n \geq n_0$, maka berlaku $T_1(n) . T_2(n) \leq c_1 f(n) . c_2 g(n)$.

Dapat disimpulkan :

Jika $n \geq n_0$, maka $T_1(n) . T_2(n) \leq (c_1 . c_2)\max(f(n),g(n))$.

Maka kombinasi dari running time $T_1(n) . T_2(n)$ adalah $O(\max(f(n),g(n)))$.

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

Teorema 2.2 tersebut disebut aturan perkalian (rule of product).

Contoh 2.9

Contoh berikut ini adalah contoh penghitungan running time dari pengurutan dengan metode buble sort.

```
    Procedure buble (var A: array[1..n] of integer) ;
        {buble sorts array A into increasing order}
    var
        i , j , temp : integer ;
        begin
1)          for i := 1 to n-1 do
2)          for j := n downto i+1 do
3)              if A[j-1] > A[j] then begin
                    { swap A[j-1] and A[j] }
4)                  temp := A[j-1] ;
5)                  A[j-1] := A[j] ;
6)                  A[j] := temp ;
                End ;
        End ; {buble}
```

Observasi pertama adalah bahwa setiap pernyataan yang menyatakan penugasan akan mengambil suatu konstanta running time, tidak tergantung

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

dari jumlah input. Disini pernyataan 4, 5, dan 6 masing-masing mempunyai running time O(1). Perhatikan bahwa O(1) adalah notasi untuk suatu konstanta jumlah. Dengan aturan penjumlahan, kombinasi running time dari ketiganya adalah $T_1(n)+T_2(n)+T_3(n) = O(\max(1,1,1))) = O(1)$.

Sekarang akan dihitung running time pada baris 3-6. Untuk pernyataan if, diuji kondisi mensyaratkan running time O(1). Belum diketahui apakah bagian dari pernyataan if (baris 4-6) akan dieksekusi atau tidak. Dari pembahasan diatas maka dicari running time dengan kondisi terburuk, dan diasumsikan bahwa hal tersebut terjadi. Jadi bagian dari pernyataan if (baris 3-6) mempunyai running time O(1).

Sekarang akan diperhatikan untuk looping pada baris 2-6. Untuk baris 2-6, bagian tubuh looping yaitu baris 3-6, mempunyai running time O(1) untuk setiap iterasi. Jumlah iterasi pada looping adalah n menurun sampai i+1 adalah sebagai berikut :

Andaikan terdapat data dengan ukuran n = 5, maka jumlah iterasi pada baris kedua adalah

i	J = n	J = i+1	Jumlah Iterasi (JI)
1	5	2	4
2	5	3	3
3	5	4	2
4	5	5	1

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

Sehingga tabel tersebut dapat disusun sebagai berikut :

$4 = 5 - (2 - k)$

$3 = 5 - (3 - k)$

$2 = 5 - (4 - k)$

$1 = 5 - (5 - k)$

Akan dicoba untuk k = 0, apakah memenuhi persamaan-persamaan di atas :

$4 \;..\; 5 - (2 - 0)$

$4 \neq 3$

$3 \;..\; 5 - (3 - 0)$

$3 \neq 2$

$2 \;..\; 5 - (4 - 0)$

$2 \neq 1$

$1 \;..\; 5 - (5 - 0)$

$1 \neq 0$

Sekarang akan dicoba untuk k= 1,

$4 \;..\; 5 - (2 - 1)$

$4 = 4$

$3 \;..\; 5 - (3 - 1)$

$3 = 3$

$2 \;..\; 5 - (4 - 1)$

$2 = 2$

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

1 .. 5 - (5 - 1)

1 = 1

Ternyata untuk k = 1, benar, sehingga persamaan-persamaan di atas dapat dibuat menjadi satu rumus yaitu :

$\Pi = n - (i+1 - 1)$

$\Pi = n - i$

Maka jumlah iterasi pada baris kedua adalah n-i kali

Sehingga dengan aturan untuk perkalian, looping pada baris 2-6 akan mempunyai running time O((n-i)x1) yaitu O(n-i).

Sekarang akan dilihat keseluruhan dari luar looping. Pernyataan 1 dieksekusi sebanyak n-1 kali. Sehingga total running timenya adalah

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

Jadi total running time dari program tersebut yaitu $O(n^2)$.

Sekarang dapat disimpulkan beberapa aturan untuk menganalisis program.

1. Running time dari setiap pernyataan penugasan, pernyataan read, dan pernyataan write akan mempunyai O(1).
2. Running time dari sebuah pernyataan dari barisan ditentukan dengan aturan penjumlahan. Jadi running timenya adalah running time terbesar dari keseluruhan pernyataan pada barisan tersebut.

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)

3. Running time untuk pernyataan kondisi (if) adalah tergantung dari pernyataan eksekusinya ditambah dengan running time untuk mengaevaluasi kondisi. Running time yang normal untuk mengevaluasi kondisi adalah O(1). Running time untuk sebuah if - then – else adalah running time untuk mengevaluasi kondisi ditambah dengan running time pada saat pernyataan dieksekusi waktu kondisinya benar atau salah.
4. Running time untuk mengeksekusi sebuah looping adalah perkalian dari keseluruhan bagian pernyataan-pernyataan dari looping tersebut (tubuh looping) dengan running time untuk mengevaluasi kondisi (biasanya O(1)).

This document is Undip Institutional Repository Collection. The author(s) or copyright owner(s) agree that UNDIP-IR may, without changing the content, translate the submission to any medium or format for the purpose of preservation. The author(s) or copyright owner(s) also agree that UNDIP-IR may keep more than one copy of this submission for purpose of security, back-up and preservation:
(http://eprints.undip.ac.id)