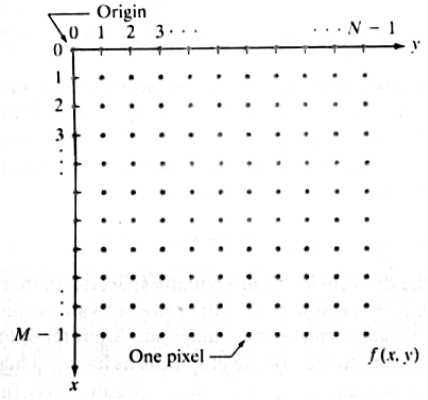
BAB 2
LANDASAN TEORI

2.1.Citra Digital

Pengolahan citra digital adalah proses pengolahan gambar dua dimensi menggunakan komputer, citra digital adalah sebuah larik (*array*) yang berisi nilai-nilai real maupun komplek yang direpresentasikan dengan deretan bit tertentu [6].

Suatu citra dapat didefinisikan sebagai fungsi f(x,y) berukuran M baris dan N kolom, dengan x dan y adalah koordinat spasial, dan amplitudo f pada titik koordinat (x,y) dinamakan intensitas atau tigkat keabuan dari citra pada titik tersebut. Apabila nilai x,y, dan nilai amplitudo f secara keseluruhan berhingga (finite) dan bernilai diskrit maka dapatdikatakan bahwa citra tersebut adalah citra digital, gambar 2.1 menunjukan posisi koordinat citra digital [6].



Gambar 2.1 *Koordinat* Citra Digital [6]

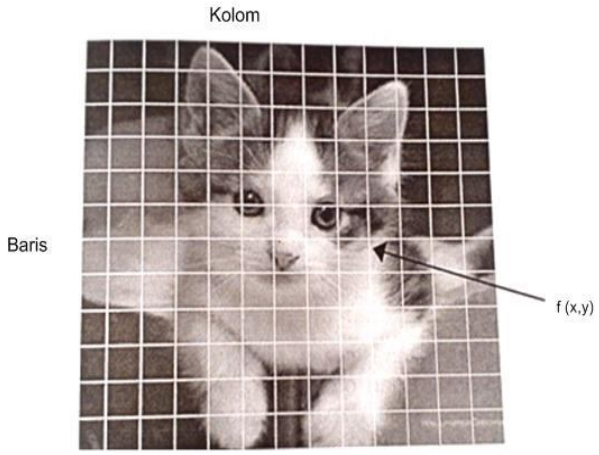
Citra digital dapat ditulis dalam bentuk matriks seperti gambar 2.2

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Gambar 2.2 Matriks Citra Digital [6]

Nilai pada suatu irisan pada baris dan kolom (pada posisi x,y) disebut dengan *picture elements, image elements, pels, atau pixels*. Istilah pixel paling

sering digunakan pada citra digital. Gambar 2.3 menunjukan ilustrasi digitalisasi citra dengan M=16 baris dan N=16 kolom.



Gambar 2.3 Ilustrasi Digitalisasi Citra [6]

2.1.1. Penerapan Citra Digital Bidang Biomedis

Pengolahan citra digital mengalami kemajuan penting dalam bidang kedokteran ketika ditemukannya tomografi terkomputerisasi (*Computerized Tomography/CT*) pada tahun 1970-an dan kini teknologi tomografi tersebut sudah maju sangat pesat. Pengolahan citra digital dapat digunakan untuk deteksi tumor atau kanker rahim, identifikasi penyakit paru-paru, identifikasi penyakit hati, identifikasi penyakit tulang, segmentasi tulang dari otot yang lainnya, klasifikasi gigi dan analisis citra mikroskopis. Beberapa dari kemajuan pada bidang kedokteran tersebut karena kemampuan pengolahan citra digital mampu menginterpretasikan sinar x (x *ray*). Kemampuan lainnya adalah aplikasi volumetric 3D Magnetic Resonance Imaging (MRI) yang mampu mendapatkan pencitraan organ dalam tubuh manusia secara jelas dengan menggunakan *scanner* MRI [6].

2.1.2. Citra MRI

Magnetic Resonance Imaging (MRI) adalah suatu alat kedokteran di bidang pemeriksaan diagnostik radiologi , yang menghasilkan rekaman gambar potongan penampang tubuh / organ manusia dengan menggunakan medan magnet

berkekuatan antara 0,064 – 1,5 tesla (1 tesla = 1000 Gauss) dan resonansi getaran terhadap inti atom hydrogen [7].

Selanjutnya MRI bila ditinjau dari tipenya terdiri dari [7]:

1. MRI yang memiliki kerangka terbuka (open gantry) dengan ruang yang luas.
2. MRI yang memiliki kerangka (gantry) biasa yang berlorong sempit.

Sedangkan bila ditinjau dari kekuatan magnetnya terdiri dari [7] :

a. MRITesla tinggi (High Field Tesla) memiliki kekuatan di atas 1 – 1,5 T
b. MRI Tesla sedang (Medium Field Tesla) memiliki kekuatan 0,5 – 1T
c. MRI Tesla rendah (Low Field Tesla) memiliki kekuatan di bawah 0,5 T.

Sebaiknya suatu rumah sakit memilih MRI yang memiliki tesla tinggi karena alat tersebut dapat digunakan untuk tehnik Fast Scan yaitu suatu teknik yang memungkinkan 1 gambar irisan penampang dibuat dalam hitungan detik, sehingga kita dapat membuat banyak irisan penampang yang bervariasi dalam waktu yang sangat singkat, dengan banyaknya variasi gambar membuat suatu lesi menjadi menjadi lebih spesifik.

Ada beberapa kelebihan MRI dibandingkan dengan pemeriksaan CT Scan yaitu [7]:

a. MRI lebih unggul untuk mendeteksi beberapa kelainan pada jaringan lunak seperti otak, sumsum tulang serta muskuloskeletal.
b. Mampu memberi gambaran detail anatomi dengan lebih jelas.
c. Mampu melakukan pemeriksaan fungsional seperti pemeriksaan difusi, perfusi dan spektroskopi yang tidak dapat dilakukan dengan CT Scan.
d. Mampu membuat gambaran potongan melintang, tegak, dan miring tanpa merubah posisi pasien.
e. MRI tidak menggunakan radiasi pengion.

Alat MRI terlihat seperti gambar 2.4



Gambar 2.4 Alat MRI [7]

2.1.3. Pengolahan Citra Digital

Pengolahan citra adalah kegiatan memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia/mesin(komputer). Inputannya adalah citra dan keluarannya juga citra tapi dengan kualitas lebih baik daripada citra masukan.

Proses pengolahan citra yang digunakan pada penelitian ini terdiri dari proses *scaling, grayscale, thresholding.*

2.1.3.1.Scaling

Scaling atau Penskalaan citra adalah sebuah operasi geometri yang memberikan efek memperbesar atau memperkecil ukuran citra *input* sesuai dengan variabel penskalaan citranya. *Scaling* digunakan untuk memperbesar (*zoom-in*) atau memperkecil (*zoom-out*) citra [6]. Rumus yang digunakan untuk proses *scaling* terlihat seperti persamaan 2.1.

$$x' = Sh\ x \qquad y' = Sv\ y \qquad (2.1)$$

Keterangan :

Sh : faktor skala horizontal

Sv : faktor skala vertikal

2.1.3.2.Grayscale

Citra *grayscale* merupakan citra digital yang hanya memiliki satu nilai kanal pada setiap pixelnya, dengan kata lain nilai bagian *RED = GREEN = BLUE*. Nilai tersebut digunakan untuk menunjukan tingkat intensitas. Warna yang

dimiliki adalah warna dari hitam, keabuan, dan putih. Tingkatan kebuan disini merupakan warna abu dengan berbagai dari hitam hingga mendekati putih. Citra *grayscale* memiliki kedalaman warna 8 bit (256 kombinasi warna keabuan) [6]. Rumus yang digunakan untuk grayscale terlihat seperti persamaan 2.2.

$$\text{Grayscale} = (R + G + B)/3 \qquad (2.2)$$

Keterangan :

R = Red (menyatakan warna merah)

G = Green (menyatakan warna hijau)

B = Blue (menyatakan warna biru)

2.1.3.3.Tresholding

Tresholding atau pengambangan membuat citra memiliki dua tingkat keabuan yaitu hitam dan putih, proses pengambangan akan menghasilkan citra biner [6]. Proses tresholding mengikuti aturan dari persamaan 2.3.

$$g\,(x,y) = \begin{cases} 1 \; if \; f\,(x,y) \geq T \\ 0 \; if \; f\,(x,y) < T \end{cases} \qquad (2.3)$$

Keterangan :

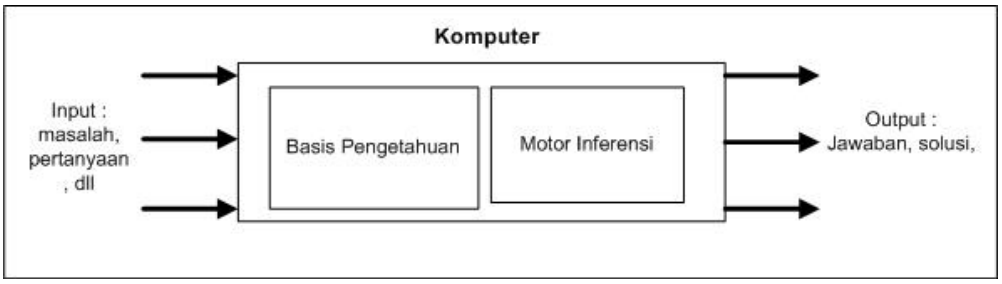
g (x,y) : citra biner dari citra *grayscale*

T : nilai ambang

2.2.Kecerdasan Buatan

Kecerdasan buatan *(Artificial Intelligence)* merupakan salah satu ilmu komputer yang membuat agar mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia [5]. Kecerdasan buatan mempunyai dua bagian utama yang sangat dibutuhkan dalam melakukan aplikasi, seperti terlihat pada gambar 2.5 komponen yang dibutuhkan adalah :

a. Basis Pengetahuan *(Knowledge Base)*, berisi fakta-fakta, teori, pemikiran dan hubungan antara satu dengan lainnya [5].
b. Motor Inferensi *(Inference Engine)*, yaitu kemampuan menarik kesimpulan berdasarkan pengalaman [5].



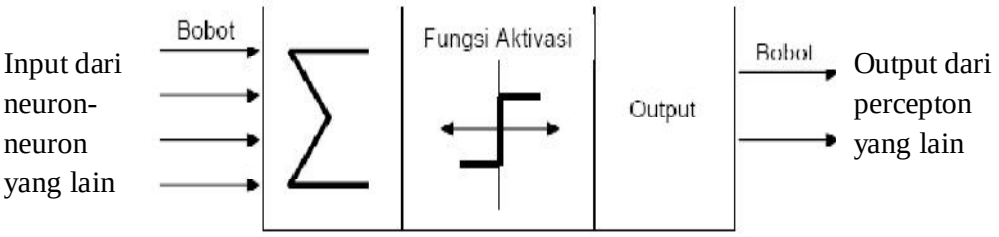
Gambar 2.5 Penerapan Kecerdasan Buatan [5]

Kecerdasan buatan terdiri dari beberapa kajian diantaranya adalah sistem pakar, jaringan syaraf tiruan, *computer vision*, simulasi *crowd*, logika *fuzzy*, algoritma genetika, dokumen *minning.*

2.2.1. Jaringan Syaraf Tiruan

Jaringan Syaraf Tiruan *(neural network)* adalah salah satu kajian pada kecerdasan buatan. Jaringan syaraf tiruan adalah salah satu representasi buatan dari otak manusia yang selalu mencoba untuk menstimulasikan proses pembelajaran pada otak manusia tersebut [5]. Istilah buatan maksudnya adalah jaringan syaraf yang diimplementasikan menggunakan program computer untuk menyelesaikan sejumlah proses perhitungan selama proses pembelajaran.

Komponen JST terdiri dari neuron – neuron dan neuron –neuron tersebut saling berhubungan. Neuron – neuron tersebut akan mentransformasikan informasi yang diterima melalui sambungan keluarnya menuju neuron – neuron yang lain, pada JST hubungan neuron-neuron tersebut dikenal dengan nama bobot [2]. Komponen JST seperti terlihar pada gambar 2.6 terdiri dari input (informasi) bobot (nilai- nilai tertentu), fungsi aktivasi berfungsi apabila input yang dimasukan sesuai dengan nilai ambang *(threshold)* yang ditentukan jika tidak sesuai maka fungsi aktivasi tidak diaktifkan, dan apabila neuron – neuron tersebut diaktifkan maka neuron akan mengirimkan output melalui bobot – bobot.



Gambar 2.6 Komponen Jaringan Syaraf Tiruan [5]

Jaringan Syaraf tiruan mempunyai dua macam proses pembelajaran yaitu pembelajaran terawasi dan pembelajaran tidak terawasi. Pembelajaran terawasi adalah jika output yang diharapkan telah diketahui sebelumnya, sedangkan pembelajaran tidak terawasi adalah proses pembelajaran yang tidak memerlukan target output [5].

1. Pembelajaran Terawasi (*Supervised Learning*)

   Metode pembelajaran pada jaringan syaraf disebut terawasi jika output yang diharapkan telah diketahui sebelumnya. Pada proses pembelajaran, satu pola input akan diberikan ke satu neuron pada lapisan input. Pola ini akan dirambatkan di sepanjang jaringan syaraf hingga sampai ke neuron pada lapisan output. Lapisan output ini akan membangkitkan pola output yang nantinya akan dicocokkan dengan pola output targetnya. Apabila terjadi perbedaan antara pola output hasil pembelajaran dengan pola target, maka disini akan muncul error. Apabila nilai error ini masih cukup besar, mengindikasikan bahwa masih perlu dilakukan lebih banyak pembelajaran lagi. Metode pembelajaran terawasi diantaranya *Hebb Rule, Perceptron, Delta Rule, Backpropagation, Learning Vektor Quantization, Heteroassociative Memory*.

2. Pembelajaran Tak Terawasi (*Unsupervised Learning*)

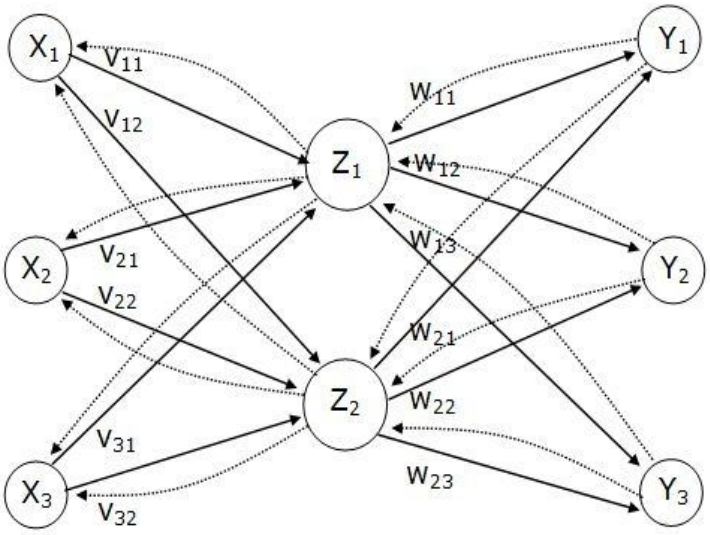
   Pada metode pembelajaran yang tak terawasi ini tidak memerlukan target output. Pada metode ini, tidak dapat ditentukan hasil yang seperti apakah yang diharapkan selama proses pembelajaran. Metode pembelajaran tidak terawasi salah satunya adalah jaringan kohonen

2.2.2. Backpropagation

Backpropagation merupakan algoritma pembelajaran yang terawasi dan biasanya digunakan oleh perceprton dengan banyak lapisan untuk mengubah bobot-bobot yang terhubung dengan neuron-neuron yang ada pada lapisan tersembunyinya. Algoritma *backpropagation* menggunakan *error output* untuk mengubah nilai bobot-bobotnya dalam arah mundur (*backward*). Untuk mendapatkan error ini, tahap perambatan maju (*forward*) harus dikerjakan terlebih dahulu. Pada saat perambatan maju, neuron-neuron diaktifkan dengan menggunakan fungsi aktivasi sigmoid, seperti terlihat pada rumus 2.4. merupakan fungsi aktivasi sigmoid biner, yaitu [5] :

$$f(x) = \frac{1}{1+e^{-x}} \qquad (2.4)$$

Arsitektur *backpropagation* terlihat seperti gambar 2.7, jaringan terdiri atas 3 unit (neuron) pada lapisan input yaitu $x_1, x_2,$ dan x_3; 1 lapisan tersembunyi dengan 2 neuron, yaitu z_1 dan z_2; serta 1 unit pada lapisan output, yaitu y. bobot yang menghubungkan $x_1, x_2,$ dan x_3 dengan neuron pertama pada lapisan tersembunyi adalah $v_{11} v_{21} v_{31}$(v_{ij} : Bobot yang menghubungkan neuron input ke-i ke neuron ke-j pada lapisan tersembunyi). Bobot yang menghubungkan z_1 dan z_2 dengan neuron pada lapisan output, adalah w_1 dan w_2. Fungsi aktivasi yang digunakan antara lapisan input dan lapisan tersembunyi, dan lapisan tersembunyi dengan lapisan *output* adalah fungsi aktifasi sigmoid biner [5].



Gambar 2.7 Arsitektur jaringan *backpropagation* [2].

Algoritma *backpropagation* [5]:

Inisialisasi bobot (ambil bobot awal dengan nilai random).

Kerjakan langkah-langkah berikut selama kondisi berhenti bernilai FALSE:

1. Untuk tiap-tiap pasangan elemen yang akan dilakukan pembelajaran, kerjakan:

   Feedforward:

   a. Tiap-tiap unit input (X_i, i=1,2,3,...,n) menerima sinyal x_i dan meneruskan sinyal tersebut ke semua unit pada lapisan yang ada di atasnya (lapisan tersembunyi).

   b. Tiap-tiap unit tersembunyi (Z_i, j=1,2,3,...,p) menjumlahkan sinyal-sinyal input terbobot:

   $$z_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \qquad (2.5)$$

   gunakan fungsi aktivasi untuk menghitung sinyal outputnya:

   $$z_j = f(z_in_j) \qquad (2.6)$$

   kirimkan sinyal tersebut ke semua unit di lapisan atasnya (unit-unit output).

   c. Tiap-tiap unit output (Y_k, k=1,2,3,...,m) menjumlahkan sinyal-sinyal input terbobot.

$$y_in_k = w_{0k} + \sum_{i=1}^{p} z_i w_{jk} \qquad (2.7)$$

gunakan fungsi aktivasi untuk menghitung sinyal outputnya:

$$y_k = f(y_in_k) \qquad (2.8)$$

dan kirimkan sinyal tersebut ke semua unit di lapisan atasnya (unit-unit output).

Backpropagation

d. Tiap-tiap unit output (Y_k, k=1,2,3,...,m) menerima target pola yang berhubungan dengan pola input pembelajaran, hitung informasi errornya:

$$\delta_k = (t_k - y_k)\, f'(y_in_k) \qquad (2.9)$$

kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai w_{jk}):

$$\Delta w_{jk} = \alpha\, \delta_k\, z_j \qquad (2.10)$$

hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai w_{0k}):

$$\Delta w_{0k} = \alpha\, \delta_k \qquad (2.11)$$

kirimkan δ_k ini ke unit-unit yang ada di lapisan bawahnya.

e. Tiap-tiap unit tersembunyi (Z_j, j=1,2,3,...,p) menjumlahkan delta inputnya (dari unit-unit yang berada pada lapisan di atasnya):

$$\delta_in_j = \sum_{k=1}^{m} \delta_k w_{jk} \qquad (2.12)$$

kalikan nilai ini dengan turunan dari fungsi aktivasinya untuk menghitung informasi error:

$$\delta_j = \delta_in_j\, f'(z_in_j) \qquad (2.13)$$

kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai v_{ij}):

$$\Delta v_{jk} = \alpha\, \delta_j\, x_i \qquad (2.14)$$

hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai v_{0j}):

$$\Delta v_{0j} = \alpha\ \delta_j \qquad (2.15)$$

f. Tiap-tiap unit output (Y_k, k=1,2,3,...,m) memperbaiki bias dan bobotnya (j=0,1,2,...,p):

$$w_{jk}(\text{baru}) = w_{jk}(\text{lama}) + \Delta w_{jk} \qquad (2.16)$$

Tiap-tiap unit tersembunyi (Z_j, j=1,2,3,...,p) memperbaiki bias dan bobotnya (i=0,1,2,...,n):

$$v_{ij}(\text{baru}) = v_{ij}(\text{lama}) + \Delta v_{ij} \qquad (2.17)$$

2. Tes kondisi berhenti

Keterangan Rumus :

z_in	:	Bobot sinyal unit tersembunyi
z	:	Keluaran unit tersembunyi
y_in	:	Bobot sinyal lapisan keluaran
y	:	Keluaran unit keluaran
δ	:	Kesalahan
Δw	:	Koreksi bobot unit keluaran
v_{ij}	:	Bobot yang menghubungkan neuron input ke-i ke neuron ke-j pada lapisan tersembunyi
Δv	:	Koreksi bobot unit tersembunyi
xi	:	Unit (neuron) pada lapisan input
n	:	Jumlah neuron (unit) pada lapisan input
v	:	Bobot awal masuk ke hidden
v0	:	Bobot bias yang menuju ke hidden
w	:	Bobot awal hidden ke output
w0	:	Bobot bias yang menuju ke output
α	:	Learning rate / rasio pembelajaran

2.3 Algoritma

Algoritma merupakan salah satu cabang ilmu komputer yang membahas prosedur penyelesaian suatu permasalahan. Algoritma adalah urutan langkah

langkah dalam menentukan suatu masalah. Algortima adalah serangkaian urutan langkah-langkah atau prosedur untuk menyelesaikan suatu masalah dengan memproses nilai masukan menjadi nilai keluaran.

Dalam mempelajari algortima, kita diperlukan untuk dapat menganalis algoritma dalam menentukan performansinya. Suatu Algoritma tidak saja harus benar, tetapi juga harus efisien dengan menghitung nilai dari performansinya. Setiap algortima memiliki nilai performansi yang berbeda-beda. Performansi suatu Algoritma dapat diukur dengan menghitung nilai kompleksitas waktunya dimana setiap langkahnya akan dihitung [8].

2.3.1 Kompleksitas Algoritma

Kompleksitas algoritma terdiri dari waktu dan ruang. Kompleksitas waktu asimptotik merupakan waktu yang dibutuhkan suatu Algoritma menyelesaikan tiap langkahnya. Setiap Algoritma memiliki kompleksitas waktu yang berbeda-beda. Komplesitas waktu asimptotik diperlukan untuk menghitung performansi suatu Algoritma. Untuk menghitung kompleksitas waktu asimptotik suatu Algoritma digunakanlah notasi "O-Besar" (*Big-O*) yang merupakan notasi kompleksitas waktu asimptotik. Definisi dari Big-O atau O(g(n)) adalah kumpulan semua fungsi yang *order of growth*-nya lebih kecil atau sama dengan g(n), sedangkan definisi dari *order of growth* adalah istilah yang dapat digunakan untuk pola varian jumlah input dalam suatu pengujian algoritma [8]. Contoh :

$$n \in O(n^2); 100n + 5 \in O(n^2)$$

$$\frac{1}{2}\, n(n-1) \in O\,(n^2)$$

$$n^3 \notin O(n^2); 0{,}0001n^3 \notin O(n^2);\ n^4 + n + 1 \notin O(n^2)$$

Perhitungan kompleksitas waktu asimptotik dengan menghitung nilai O-besar dari setiap instruksi di dalam Algoritma dengan contoh dibawah ini, kemudian diterapkan teorema O-Besar.

1. Pengisian nilai (*assignment*), perbandingan, operasi aritmetik, *read*, *write* membutuhkan waktu *O*(1).

2. Pengaksesan elemen larik atau memilih *field* tertentu dari sebuah *record* membutuhkan waktu $O(1)$.

Read(x);	$O(1)$
x:=x+a[k];	$O(1)+ O(1)+O(1)= O(1)$
Writeln(x);	$O(1)$

Kompleksitas waktu di atas adalah $O(1)$, didapat dari

$= O(1)+ O(1)+ O(1)$

$= O(\max(1,1))+ O(1)$

$= O(1)+ O(1)$

$= O(\max(1,1))$

$= O(1)$

3. If c then s1 else s2. Membutuhkan waktu Tc + max(Ts1,Ts2).

```
read(x); O(1)
if x mod 2=0 then O(1)
begin
 x:=x+1; O(1)
writeln(x);O(1)
end
else
writeln(x); O(1)
```

Kompleksitas waktu di atas adalah $O(1)$, didapat dari

$= O(1) + O(1) \max (O(1)+ O(1), O(1))$

$= O(1) + \max(O(1), O(1))$

$= O(1)$

4. Kalang for. Kompleksitas waktu kalang for adalah jumlah pengulangan dikali dengan kompleksitas waktu badan kalang.

For i=1 to n do	$O(n)$
Jumlah:= jumlah +a[i];	$O(1)$

Kompleksitas waktu di atas adalah $O(n)$, didapat dari

$= O(n) \, . \, O(1)$

$= O(n.1)$

$= O(n)$

5. While c do s; dan repeat s until c; untuk kedua buah kalang, kompleksitas waktunya adalah jumlah pengulangan dikali dengan waktu badan c dan s.

```
i:=2; O(1)
while i<=n do O(n)
begin
jumlah:=jumlah+a[i]; O(1)
i:= i +1; O(1)
end;
```

Kompleksitas waktu di atas adalah $O(1)$, didapat dari

$= O(1) + O(n)\{O(1)+O(1)\}$

$= O(1) + O(n)\ O(1)$

$= O(1) + O(n.1)$

$= O(1) + O(n)$

$= O(n)$

Berikut ini adalah pengelompokan algoritma berdasarkan notasi *O*-besar dapat dilihat pada Tabel 2.1.

Tabel 2.1 Pengelompokan Algoritma Berdasarkan Notasi *O*-Besar

Kelompok Algoritma	Nama
$O(1)$	konstan
$O(\log n)$	logaritmik
$O(n)$	lanjar
$O(n \log n)$	n log n
$O(n^2)$	kuadratik
$O(n^3)$	kubik
$O(2^n)$	eksponensial
$O(n!)$	faktorial

Urutan spektrum kompleksitas waktu algoritma adalah :

$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2)\ O(n^3) < \ldots < O(2^n) < O(n!)$

algoritma polinomial — algortima eksponensial

Penjelasan masing-masing kelompok algoritma dapat dilihat pada Tabel 2.2.

Tabel 2.2 Kelompok Algoritma dan Penjelasannya

Kelompok Algortima	Penjelasan

O(1)	Kompleksitas *O*(1) berarti waktu pelaksanaan algoritma adalah tetap, tidak bergantung pada ukuran masukan. Contohnya prosedur tukar di bawah ini:
	
	procedure tukar**(var** a : **integer**; **var** b : **integer**);
	var
	temp: **integer**;
	begin
	temp:=a;
	a:=b;
	b:=temp;
	end;
	
	Di sini jumlah operasi penugasan (*assignment*) ada tiga buah dan tiap operasi dilakukan satu kali. Jadi, $T(n) = 3 = O(1)$.

O(log n)	Kompleksitas waktu logaritmik berarti laju pertumbuhan waktunya berjalan lebih lambat daripada pertumbuhan *n*. Algoritma yang termasuk kelompok ini adalah algoritma yang memecahkan persoalan besar dengan mentransformasikannya menjadi beberapa persoalan yang lebih kecil yang berukuran sama (misalnya algoritma pencarian_biner). Di sini basis algoritma tidak terlalu penting sebab bila *n* dinaikkan dua kali semula, misalnya, log *n* meningkat sebesar sejumlah tetapan.
O(n)	Algoritma yang waktu pelaksanaannya lanjar umumnya terdapat pada kasus yang setiap elemen masukannya dikenai proses yang sama, misalnya algoritma pencarian_beruntun. Bila *n* dijadikan dua kali semula, maka waktu pelaksanaan algoritma juga dua kali semula.
O(n log n)	Waktu pelaksanaan yang *n* log *n* terdapat pada algoritma yang memecahkan persoalan menjadi beberapa persoalan yang lebih kecil, menyelesaikan tiap persoalan secara independen, dan menggabung solusi masing-masing persoalan. Algoritma yang diselesaikan dengan teknik bagi dan gabung mempunyai kompleksitas asimptotik jenis ini. Bila $n = 1000$, maka *n* log *n* mungkin 20.000. Bila *n* dijadikan dua kali semual, maka *n* log *n* menjadi dua kali semula (tetapi tidak terlalu banyak).

Kelompok Algoritma	Penjelasan
$O(n^2)$	Algoritma yang waktu pelaksanaannya kuadratik hanya praktis digunakan untuk persoalana yang berukuran kecil. Umumnya algoritma yang termasuk kelompok ini memproses setiap masukan dalam dua buah kalang bersarang, misalnya pada algoritma urut_maks. Bila *n* = 1000, maka waktu pelaksanaan algoritma adalah 1.000.000. Bila *n* dinaikkan menjadi dua kali semula, maka waktu pelaksanaan algoritma meningkat menjadi empat kali semula.
$O(n^3)$	Seperti halnya algoritma kuadratik, algoritma kubik memproses setiap masukan dalam tiga buah kalang bersarang, misalnya algoritma perkalian matriks. Bila *n* = 100, maka waktu pelaksanaan algoritma adalah 1.000.000. Bila *n* dinaikkan menjadi dua kali semula, waktu pelaksanan algoritma meningkat menjadi delapan kali semula
$O(2^n)$	Algoritma yang tergolong kelompok ini mencari solusi persoalan secara "*brute force*", misalnya pada algoritma mencari sirkuit Hamilton. Bila *n* = 20, waktu pelaksanaan algoritma adalah 1.000.000. Bila *n* dijadikan dua kali semula, waktu pelaksanaan menjadi kuadrat kali semula!
O(n!)	Seperti halnya pada algoritma eksponensial, algoritma jenis ini memproses setiap masukan dan menghubungkannya dengan *n* - 1 masukan lainnya, misalnya algoritma Persoalan Pedagang Keliling (*Travelling Salesperson Problem*). Bila *n* = 5, maka waktu pelaksanaan algoritma adalah 120. Bila *n* dijadikan dua kali semula, maka waktu pelaksanaan algoritma menjadi faktorial dari 2*n*.