BAB II

LANDASAN TEORI

2.1. Sistem Persamaan Linear

Suatu himpunan persamaan yang berbentuk

$$\begin{array}{ccccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & r_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & r_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \cdots & + & a_{mn}x_n & = & r_n \end{array} \qquad (2.1)$$

dinamakan suatu sistem persamaan linear dalam n besaran tidak diketahui $x_1, x_2, \ldots, x_n$. Jika $r_1, r_2, \ldots, r_n$ semuanya nol, maka sistem tersebut dinamakan *homogen*. Jika tidak semuanya nol dinamakan *tak-homogen*. Suatu himpunan bilangan $x1, x2, \ldots, xn$ yang memenuhi bentuk di atas dinamakan penyelesaian atau penyelesaian sistem.

Dalam bentuk matrik, (2.1) dapat ditulis sebagai berikut

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{pmatrix} \qquad (2.2)$$

atau disingkat

$$AX = R \qquad (2.3)$$

dimana A, X, dan R menyatakan matrik yang sesuai pada (2.2).

2.2. Matrik dan Vektor

Matrik terdiri dari susunan berbentuk siku empat yang elemen-elemennya dinyatakan oleh sebuah lambang tunggal. Seperti yang diperlihatkan pada gambar 2.4, [A] adalah cara penulisan pendek untuk matriks dan a_{ij} menunjukkan elemen matrik.

Himpunan elemen yang mendatar disebut *baris* dan himpunan tegak disebut *kolom*. Indeks pertama *i* selalu menunjuk nomor baris tempat elemen itu terletak dan indeks kedua *j* selalu menunjuk nomor kolom tempat elemen itu terletak. Misalnya elemen a_{23} berada di baris 2 dan kolom 3.

Matrik dalam gambar 2.4 mempunyai *m* baris dan *n* kolom, serta dikatakan *berukuran* m-kali-n (atau m x n).

$$[A] = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{bmatrix} \tag{2.4}$$

Matrik-matrik dengan ukuran m – 1, seperti yang diperlihatkan pada gambar 2.5 berikut ini :

$$[B] = [\, b_1 \;\; b_2 \;\; b_3 \dots b_n \,] \tag{2.5}$$

disebut *vektor baris*. Perhatikan bahwa untuk kesederhanaan, indeks pertama dari masing-masing elemen dibuang. Kadang-kadang diperlukan pula notasi tulisan

cepat untuk membedakan vektor baris dengan matrik jenis lain, yaitu dengan membuka kurung seperti $\lfloor B \rfloor$.

Matrik-matrik dengan ukuran kolom n 1, seperti yang diperlihatkan pada gambar 2.6 berikut ini :

$$[A] = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix} \tag{2.6}$$

disebut vektor kolom. Untuk kesederhanaan, indeks kedua dari masing-masing elemen dibuang. Sebagaimana dengan vektor baris, kadang-kadang diperlukan notasi tulisan cepat untuk membedakan vektor kolom dengan matrik jenis lain. Salah satu cara ialah dengan membuat kurung seperti $\{B\}$.

Jadi, dapat disimpulkan bahwa sebuah vektor merupakan sebuah matrik, tetapi tidak untuk sebaliknya. Dengan kata lain, sebuah vektor merupakan matrik yang berdimensi satu. Sedangkan untuk matrik sendiri mempunyai dua dimensi dalam menuliskan elemen-elemennya, yang terdiri dari baris dan kolom.

Matrik-matrik dengan m − n disebut matrik bujur sangkar. Misalnya matrik 4 x 4 adalah :

$$[A] = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \tag{2.7}$$

Diagonal terdiri atas elemen-elemen a_{11}, a_{22}, a_{33}, dan a_{11} diberi istilah *diagonal utama* matrik.

Terdapat sejumlah bentuk khas matrik-matrik bujursangkar yang perlu diperhatikan, antara lain sebagai berikut :

1. *Matrik Simetri* adalah matrik dimana jumlah baris sama dengan jumlah kolomnya. Misalnya :

$$[A] = \begin{bmatrix} 2 & 1 & 3 \\ 4 & 5 & 1 \\ 7 & 3 & 1 \end{bmatrix} \qquad (2.7a)$$

Merupakan matriks simetri 3 x 3.

2. *Matrik Diagonal* adalah matrik bujursangkar dimana semua elemen bukan diagonal sama dengan nol. Misalnya :

$$[A] = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (2.7b)$$

3. *Matrik Satuan* adalah matrik diagonal dimana semua elemen pada diagonal sama dengan satu. Misalnya :

$$[I] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (2.7c)$$

Lambang **[I]** digunakan untuk menyatakan matrik satuan

4. *Matrik Pita* mempunyai semua elemen sama dengan nol, dengan pengecualian pada suatu pita yang berpusat pada diagonal utama :

$$[A]=\begin{bmatrix} 2 & 4 & 0 & 0 \\ 3 & 2 & 2 & 0 \\ 0 & 5 & 3 & 1 \\ 0 & 0 & 3 & 1 \end{bmatrix} \qquad (2.7d)$$

Matrik di atas mempunyai lebar pita 3 dan diberi nama khas, yaitu *matrik tridiagonal*.

2.3. Nilai eigen dan Vektor eigen

Misalkan $\Lambda = (a_{jk})$ adalah suatu matrik n x n dan X suatu vektor kolom. Persamaan

$$AX = \lambda X$$

di mana λ suatu bilangan dapat ditulis sebagai

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \lambda \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \qquad (2.8)$$

atau

$$\begin{array}{ccccccccccc} (a_{11}-\lambda)x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & 0 & & \\ a_{21}x1 & + & (a_{22}-\lambda)x_2 & + & \cdots & + & a_{2n}x_n & = & 0 & & \\ \cdots & & \cdots & & \cdots & & \cdots & \ddots & \vdots & & (2.9) \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \cdots & + & (a_{nn}-\lambda)x_n & = & 0 & & \end{array}$$

Persamaan (2.9) akan memiliki penyelesaian tak-trivial jika dan hanya jika

$$\begin{vmatrix} a_{11}-\lambda & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22}-\lambda & \cdots & a_{2n} \\ \cdots & \cdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn}-\lambda \end{vmatrix} = 0 \qquad (2.10)$$

yang merupakan suatu persamaan suku banyak berderajat n dalam λ. Akar dari persamaan suku banyak ini dinamakan nilai eigen atau nilai karakteristik (nilai ciri) dari matrik A. bersesuaian dengan setiap nilai eigen akan ada penyelesaian $X \neq 0$ yang merupakan suatu penyelesaian tak-trivial, yang dinamakan suatu vektor eigen atau vektor karakteristik dari nilai eigennya. Persamaan (2.10) juga dapat ditulis

$$\det(A-\lambda I)=0 \qquad (2.11)$$

dan persamaan dalam λ ini seringkali dinamakan persamaan karakteristik.

2.4. Metode Penyelesaian Persoalan Nilai eigen dan Vektor eigen

Dalam penyelesaian sistem persamaan homogen, ditemui persoalan nilai eigen dan vektor eigen. Dalam penyelesaian persoalan nilai eigen dan vektor eigen, sebenarnya banyak metode yang dapat digunakan untuk menyelesaikannya. Tetapi dalam hal ini, hanya ada 4 metode saja yang akan dianalisis, yaitu : metode Fadeev-Leverrier, metode Bairstow, metode Power, dan metode Deflation.

Untuk menyelesaikan persoalan nilai eigen dan vektor eigen, menggunakan kombinasi dari ke empat metode di atas, yaitu metode Fadeev-Leverrier dengan metode Bairstow dan metode Power dengan metode Deflation.

2.4.1. Metode Fadeev-Leverrier

Metode Fadeev-Leverrier adalah pendekatan yang efisien untuk mendapatkan koefisien p_i dari polinomial

$$(-1)^n(\lambda^n - p_{n-1}\lambda^{n-1} - \cdots - p_1\lambda - p_0) = 0 \qquad (2.12)$$

yang merupakan hasil dari pengembangan pada determinan sistem

$$[[A] - \lambda[I]]\{X\} = 0 \qquad (2.13)$$

Untuk mendapatkan koefisien p_i, metode ini menghasilkan urutan dari matrik [B] yang bisa digunakan untuk menentukan nilai p_i. Untuk jelasnya,langkah-langkah yang digunakan dapat dilihat sebagai berikut :

$$[B]_{n-1} = [A] \qquad (2.14)$$

dan

$$P_{n-1} = tr[B]_{n-1} \qquad (2.15)$$

dimana $tr[B]_{n-1}$ adalah trace dari matrik $[B]_{n-1}$ yaitu merupakan jumlah dari koefisien diagonal matrik $[B]_{n-1}$.

Kemudian diteruskan dengan menghasilkan matrik

$$[B]_{n-2} = [A][\,[B]_{n-1} - p_{n-1}[I]\,] \qquad (2.16)$$

yang bisa digunakan untuk menghitung

$$P_{n-2} = \frac{1}{2}\mathrm{tr}\,[B]_{n-2} \qquad (2.17)$$

dan

$$[B]_{n-3} = [A][\ [B]_{n-2} - p_{n-2}[I]\ ] \tag{2.18}$$

yang bisa digunakan untuk menghitung

$$P_{n-3} = \frac{1}{3}\,\mathrm{tr}\,[B]_{n-3} \tag{2.19}$$

dan seterusnya hingga p_0 ditentukan dari

$$[B]_0 = [A][\ [B]_1 - p_1[I]\ ] \tag{2.20}$$

dan

$$P_0 = \frac{1}{\mathrm{n}}\,\mathrm{tr}\,[B]_0 \tag{2.21}$$

2.4.2. Metode Bairstow

Setelah menerapkan metode Fadeev-Leverrier, akar-akar dari persamaan polinomial yang terbentuk harus dihitung untuk mendapatkan nilai eigen. Oleh karena itulah, digunakan metode Bairstow sebagai salah satu penyelesaian.

Sebagai permulaan, secara umum polinomial dapat ditulis dalam bentuk

$$f_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \tag{2.22}$$

Walaupun ini adalah bentuk yang umum, ada bentuk lain untuk memahami polinomial. Sebagai contoh, polinomial berderajat lima dapat ditulis dalam bentuk

$$f_5(x) = (x+1)(x-4)(x-5)(x+7)(x-2) \tag{2.23}$$

jika bentuk kedua ini dikalikan, maka polinomial dengan bentuk seperti (2.22) akan diperoleh. Tetapi, bentuk (2.23) mempunyai kelebihan, yaitu dengan jelas menunjukkan akar-akar fungsi tersebut. Maka dapat ditunjukkan bahwa x = -1, 4, 5, -7 dan 2 merupakan akar-akar karena masing-masing menyebabkan persamaan (2.23) menjadi nol.

Sekarang seandainya persamaan (2.23) di atas dibagi dengan salah satu faktornya, sebagai contoh x + 7, maka akan menghasilkan polinomial berderajat empat dengan sisa nol. Sebaliknya, jika persamaan (2.23) dibagi dengan suatu faktor (contohnya, x + 6), maka akan menghasilkan polinomial berderajat empat juga, tetapi tentu saja dengan sisa tidak sama dengan nol.

Dengan dasar seperti di atas, dapat dibuat suatu algoritma untuk mencari akar-akar dari suatu polinomial :

1. Tentukan suatu nilai awal untuk nilai akar, x t.
2. Bagi polinomial tersebut dengan faktor x – t.
3. Hitung apakah ada sisa. Jika tidak, maka nilai akar ditemukan sama dengan t. Jika sisa tidak sama dengan nol, maka nilai awal harus diubah secara sistematis dan prosedur diulang hingga tidak ada lagi sisa dan akar telah ditemukan.
4. Setelah langkah di atas selesai, keseluruhan prosedur diulang hingga seluruh akar-akar yang tersisa ditemukan.

Metode Bairstow secara umum menggunakan pendekatan seperti di atas. Dapat disimpulkan, bahwa metode Bairstow berdasarkan proses matematika yaitu pembagian suatu polinomial dengan suatu faktor.

Untuk mengijinkan perhitungan yang melibatkan akar-akar imajiner, metode Bairstow membagi suatu polinomial dengan suatu faktor kuadrat, $x2$-rx-s.

Jika ini pembagian ini dilakukan pada persamaan (2.22), hasilnya adalah suatu polinomial baru

$$f_{n-2}(x) = b_n x^{n-2} + b_{n-1} x^{n-3} + \cdots + b2$$

dengan sisa

$$R = b_1(x - r) - b_0 \qquad (2.24)$$

dan kemudian untuk mendapatkan nilai dari b dapat dilihat, sebagai berikut :

$$b_n = a_n$$

$$b_{n-1} = a_{n-1} + rb_n$$

$$b_i = a_i + rb_{i+1} + sb_{i+2} \quad \text{untuk i = n - 2 sampai 0}$$

Jika $x^2 - rx - s$ adalah pembagi yang tepat dari suatu polinomial, maka akar-akar imajiner bisa ditentukan dengan rumus kuadrat. Dengan begitu, metode ini dapat disederhanakan untuk menentukan nilai-nilai dari r dan s yang

menghasilkan faktor kuadrat sebagai pembagi yang tepat. Dengan kata lain, nilai-nilai yang menghasilkan nilai sisa sama dengan nol.

Berdasarkan persamaan (2.24), dapat disimpulkan bahwa agar nilai sisa R bernilai nol, maka b_0 dan b_1 haruslah bernilai nol juga. Tetapi belum tentu nilai r dan s yang telah ditentukan akan menghasilkan nilai nol untuk b_0 dan b_1, maka diperlukan suatu cara yang sistematis untuk merubah nilai r dan s sehingga nilai b_0 dan b_1 mendekati nol. Untuk itu dapat dilihat langkah-langkah berikut :

$$c_n = b_n$$

$$c_{n-1} = b_{n-1} + rc_n$$

$$c_i = b_i + rc_{i+1} + sc_{i+2} \quad \textit{untuk } i = n-2 \text{ sampai } 1$$

kemudian

$$\begin{aligned} c_2\Delta r + c_3\Delta s &= -b_1 \\ c_1\Delta r + c_2\Delta s &= -b_0 \end{aligned}$$

Persamaan di atas diselesaikan untuk Δr dan Δs, dimana digunakan untuk memperbaiki nilai awal r dan s.

Pada setiap langkah, suatu tingkat kesalahan untuk r dan s bisa diperkirakan dengan

$$\left|\varepsilon_{ar}\right| = \left|\frac{\Delta r}{r}\right| \times 100\%$$

dan

$$|\varepsilon_{as}| = \left|\frac{\Delta s}{s}\right| \times 100\%$$

Pada saat kedua nilai kesalahan ini ada di bawah nilai pembanding yang telah ditentukan ε_s maka nilai-nilai dari akar-akar bisa diperhitungkan dengan

$$x = \frac{r \pm \sqrt{r^2 + 4s}}{2} \tag{2.25}$$

Pada titik ini, ada tiga kemungkinan :

1. Persamaan polinomial yang tersisa adalah polinomial berderajat tiga atau lebih. Untuk kasus ini, metode Bairstow diterapkan kembali ke persamaan untuk mendapatkan nilai baru untuk r dan s. Nilai r dan s sebelumnya bisa digunakan sebagai nilai awal untuk penerapan ini.
2. Persamaan yang tersisa adalah persamaan kuadrat. Untuk kasus ini, dua akar yang tersisa langsung dihitung dengan persamaan (2.25).
3. Persamaan yang tersisa adalah polinomial berderajat satu. Untuk kasus ini, satu akar yang tersisa dapat dihitung dengan

$$x = -\frac{s}{r}$$

2.4.3. Metode Power

Metode umum untuk mendapatkan nilai eigen dari matrik A adalah dengan mencari akar-akar λ dari

$$|A - \lambda I| = 0$$

tetapi hal ini sangatlah menyulitkan jika matrik A berukuran besar. Bahkan untuk menghitung determinan dari n x n matrik adalah pekerjaan yang sangat besar bila n bernilai besar, dan di atas semua itu harus diselesaikan pula persamaan polinomial berderajat n untuk λ.

Metode Power dalam hal ini, memberi jalan langsung untuk menghitung satu nilai eigen λ yaitu nilai eigen yang terbesar, dan vektor eigen yang berhubungan dengan nilai eigen tersebut. Metode ini adalah metode iterasi, dimana dimulai dengan nilai tebakkan awal x_0, dan menghasilkan urutan dari pendekatan x_k.

Algoritma untuk metode ini sangatlah sederhana, yaitu sebagai berikut :

1. Ambil suatu nilai untuk x_0 sebagai tebakkan awal.
2. Do k = 1 to m
   a. Tentukan $y_k = Ax_{k-1}$
   b. Tentukan α_k = nilai terbesar dari elemen y_k
   c. Tentukan $x_k = y_k / \alpha_k$
3. End do.

Perulangan akan dilakukan sebanyak m kali. Pada setiap langkah, x_{k-1} dikalikan sebelumnya dengan A. Nilai untuk x_k didapatkan dari pembagian y_k

dengan nilai absolut elemen terbesar dari y_k yang dilambangkan dengan α_k. Jika metode ini dijalankan, maka x_k adalah vektor eigen dan α_k adalah nilai eigen.

2.4.4. Metode Deflation

Pada penggunaan metode Power di atas, hanya menghasilkan nilai eigen yang terbesar. Untuk itulah digunakan metode Deflation untuk melengkapi metode Power sehingga dapat dicari nilai-nilai eigen yang lain. Metode Deflation ini hanya dapat digunakan untuk matrik simetris.

Anggap bahwa metode Power telah diterapkan pada matrik A untuk mendapatkan nilai eigen λ_1 dan vektor eigen e_1, kemudian anggap

$$A_1 = A - \lambda_1 \frac{e_1 e_1^t}{e_1^t e_1}$$

untuk setiap vektor eigen e_i dari A, A_1 akan memenuhi

$$A_1 e_i = A e_i - \lambda_1 \left(\frac{e_1^t e_i}{e_1^t e_1} \right) e_1$$

pada i = 1

$$A_1 e_1 = A e_1 - \lambda_1 e_1 = 0$$

sedangkan untuk i > 1

$$A_1 e_i = A e_i = \lambda_i e_i$$

maka nilai eigen dari matrik A_1 adalah $\lambda_2, \ldots, \lambda_n$, 0, dan metode Power akan menghasilkan nilai eigen terbesar kedua λ_2.

Maka dapat dilihat bahwa metode ini bisa diteruskan dengan menentukan

$$A_2 = A_1 - \lambda_2 \frac{e_2 e_2^T}{e_2^T e_2}, \cdots \; A_i = A_{i-1} - \lambda_i \frac{e_i e_i^T}{e_i^T e_i}$$

Pada setiap langkah, nilai eigen terbesar λ_i diubah menjadi nol, dan bila metode Power diterapkan pada matrik A_i akan dihasilkan nilai eigen terbesar yang tersisa λ_{i+1} dan seterusnya hingga seluruh nilai eigen ditemukan.

2.5. Analisis Algoritma dan Kompleksitas

2.5.1 Analisis Algoritma

Algoritma dapat diartikan sebagai himpunan berhingga langkah-langkah untuk menyelesaikan masalah dengan menggunakan komputer. Jadi dapat diartikan bahwa analisis algoritma adalah suatu cara untuk memeriksa dengan teliti langkah-langkah dalam menyelesaikan masalah dengan bantuan komputer. Untuk menyelesaikan masalah dengan menggunakan komputer, haruslah memenuhi kriteria sebagai berikut :

1. Setiap langkahnya harus pasti (definite).
2. Mempunyai minimal satu ouput (input boleh tidak ada).
3. Adanya kriteria berhenti (Stopping Criteria).

Dalam menganalisa suatu algoritma, dibutuhkan suatu takaran untuk mengukur efesiensi dari algoritma itu sendiri. Untuk itu ada beberapa faktor yang perlu diperhatikan, diantaranya :

1. Waktu komputasi / proses (running).
2. Implementasi dalam pembuatan program.
3. Kebutuhan memori.
4. Jumlah prosesor yang digunakan.

Terdapat dua tipe analisis algoritma, yaitu : [BAM00]

1. Analisis untuk memeriksa kebenaran algoritma.

   Hal ini dilakukan dengan menelusuri algoritma, penelusuran logika, implementasi algoritma, mengujinya dengan data, atau menggunakan teknik matematika untuk membuktikan kebenaran.

2. Analisis terhadap efisiensi algoritma

   Hal ini dilakukan dengan perhitungan kompleksitas komputasi (waktu).

2.5.2 Kompleksitas

Kompleksitas dari sebuah komputasi dapat diartikan kebutuhan dalam mengukur waktu dari komputasi itu sendiri. Kompleksitas dari proses penyelesaian linier adalah mengukur kebutuhan waktu yang diperlukan dalam melakukan proses perhitungan.

Program yang baik adalah program yang efisien dalam penggunaan sumber daya waktu. Pengukuran waktu suatu program sangat kompleks, karena waktu yang sebenarnya dibutuhkan oleh program sangat ditentukan oleh komputer serta kompilator yang digunakan. Secara teoritis, model perhitungan waktu harus independen dari pertimbangan mesin dan kompilator.

Penentuan jumlah waktu yang diperlukan algoritma secara eksak disebabkan oleh : [BAM00]

1. Kesulitan pertama adalah jumlah waktu secara eksak bergantung implementasi dari algoritma dan mesin yang menjalankan. Untuk itu diperlukan metode yang lebih umum yang tidak tergantung bahasa dan mesin yang menjalankan program.
2. Jumlah waktu yang diperlukan bergantung jumlah masukkan. Misalnya, algoritma penjumlahan nilai-nilai pada *array* berukuran 100.000 jelas diharapkan memerlukan waktu yang lebih banyak dibanding *array* berukuran 100. Umumnya perkiraan waktu yang diperlukan algoritma dinyatakan sebagai fungsi dari ukuran masukkan.

Notasi **big-O** adalah menghilangkan semua konstanta dan faktor kecuali yang mendominasi dari suatu fungsi langkah {g(n)}. Contoh :

Pada array dengan n = 100

Big-O operasi pada *array* adalah sebagai berikut :

1. Pencarian nilai terbesar (maksimum) mempunyai **O(n)**.
2. Pengambilan data dari elemen tertentu yang diketahui posisinya mempunyai **O(1).**

Pernyataan itu bukan berarti untuk menemukan nilai tebesar memerlukan 100 kali dibanding untuk menemukan satu elemen tunggal. **Big-O** hanya menyatakan bahwa menemukan nilai terbesar sebanding jumlah elemen array, tapi

mengambil nilai tertentu akan memerlukan waktu konstan dan tidak bergantung pada jumlah elemen *array* [BAM00].

2.6. Pemrograman Delphi 6.0

Borland Delphi 6.0 yang untuk selanjutnya disingkat dengan Delphi merupakan program aplikasi database yang berbasis *Object Pascal* dari Borland.

Kelebihan Delphi dibandingkan dengan program aplikasi lainnya adalah faktor produktivitas. Dengan memakai Delphi adalah cara yang paling sederhana untuk membangun aplikasi berbasis *Windows*. Produktivitas dari pengembangan perangkat lunak yang dimaksud dibagi menjadi lima atribut penting, yaitu : [ING00]

1. Kualitas dari lingkungan pengembangan visual.
2. Kecepatan compiler dibandingkan dengan kompleksitasnya.
3. Kekuatan dari bahasa pemrograman dibandingkan dengan kompleksitasnya.
4. Fleksibilitas dari arsitektur basis data
5. Pola desain dan pemakaian yang diwujudkan oleh kerangka kerjanya.

Alasan lain mengapa memilih bahasa pemrograman Borland Delphi 6.0 adalah bahasa pemrograman yang dirancang untuk bekerja di dalam *platform Microsoft Windows*. Selain itu penggunaan bahasa *Object Pascal* yang berarti merupakan bahasa pemrograman terstruktur, sehingga memudahkan dalam *coding* dan *debugging*.