BAB II

TEORI PENUNJANG

2.1. Algoritma

Istilah algoritma pertama kali diperkenalkan oleh seorang ahli matematika yaitu Abu Ja'far Muhammad Ibnu Musa Al Khawarizmi (Suryadi, MT. 1994). Secara bahasa, algoritma berarti suatu metode khusus untuk menyelesaikan suatu masalah yang nyata.

Definisi 2.1. 1. Algoritma merupakan suatu metode khusus yang tepat dan terdiri dari serangkaian langkah yang terstruktur dan dituliskan secara sistematis yang akan dikerjakan untuk menyelesaikan suatu masalah dengan bantuan komputer. ♦

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

Gambar 2.1. 1. Diagram alir proses penyelesaian masalah

Kriteria algoritma yang baik meliputi :

1. Ada *input*

Suatu algoritma harus mempunyai satu atau lebih *input* merupakan suatu kuantitas yang diberikan sebagai nilai awal sebelum suatu algoritma dimulai.

2. Ada *output*

Suatu algoritma harus mempunyai *output* yang merupakan solusi dari masalah yang sedang diselesaikan.

3. Tertentu *(definiteness)*

Langkah-langkah yang dikerjakan sebuah algoritma harus didefinisikan dengan tepat dan jelas untuk setiap masalah yang sedang diselesaikan.

4. Berhingga *(finiteness)*

Banyaknya langkah atau instruksi harus berhingga agar waktu proses dari suatu algoritma relatif lebih singkat dan diperoleh hasil yang sesuai dengan masalah yang ada.

5. Efektif (*effectiveness*)

Suatu algoritma dikatakan efektif jika algoritma tersebut dapat menghasilkan suatu solusi yang sesuai dengan masalah yang diselesaikan.

Algoritma dapat disajikan dengan beberapa cara yaitu :

- dengan bahasa alamiah yaitu algoritma ditulis dengan ungkapan sehari-hari.
- dengan sandi semu *(Pseudocode)* yaitu algoritma ditulis dengan menggunakan perintah bahasa pemrograman tertentu ditambah dengan bahasa alamiah.
- dengan bagan alir *(flowchart)* yaitu algoritma ditulis dalam bentuk simbol/diagram.

2.2. Matriks dan Sistem Persamaan Linear

2.2. 1. Matriks

Definisi 2.2. 1. Sebuah matriks adalah susunan persegi panjang dari elemen-elemen yang disusun dalam baris dan kolom. ♦

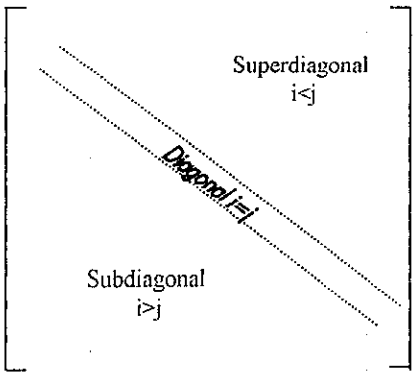
Jika A adalah sebuah matriks maka a_{ij} menyatakan elemen yang terdapat di dalam baris i dan kolom j dari A. Jadi matriks Amxn yang umum dapat ditulis :

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \quad \text{atau } [a_{ij}]_{mxn}$$

Definisi 2.2.2. Dua matriks $A=[a_{ij}]$ dan $B=[b_{ij}]$ disebut sama ($A=B$) jika dan hanya jika keduanya berukuran sama dan setiap elemen yang terletak bersesuaian juga sama yaitu $a_{ij}= b_{ij}$ untuk semua i dan j. ♦

Definisi 2.2.3. Matriks bujur sangkar berukuran n adalah sebuah matriks dengan n baris dan n kolom. ♦

Jika $A=[a_{ij}]$ merupakan matriks bujur sangkar berukuran n maka elemen $a_{11},a_{22},\ldots,a_{nn}$ disebut elemen diagonal dari A. Semua elemen a_{ij} dengan $i<j$ disebut elemen superdiagonal dari A (di atas diagonal) dan semua elemen a_{ij} dengan $i>j$ disebut elemen subdiagonal dari A (di bawah diagonal).



Gambar 2.2. 1. Elemen diagonal, superdiagonal dan subdiagonal dari matriks bujur sangkar

Definisi 2.2.4. Matriks A=[a_{ij}] berukuran n bersifat tridiagonal jika a_{ij}=0 untuk setiap $|i-j|>1$. ♦

Contoh 2.2. 1.

$$A=\begin{bmatrix}3&1&0&0\\1&3&1&0\\0&2&2&1\\0&0&1&6\end{bmatrix} \qquad B=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&1&0\end{bmatrix} \qquad C=\begin{bmatrix}4&7&0&0\\0&8&1&0\\0&1&1&0\\0&0&0&1\end{bmatrix}$$

Definisi 2.2. 5. Jika $A=[a_{ij}]_{mxn}$ dan $B=[b_{ij}]_{nxp}$ maka perkalian A dan B ditulis $C=AB$ adalah matriks $C=[c_{ij}]_{mxp}$ dengan $c_{ij}=\sum_{k=1}^{n} a_{ik}b_{kj}$ untuk $i=1,2,\ldots,m$ dan $j=1,2,\ldots,p$. ♦

2.2.2. Sistem Persamaan Linear

Sebuah garis dalam bidang xy dapat dinyatakan oleh persamaan yang berbentuk $a_1x+a_2y=b$. Persamaan ini dinamakan persamaan linear dalam variabel x dan y. Secara umum persamaan linear dalam n variabel $x_1,x_2,\ldots,x_n$ didefinisikan sebagai persamaan yang dapat dinyatakan dengan :

$$a_1x_1 + a_2x_2 +\ldots+ a_nx_n = b \qquad (2.2.\ 1)$$

dengan $a_1,a_2,\ \ldots,a_n$ dan b merupakan konstanta riil.

Contoh 2.2. 2. Berikut diberikan contoh persamaaan linear :

1. $x+3y=7$
2. $y=1/2x+3z+1$. □

Definisi 2.2.6. Sistem persamaan linear adalah sebuah himpunan berhingga dari persamaan-persamaan linear dalam beberapa variabel. ♦

Sebuah sistem m persamaan linear dengan n variabel $x_1,x_2,\ldots,x_n$ dapat ditulis sebagai :

$$\left.\begin{array}{rcl} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &=& b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &=& b_2 \\ \vdots \quad\quad \vdots \quad\quad\quad\quad \vdots &\quad& \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &=& b_m \end{array}\right\} \qquad (2.2.2)$$

dengan $x_1,x_2,\ldots,x_n$ merupakan variabel yang tidak diketahui (*unknown constanta*) sedangkan a dan b menyatakan konstanta. Menurut definisi 2.2.2 maka m persamaan dalam sistem persamaan linear (2.2.2) dapat ditulis dengan persamaan matriks sebagai berikut :

$$\begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \qquad (2.2.3)$$

Matriks dalam (2.2.3) dapat ditulis sebagai matriks $AX=B$.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1(n-1)} & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2(n-1)} & a_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{(m-1)1} & a_{(m-1)2} & \cdots & a_{(m-1)(n-1)} & a_{(m-1)n} \\ a_{m1} & a_{m2} & \cdots & a_{m(n-1)} & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{m-1} \\ b_m \end{bmatrix} \qquad (2.2.4)$$

Contoh 2.2.3. Sistem persamaan

$$x_1 + x_2 + 2x_3 = 9$$

$$2x_1 + 4x_2 - 3x_3 = 1$$

$$3x_1 + 6x_2 - 5x_3 = 0$$

dapat dinyatakan dengan matriks sebagai berikut :

$$\begin{bmatrix} 1 & 1 & 2 \\ 2 & 4 & -3 \\ 3 & 6 & -5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 9 \\ 1 \\ 0 \end{bmatrix} . \quad \square$$

2.2.3. Sistem Tridiagonal

Dari definisi 2.2.4. dijelaskan bahwa matriks tridiagonal adalah matriks yang semua elemen diluar diagonal, subdiagonal dan superdiagonal bernilai nol. Namun dalam subbab ini hanya akan dibahas tentang sistem linear dengan asumsi bahwa elemen diagonal, subdiagonal dan superdiagonal mempunyai nilai ($\neq 0$). Sistem tridiagonal sering dipakai dalam praktek rekayasa dan ilmiah. Sebagai contoh, metode numerik seperti interpolasi spline kubik (Bab III) melibatkan penyelesaian sistem tridiagonal.

Walaupun eliminasi Gauss dapat digunakan untuk memecahkan sistem tridiagonal tetapi metode tersebut tidak efisien karena jika *pivoting* dilakukan, tak satupun elemen-elemen diluar diagonal, subdiagonal dan superdiagonal berubah dari nilai nolnya. Sehingga akan terjadi pemborosan ruang dan waktu pada penyimpanannya. Jika telah diketahui bahwa *pivoting* pada elemen-elemen nol itu tidak ada gunanya maka dapat dikembangkan suatu algoritma yang tidak melibatkan elemen nol diluar elemen diagonal, subdiagnal dan superdiagonal. Algoritma alternatif berikut merupakan metode pilihan .

Sistem tridiagonal dapat dikatakan sebagai matriks dengan lebar pita tiga, secara umum matriks tridiagonal dapat dinyatakan dengan :

$$\begin{bmatrix} b_1 & c_1 & 0 & \cdots & 0 & 0 \\ a_1 & b_2 & c_2 & \cdots & 0 & 0 \\ 0 & a_1 & b_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & \cdots & a_{n-1} & b_n \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ \vdots \\ m_{n-1} \\ m_n \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix} \qquad (2.2.5)$$

Diasumsikan $b_1 \neq 0$, kemudian m_1 dari persamaan pertama dieliminasikan ke persamaan kedua sehingga didapatkan persamaan kedua baru sebagai berikut :

$$b'_2 m_2 + c_2 m_3 = u'_2 \qquad (2.2.6)$$

dengan $b'_2 = b_2 - \frac{a_1}{b_1} c_1$ dan $u'_2 = u_2 - \frac{a_1}{b_1} u_1$.

Selanjutnya mengasumsikan $b'_2 \neq 0$ dan mengeliminasi m_2 dari persamaan (2.2.6) ke persamaan ketiga dalam (2.2.5) sehingga diperoleh persamaan ketiga baru :

$$b'_3 m_3 + c_3 m_4 = u'_3 \qquad (2.2.7)$$

dengan $b'_3 = b_3 - \frac{a_2}{b'_2} c_2$ dan $u'_3 = u_3 - \frac{a_2}{b'_2} u'_2$.

Dengan meneruskan prosedur ini dalam langkah ke-i, m_i dari persamaan ke-i (diasumsi $b'_i \neq 0$) dieliminasikan ke persamaan ke-$i+1$ sehingga diperoleh persamaan ke-i+1 yang baru :

$$b'_{i+1} m_{i+1} + c_{i+1} m_{i+2} = u'_{i+1} \qquad (2.2.8)$$

dengan $b'_{i+1} = b_{i+1} - \frac{a_i}{b'_i} c_i$ dan $u'_{i+1} = u_{i+1} - \frac{a_i}{b'_i} u'_i$ untuk i=1,2,...,n-1.

Langkah selanjutnya, pensubstitusian kembali dengan mengasumsikan bahwa $b'_n \neq 0$ dan $m_n = \frac{u'_n}{b'_n}$, sehingga untuk $i=n-1,n-2,..,1$ diperoleh nilai $m_i = \frac{u'_i - c_i m_{i+1}}{b'_i}$.

Algoritma 2.2.1. menggambarkan metode di atas yang merupakan metode yang efisien untuk sistem tridiagonal (2.2.5) dengan `type larik2=array[0..100] of real;`

```
procedure eliminasi_untuk_sistem_tridiagonal(n:integer;
                                    a,c:larik2; var b,u,m:larik2);
var i:integer;t:real;
begin
for i←2 to n-1 do
 begin
   t:=a[i-1]/b[i-1];
   b[i]:=b[i]-t*c[i-1];
   u[i]:=u[i]-t*u[i-1];
 end;
m[n-1]:=u[n-1]/b[n-1];
for i←n-2 downto 1 do
   m[i]:=(u[i]-c[i]*m[i+1])/b[i];
end;
```

Algoritma 2.2.1. Eliminasi untuk sistem tridiagonal.

2.3. Kontinuitas dan Diferensial

Elemen-elemen dari suatu himpunan adalah segala sesuatu yang membentuk himpunan. Jika A menotasikan himpunan dan a ada dalam A maka dikatakan a adalah elemen A dan dinotasikan dengan $a \in A$. Jika a tidak berada dalam A maka dikatakan a bukan elemen A dan dinotasikan dengan $a \notin A$.

Definisi 2.3. 1. Suatu fungsi f dari himpunan X ke Y adalah suatu aturan yang memetakan setiap $x \in X$ dengan tepat satu $y \in Y$. ♦

Definisi 2.3. 2. Suatu fungsi polinomial berderajat n didefinisikan sebagai $f(x) = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1} + a_n x^n$, $a_n \neq 0$ dengan n adalah bilangan bulat positif dan $a_0, a_1, \ldots, a_n$ adalah konstanta riil. ♦

2.3. 1. Kekontinuan Fungsi

Definisi 2.3. 3. Suatu fungsi $f(x)$ dikatakan mempunyai limit kanan di titik $x=a$ sebesar L atau $\lim_{x \to a^+} f(x) = L$, jika setiap $\varepsilon > 0$ terdapat $\delta > 0$ sedemikian sehingga

$0 < x-a < \delta \Rightarrow |f(x)-L| < \varepsilon$. ♦

Definisi 2.3. 4. Suatu fungsi $f(x)$ dikatakan mempunyai limit kiri di titik $x=a$ sebesar L atau $\lim_{x \to a^-} f(x) = L$, jika setiap $\varepsilon > 0$ terdapat $\delta > 0$ sedemikian sehingga

$\delta < x-a < 0 \Rightarrow |f(x)-L| < \varepsilon$. ♦

Definisi 2.3. 5. Suatu fungsi *f(x)* dikatakan mempunyai limit di satu titik $x=a$ sebesar L atau $\lim_{x\to a} f(x) = L$, jika setiap $\varepsilon > 0$ terdapat $\delta > 0$ sedemikian sehingga $0<|x-a|<\delta \Rightarrow |f(x)-L|<\varepsilon$. ♦

Contoh 2.3. 1. Buktikan bahwa jika $a>0$, $\lim_{x\to a} \sqrt{x} = \sqrt{a}$!.

Analisis pendahuluan. Andaikan diberikan $\varepsilon>0$, akan dicari δ sedemikian sehingga $0<|x-a|<\delta \Rightarrow |\sqrt{x}-\sqrt{a}|<\varepsilon$. Pandang pertidaksamaan sebelah kanan.

$$|\sqrt{x}-\sqrt{a}|<\varepsilon \Leftrightarrow \left|\frac{(\sqrt{x}-\sqrt{a})(\sqrt{x}+\sqrt{a})}{(\sqrt{x}+\sqrt{a})}\right|<\varepsilon$$

$$\Leftrightarrow \left|\frac{x-a}{(\sqrt{x}+\sqrt{a})}\right|<\varepsilon$$

$$\Leftrightarrow \frac{|x-a|}{\sqrt{x}+\sqrt{a}} \le \frac{|x-a|}{\sqrt{a}} < \varepsilon$$

$$\Leftrightarrow |x-a| < \varepsilon\sqrt{a}$$

Sehingga didapatkan $\delta \le \varepsilon\sqrt{a}$.

Analisis formal. Andai diberikan $\varepsilon>0$, pilih $\delta = \varepsilon\sqrt{a}$ .Maka $0<|x-a|<\delta$ menunjukkan

$$|\sqrt{x}-\sqrt{a}| = \left|\frac{(\sqrt{x}-\sqrt{a})(\sqrt{x}+\sqrt{a})}{(\sqrt{x}+\sqrt{a})}\right| = \left|\frac{x-a}{(\sqrt{x}+\sqrt{a})}\right| = \frac{|x-a|}{\sqrt{x}+\sqrt{a}} < \frac{|x-a|}{\sqrt{a}}$$

$$< \frac{\delta}{\sqrt{a}} = \varepsilon$$

Nilai ε akan terdefinisi jika nilai $a>0$. Sehingga didapatkan $|\sqrt{x}-\sqrt{a}|<\varepsilon$. □

Definisi 2.3. 6. Suatu fungsi *f(x)* dikatakan kontinu di titik $x=a$ jika :

1). *f(a)* ada

2). $\lim_{x\to a} f(x)$ ada

3). $\lim_{x\to a} f(x) = f(a)$. ♦

Definisi 2.3. 7. Suatu fungsi $y=f(x)$ dikatakan kontinu pada interval tertutup $[a,b]$ jika *f(x)* kontinu di setiap titik pada interval tersebut. ♦

Contoh 2.3. 2. Fungsi $f(x) = \sqrt{x}$ kontinu pada interval [1,5].

Penyelesaian :

Untuk kekontinuan fungsi pada titik $x=a$ maka sesuai definisi 2.3.6 akan ditunjukkan bahwa *f(a)* ada, $\lim_{x\to a} f(x)$ ada dan $\lim_{x\to a} f(x) = f(a)$ sebagai berikut :

1). $f(a) = \sqrt{a}$ untuk semua $a\in[1,5]$

2). $\lim_{x\to a} f(x)$ ada. Dalam contoh 2.3.1 diketahui bahwa $\lim_{x\to a} \sqrt{x} = \sqrt{a}$ untuk $a>0$.

3). $\lim_{x\to a} \sqrt{x} = \sqrt{a}$.

Dari 1,2 dan 3 didapatkan bahwa *f(x)* kontinu di setiap titik $x=a$ untuk $a>0$ sehingga *f(x)* kontinu di setiap titik $x\in[1,5]$. Sesuai definisi 2.3.6. maka fungsi $f(x) = \sqrt{x}$ kontinu pada interval [1,5]. □

2.3. 2. Diferensial

Salah satu sifat dari fungsi diferensial (fungsi yang memiliki turunan) adalah semua fungsi merupakan fungsi yang kontinu. Jika sebuah kurva mempunyai sebuah garis singgung di sebuah titik maka kurva itu tidak dapat melompat atau sangat berayun di titik tersebut.

Definisi 2.3. 8. Jika fungsi $f(x)$ didefinisikan dalam interval (a,b) dan $c \in (a,b)$ maka $f(x)$ dikatakan terdiferensial di titik $x=c$ jika nilai dari $\lim_{x \to c} \frac{f(x)-f(c)}{x-c}$ ada dan dinotasikan dengan $f'(c) = \lim_{x \to c} \frac{f(x)-f(c)}{x-c}$. ♦

Operasi pendiferensialan dari suatu fungsi $f(x)$ menghasilkan sebuah fungsi baru $f'(x)$. Jika $f'(x)$ didiferensialkan masih menghasilkan fungsi baru lagi maka fungsi baru dinyatakan dengan $f''(x)$ dan disebut deferensial kedua dari fungsi $f(x)$ dan seterusnya.

Teorema 2.3. 1. Jika suatu fungsi $f(x)$ terdeferensial di titik $x=c$ maka $f(x)$ kontinu di titik x=c. ♦

Bukti : Dari definisi 2.3.8 diketahui bahwa $f'(c) = \lim_{x \to c} \frac{f(x)-f(c)}{x-c}$. Akan diperlihatkan bahwa $\lim_{x \to c} f(x) = f(c)$.

Secara formal dipergunakan kenyataan bahwa limit dari suatu hasil kali fungsi adalah hasil kali dari limit-limitnya untuk memperlihatkan bahwa :

$$\lim_{x\to c}[f(x)-f(c)] = \lim_{x\to c}\left[(x-c)\frac{f(x)-f(c)}{x-c}\right]$$

$$= \lim_{x\to c}(x-c)\lim_{x\to c}\frac{f(x)-f(c)}{x-c}$$

$$= 0.f'(c)$$

$$= 0$$

Persamaan $\lim_{x\to c}[f(x)-f(c)] = 0$ mengakibatkan $\lim_{x\to c} f(x) = f(c)$. □

Contoh 2.3. 1. Fungsi $y=x^2$ adalah kontinu karena diferensiabel. □

Meskipun diferensiabilitas mengakibatkan kontinuitas tapi kebalikannya tidak boleh. Jika fungsi *f(x)* kontinu di *c* maka tidak berarti *f* mempunyai turunan di *c*.

Contoh 2.3. 2. Fungsi $y=|x|$ kontinu di $x=0$ meskipun tidak memiliki turunan di $x=0$.

2.4. Interpolasi

Interpolasi merupakan salah satu metode pencocokan kurva yang digunakan untuk menaksir (mengestimasi) nilai antara (*intermediate values*) di antara titik-titik data yang diketahui. Interpolasi polinom terdiri atas penentuan polinom orde ke-*n* yang cocok dengan *n*+1 titik data. Pada umumnya untuk *n*+1 titik data diinterpolasi oleh satu polinom orde *n* atau kurang yang melalui semua titik. Misalnya terdapat satu garis lurus (suatu polinom orde pertama) yang menghubungkan dua titik dan satu parabola yang menghubungkan tiga titik.

Banyak interpolasi polinom yang mengungkapkan tentang masalah pencocokan kurva ini. Antara lain interpolasi beda–terbagi newton, interpolasi lagrange dan interpolasi spline.

Dalam subbab ini hanya akan dibahas tentang interpolasi lagrange dan interpolasi spline linear sebagai konsep dasar dari interpolasi spline kubik.

2.4. 1. Interpolasi Lagrange

Bentuk umum interpolasi lagrange adalah

$$y = f_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i) \qquad (2.4.\ 1)$$

dengan $L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{(x - x_j)}{(x_i - x_j)}$ dan n adalah derajat dari persamaan lagrange.

Sebagai contoh, bentuk interpolasi lagrange versi linear (n=1) adalah :

$$f_1(x) = \frac{(x - x_1)}{(x_0 - x_1)} f(x_0) + \frac{(x - x_0)}{(x_1 - x_0)} f(x_1) \qquad (2.4.\ 2)$$

dan versi orde kedua (n=2) adalah :

$$f_2(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} f(x_0) + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} f(x_1) + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} f(x_2) \qquad (2.4.\ 3)$$

2.4. 2. Interpolasi Spline

Definisi 2.4. 1. Misalnya $[a,b]$ adalah selang berhingga yang terdiri dari titik $a = x_0 < x_1 < \ldots < x_n = b$. Fungsi spline berderajat m adalah suatu polinom sepotong-sepotong $S(x)$ yang memenuhi sifat-sifat :

1. $S(x)$ merupakan polinom berderajat m pada masing-masing subselang $[x_i, x_{i+1}]$
2. $S(x)$ terdiferensial sebanyak m-1 kali pada tiap titik x_i, $0 \leq i \leq n$. □

Interpolasi spline merupakan interpolasi polinom sepotong-sepotong (*piecewise polynomial interpolation*), ini berarti bahwa untuk suatu fungsi *f(x)* tertentu pada selang $a=x_0<x_1<...<x_n=b$ dapat dihampiri dengan sebuah polinom *S(x)* di setiap subselang $[x_i,x_{i+1}]$ untuk $i=1,2,\ldots,n-1$. Selanjutnya polinom-polinom *S(x)* ini disebut sebagai polinom spline.

Seperti halnya pada interpolasi polinomial, jika pada interpolasi spline diberikan kumpulan titik data (x_i, y_i), $0 \le i \le n$ maka $S(x_i) = y_i$. Seandainya diketahui $n+1$ buah titik data dengan interpolasi spline didapatkan n buah polinom spline atau satu polinom di setiap subselang.

Pada interpolasi spline setiap polinom *S(x)* terdiferensial beberapa kali pada titik ujung subselang (simpul), sehingga diperoleh suatu fungsi polinom *S(x)* yang kontinu di setiap titik data, terutama pada simpulnya. Interpolasi spline ada tiga macam yaitu spline linier, spline kuadratik dan spline kubik.

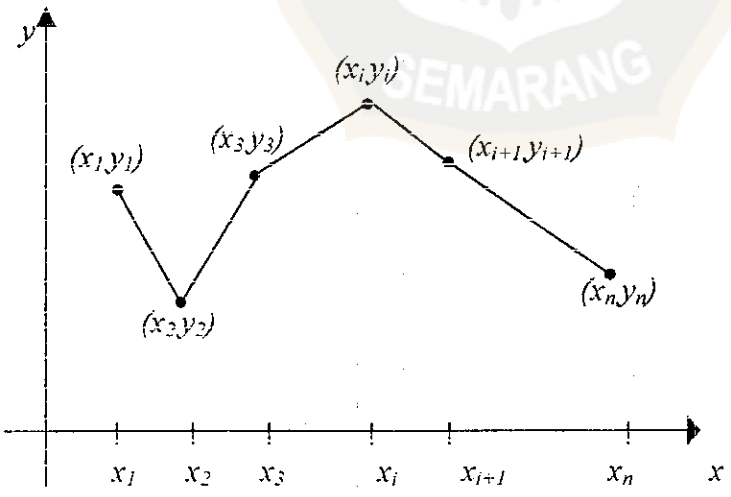
Pada interpolasi spline linier, setiap titik data dalam subselang dihubungkan oleh kurva linier. Kekurangan spline linier adalah ketidakmulusannya, titik tempat dua spline bertemu (disebut simpul) kemiringannya berubah secara mendadak. Pada interpolasi spline kuadratik, setiap titik data dalam subselang dihubungkan oleh kurva kuadratik. Kelemahan spline kuadrat adalah kurva yang dihasilkan terlihat kasar. Hal ini karena spline kuadrat tidak menjamin adanya turunan kedua yang sama di tiap simpulnya. Spline kubik yang dibahas dalam bab III menjamin turunan pertama dan kedua yang kontinu di tiap simpul.

2.4. 3. Interpolasi Spline Linear

Interpolasi spline linear merupakan interpolasi spline yang paling sederhana yaitu menggunakan polinom orde pertama. Interpolasi spline linear akan menghasilkan suatu kurva linear yang melalui titik-titik data. Polinom lagrange versi linear dalam subbab 2.4.1. digunakan untuk menurunkan kurva linear sepotong-sepotong :

$$S_i(x) = y_i \frac{x - x_{i+1}}{x_i - x_{i+1}} + y_{i+1} \frac{x - x_i}{x_{i+1} - x_i}, \quad 0 \le i \le n-1 \qquad (2.4.4)$$

Bentuk persamaan 2.4.4 menghasilkan suatu kurva yang berbentuk garis (gambar 2.4.1) atau dapat juga dinyatakan dengan menggunakan rumus slope untuk garis yaitu $S_i(x) = y_i + d_i(x - x_i)$ dengan $d_i = \frac{(y_{i+1} - y_i)}{(x_{i+1} - x_i)}$, $0 \le i \le n-1$.



Gambar 2.4. 1. Interpolasi spline linear

Polinom spline linear dalam selang $[x_0,x_n]$ dapat ditulis dalam bentuk :

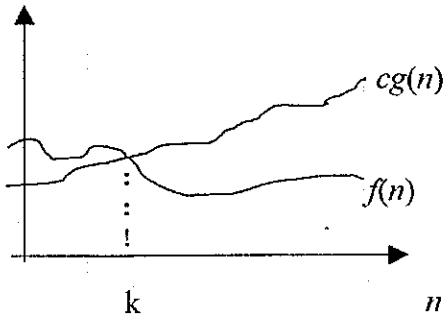
$$S(x)=\begin{cases} y_0+d_0(x-x_0) & x_0 \le x \le x_1 \\ y_1+d_1(x-x_1) & x_1 \le x \le x_2 \\ \vdots & \vdots \\ y_{n-1}+d_{n-1}(x-x_{n-1}) & x_{n-1} \le x \le x_n \end{cases}$$

Diasumsikan $x_0<x_1<\ldots< x_n$. Untuk suatu nilai x yang terdapat dalam interval $[x_i,x_{i-1}]$ maka nilai $y(x)$ dapat dihitung yaitu dengan menghitung selisih $(x-x_i)$ dan nilai polinom spline $S(x)$ adalah :

$$S(x) = S_i(x) = y_i + d_i(x - x_i) \text{ untuk } x_i \le x \le x_{i+1} \qquad (2.4.\ 1)$$

2.5. Notasi "*Big Oh*"

Definisi 2.5. 1. Diberikan S adalah himpunan semua fungsi dengan domain D di mana $D= N$, Z atau R. Diberikan $f, g \in S$. Dikatakan $f(n)=O(g(n))$ (baca $f(n)$ adalah "*Big Oh*" dari $g(n)$) jika terdapat dua buah bilangan positif c dan k sedemikian sehingga $|f(n)| \le c\,|g(n)|$ untuk setiap $n \in D$ dengan $n \ge k$. ♦



Gambar 2.5. 1. Fungsi $f(n)=O(g(n))$

Cara lain untuk menunjukkan bahwa $f(n)=O(g(n))$ adalah :

1. Jika $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$, untuk suatu $c \in R^*$ maka $f(n)=O(g(n))$, jika $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$ maka $f(n)\neq O(g(n))$.

2. Jika $\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) = \infty$ atau jika $\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) = 0$ dan

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)} = \lim_{n\to\infty} \frac{f''(n)}{g''(n)} = \ldots = \lim_{n\to\infty} \frac{f^{(m-1)}(n)}{g^{(m-1)}(n)} = \frac{\infty}{\infty}$$ atau

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)} = \lim_{n\to\infty} \frac{f''(n)}{g''(n)} = \ldots = \lim_{n\to\infty} \frac{f^{(m-1)}(n)}{g^{(m-1)}(n)} = \frac{0}{0}$$ sehingga didapatkan $\lim_{n\to\infty} \frac{f^{(m)}(n)}{g^{(m)}(n)} = c$ maka $f(n)=O(g(n))$, tetapi jika didapatkan $\lim_{n\to\infty} \frac{f^{(m)}(n)}{g^{(m)}(n)} = \infty$ maka maka $f(n)\neq O(g(n))$.

Contoh 2.5. 1. Misalkan $f(n)=37n^2+120n+17$ dan $g(n)=n^3/2$. Akan ditunjukkan bahwa $f(n)=O(g(n))$ tetapi $g(n)\ )\neq O(f(n))$.

Bukti :

- Akan ditunjukkan $f(n)= O(g(n))$.

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{37n^2+120n+17}{n^3/2}$$

$$= \lim_{n\to\infty} \left(\frac{74}{n} + \frac{240}{n^2} + \frac{34}{n^3} \right)$$

$$= 0$$

Karena limit dari perbandingan $f(n)$ dan $g(n)$ ada maka $f(n)=O(g(n))$.

- Akan ditunjukkan $g(n) \neq O(f(n))$.

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} = \lim_{n\to\infty} \frac{n^3/2}{37n^2+120n+17}$$

$$= \lim_{n\to\infty} \left(\frac{n^3}{74n^2+240n+34} \right)$$

$$= \infty$$

Karena limit dari perbandingan $f(n)$ dan $g(n)$ sebesar ∞ maka $f(n) \neq O(g(n))$. □

Contoh 2.5. 2. Misalkan $f(n)=n \log n$ dan $g(n)=n^2$. Akan ditunjukkan bahwa $f(n)=O(g(n))$ tetapi $g(n) \neq O(f(n))$.

Bukti :

- Akan ditunjukkan $f(n)=O(g(n))$.

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n \log n}{n^2}$$

$$= \lim_{n\to\infty} \frac{\log n}{n}$$

$$\lim_{n\to\infty} \frac{f'(n)}{g'(n)} = \lim_{n\to\infty} \frac{(\log e)/n}{1}$$

$$= \lim_{n\to\infty} \frac{\log e}{n}$$

$$= 0$$

Karena limit dari perbandingan $f(n)$ dan $g(n)$ ada maka $f(n)=O(g(n))$.

- Akan ditunjukkan $g(n) \neq O(f(n))$.

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} = \lim_{n\to\infty} \frac{n^2}{n \log n}$$

$$= \lim_{n\to\infty} \frac{n}{\log n}$$

$$\lim_{n\to\infty} \frac{g'(n)}{f'(n)} = \lim_{n\to\infty} \frac{1}{(\log e)/n}$$

$$= \lim_{n\to\infty} \frac{n}{\log e}$$

$$= \infty$$

Karena limit dari perbandingan $f(n)$ dan $g(n)$ sebesar ∞ maka $f(n)) \neq O(g(n))$. □

Teorema 2.5. 1. Jika $f(n)=a_0 + a_1 n + \ldots + a_m n^m$ adalah fungsi polinomial dan $g(n)=n^m$ maka $f(n)=O(g(n))$. ♦

Bukti :

Dipilih $c=|a_0|+|a_1|+\ldots+|a_m|$ dan $n \geq 1$.

$$|f(n)| = |a_0 + a_1 n + \ldots + a_m n^m|$$

Dengan pertidaksamaan segitiga didapatkan

$$|f(n)| \leq |a_0| + |a_1 n| + \ldots + |a_m n^m|$$

$$\leq |a_0| + |a_1| n + \ldots + |a_m| n^m$$

$$\leq |a_0| n^m + |a_1| n^m + \ldots + |a_m| n^m$$

$$= (|a_0| + |a_1| + \ldots + |a_m|) n^m = cn^m = c|g(n)|$$

Terbukti bahwa $f(n)=O(g(n))$. □

Teorema 2.5. 2. Misalkan S adalah himpunan semua fungsi dengan domain D, di mana $D=N,Z$ atau R dan kodomain $[0,\infty)$. Diberikan $f_1(n)$, $f_2(n)$, $g_1(n)$ dan $g_2(n)\in S$ sedemikian sehingga $f_1(n)=O(g_1(n))$ dan $f_2(n)=O(g_2(n))$ maka berlaku :

(a). $f_1(n)+f_2(n)=O(max\{g_1(n),g_2(n)\})$

(b). $f_1(n)\,f_2(n)=O(g_1(n)g_2(n))$. ♦

Bukti :

(a). Karena. $f_1(n) = O(g_1(n))$ dan $f_2(n) = O(g_2(n))$ maka terdapat bilangan positif c_1, c_2, k_1 dan k_2 sedemikian sehingga jika $n\in$D dan $n \geq k_1$, berlaku $|f_1(n)| \leq c_1 |g_1(n)|$ dan jika $n \geq k_2$ berlaku $|f_2(n)| \leq c_2 |g_2(n)|$. Selanjutnya ambil $k=max\{k_1,k_2\}$ dan $c= c_1+ c_2$.

$$|f_1(n)+f_2(n)| = |f_1(n)|+|f_2(n)|$$
$$\leq c_1|g_1(n)|+c_2|g_2(n)|$$
$$= c_1 g_1(n)+c_2 g_2(n)$$
$$\leq c_1\, max\{g_1(n),g_2(n)\}+c_2\, max\{g_1(n),g_2(n)\}$$
$$= (c_1+c_2)\, max\{g_1(n),g_2(n)\}$$
$$= c\, max\{g_1(n),g_2(n)\}$$
$$= c|max\{g_1(n),g_{2(n)}\}|$$
$$f_1(n)+f_2(n) = O(max\{g_1(n),g_2(n)\})$$

(b). Karena $f_1(n) = O(g_1(n))$ dan $f_2(n) = O(g_2(n))$ maka terdapat bilangan positif c_1, c_2, k_1 dan k_2 sedemikian sehingga jika $n\in$D dan $n \geq k_1$, berlaku

$|f_1(n)| \le c_1|g_1(n)|$ dan jika $n \ge k_2$ berlaku $|f_2(n)| \le c_2|g_2(n)|$. Selanjutnya ambil $k=max\{k_1,k_2\}$ dan $c= c_1 . c_2$.

$$|f_1(n)f_2(n)| = |f_1(n)||f_2(n)|$$

$$\le c_1|g_1(n)|c_2|g_2(n)|$$

$$= c_1c_2|g_1(n)g_2(n)|$$

$$= c|g_1(n)g_2(n)|$$

$f_1(n)f_2(n) = O(g_1(n)g_2(n))$. □

Contoh 2.5. 3. Jika suatu fungsi $f(n)=3n^3+2n^2$ merupakan fungsi dari waktu tempuh suatu algoritma maka "*Big Oh*-nya" adalah n^3 yang dinotasikan $f(n)=O(n^3)$.

Analisa :

Berdasarkan definisi 2.5.1 Jika $f(n)=O(n^3)$ maka terdapat dua bilangan positif c dan k .

Ambil c=5, $n\ge1$.

$$|f(n)| = |3n^3+2n^2|$$

$$\le |3n^3| + |2n^2|$$

$$\le 3n^3 + 2n^3$$

$$= 5n^3$$

$$= c|g(n)|$$

Jadi terbukti bahwa "*Big Oh*" dari $f(n)=3n^3+2n^2$ adalah n^3 dinotasikan $f(n)=O(n^3)$. □

2.6. Waktu Tempuh (*Running Time*)

Pada umumnya analisa algoritma dilakukan untuk mengetahui efisiensi suatu algoritma dalam menyelesaikan suatu permasalahan. Tingkat efisiensi suatu algoritma menyangkut waktu tempuh (*running time*) dari algoritma tersebut.

Proses dari suatu algoritma dalam mencari solusi dari suatu masalah memerlukan waktu tertentu. Satuan waktu yang dibutuhkan diharapkan dalam waktu yang relatif singkat. Adapun hal-hal yang mempengaruhi waktu tempuh antara lain :

1. banyaknya langkah

   Makin banyak langkah atau intruksi yang digunakan maka makin lama waktu tempuh yang dibutuhkan dalam proses tersebut.

2. besar input data

   Ukuran atau besar input yang digunakan akan sangat berpengaruh pada proses perhitungan. Semakin besar input yang diberikan makin lama waktu tempuh yang dibutuhkan.

3. jenis operasi

   Waktu tempuh juga dipengaruhi oleh jenis operasi yang digunakan. Jenis operasi antara lain meliputi operasi aritmatika, operasi nalar atau logika.

4. Komputer dan kompilator

   Faktor ini diluar dari rancangan atau pembuatan algoritma yang efisien. Walaupun algoritma yang dibuat sudah mencapai waktu yang efisien namun bila digunakan komputer yang berkemampuan lambat maka waktu tempuhnya akan menjadi lebih lambat. Kompilator yang digunakan juga berpengaruh terhadap waktu tempuh suatu algoritma.

Untuk menyederhanakan perhitungan dalam melakukan analisa algoritma maka perhitungan hanya dikenakan pada satu *loop* atau iterasi dari algoritma. Aturan umum dalam melakukan analisa algoritma yaitu :

1. *Rule* 1 : jika $f_1(n)=O(g_1(n))$ dan $f_2(n)=O(g_2(n))$ maka berlaku :

   (a). $f_1(n)+f_2(n)=O(max\{g_1(n),g_2(n)\})$

   (b). $f_1(n)f_2(n)=O(g_1(n)g_2(n))$.

2. *Rule* 2 : *running time* dari suatu *loop* adalah jumlah iterasi yang dilakukan untuk menyelesaikan *statemen-statemen* dalam *loop*.
3. *Rule* 3 : dalam *loop* bersarang, yang menjadi pedoman adalah *loop* terdalam. *Running time*nya merupakan hasil kali ukuran semua *input* pada *loop* yang diselesaikan.
4. *Rule* 4 : waktu tempuh untuk *statemen* berurutan ditentukan dengan aturan jmlahan yaitu bahwa waktu tempuh pada barisan tidak melebihi suatu konstanta yang merupakan waktu tempuh terbesar pada beberapa *statement* berurutan .

Salah satu metode yang digunakan untuk menghitung *running time* suatu algoritma adalah dengan membuat tabel *cost* dan *time* untuk total langkah yang telah dilaksanakan oleh statemen. Notasi *cost* merupakan suatu konstanta yang diberikan untuk setiap statement yang besarnya tergantung dari jenis kompiler dan mesin eksekusi tertentu. Setiap langkah yang dieksekusi mempunyai nilai *cost* yang berbeda, diasumsikan bahwa eksekusi tiap langkah ke-*i* diberikan suatu konstanta c_i. Sedangkan *time* merupakan jumlah langkah tiap *statement* yang telah dieksekusi. Dengan mengkombinasikan keduanya maka diperoleh perhitungan total setiap *statement*, dan penjumlahan total tiap *statemen* diperoleh total langkah keseluruhan algoritma.

Dalam menganalisis suatu algoritma diperlukan suatu gambaran yang sederhana. Pertama, menggantikan nilai *cost* yang sesungguhnya pada masing-masing statement dengan konstanta c_i. Misalkan diketahui *running time* suatu algoritma $an+b$ untuk nilai a dan b yang tergantung pada konstanta c_i. Untuk suatu input yang sangat besar nilai konstanta c_i tidak berpengaruh sehingga konstanta c_i selanjutnya diabaikan. Kedua, bahwa *running time* dinyatakan dengan suatu fungsi pertumbuhan (misalkan dalam notasi "*Big Oh*"). Suatu algoritma dikatakan lebih efisien jika *running time*-nya mempunyai derajat rendah pada fungsi pertumbuhannya.

Contoh 2.6. 1. Misalkan a merupakan sebuah *array* yang terdiri dari n elemen yaitu $a[1], a[2], \ldots, a[n]$. Akan dihitung penjumlahan dari a.

```
Function sum(a:elementlist; n:integer):real;
var s: real;
    i:integer;
begin
  s:=0;
  for i:=1 to n do
    s:=s+a[i];
    sum:=s;
end; {of sum}
```

Analisa algoritma :

Tabel 2.6. 1. Nilai *cost* dan *time* pada *procedure* penjumlahan

	Langkah	*cost*	*time*
1.	`var s:real; i:integer;`	0	0
2.	`begin`	0	0
3.	`s:=0;`	c_1	1
4.	`for i:=1 to n do`	c_2	$n+1$
5.	`s:=s+a[i];`	c_3	n
6.	`sum:=s;`	c_4	1
7.	`end; {of sum}`	0	1

Pada tabel 2.7.1 diberikan nilai untuk langkah tiap eksekusi (*cost*) dan *time* dari masing-masing *statemen* dalam *function sum*. Perlu diperhatikan bahwa *time* dari baris ke-4 sebesar $n+1$ bukan n, ini karena i mempunyai penambahan menjadi $n+1$ sebelum *loop for* diakhiri. Sehingga total *running time* $T(n)$ akan diperoleh sebagai berikut :

$$T(n) = c_1 + c_2(n+1) + c_3 n + c_4$$

$$= (c_2 + c_3)n + (c_1 + c_2 + c_4)$$

Running time ini dapat ditulis dengan $T(n)=an+b$ dengan a dan b adalah konstanta yang tergantung pada nilai *cost* statemen c_i. Dengan notasi "*Big Oh*" maka diketahui bahwa *running time*nya sebesar n atau dinotasikan dengan $T(n)=O(n)$. □

Running time suatu algoritma $T(n)$ dengan n input data dapat dibedakan atas tiga keadaan yaitu *worst case running time* di mana waktu yang ditempuh oleh algoritma adalah waktu maksimum, *average case running time* di mana analisa waktu rata-rata yang ditempuh oleh algoritma, *best case running time* di mana waktu yang ditempuh oleh algoritma adalah waktu minimum.