BAB II
TINJAUAN PUSTAKA

2.1 Tinjauan Perusahaan

2.1.1 Sejarah Pusat Penelitian dan Pengembangan Sumber Daya Air

Adapun sejarah Pusat Penelitian dan Pengembangan Sumber Daya Air dari tahun ke tahun dapat dilihat sebagai berikut:

1936 Departement voor Verkeer en Waterstaat (V en W) mendirikan Hydrodinamisch Laboratorioum

1947 Instituut voor Weg en Waterbouwkundige Onderzoekingen

1950 Institut Teknik Air dan Tanah

1966 Lembaga Penyelidikan Masalah Air (LPMA)

1974 Direktorat Penyelidikan Masalah Air (DPMA)

1984 Pusat Penelitian dan Pengembangan Pengairan, berada dibawah Badan Litbang Departemen Pekerjaan Umum

1999 Pusat Penelitian dan Pengembangan Teknologi Sumber Daya Air, berada dibawah Badan Litbang Departemen Permukiman dan Pengembangan Wilayah (Kimbangwil)

2001 Pusat Penelitian dan Pengembangan Sumber Daya Air, berada dibawah Badan Litbang Departemen Permukiman dn Prasarana Wilayah

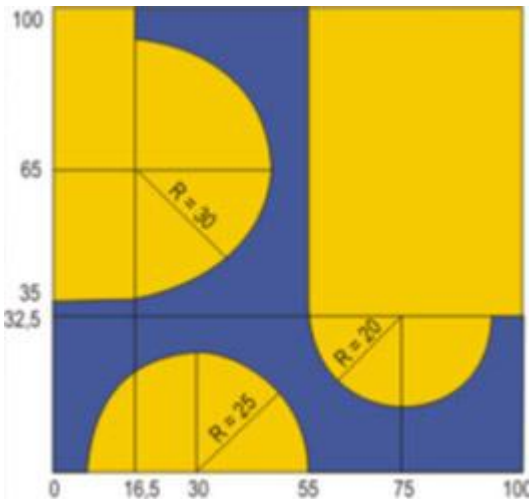
2004 Pusat Penelitian dan Pengembangan Sumber Daya Air, berada dibawah Badan Litbang Departemen Pekerjaan Umum

2010 Pusat Penelitian dan Pengembangan Sumber Daya Air, berada dibawah Badan Litbang Kementerian Pekerjaan Umum

2.1.2 Logo Pusat Penelitian dan Pengembangan Sumber Daya Air

Berikut adalah Motto dan lambang Pusat Penelitian dan Pengembangan Sumber Daya Air .

.



Gambar II.1 Lambang Pusat Penelitian dan Pengembangan Sumber Daya Air

Keterangan gambar:

1. Lambang Departemen PU berlukiskan Baling-baling dengan ketentuan seperti tercantum pada gambar.
2. Warna dasar lambang adalah kuning (kuning kunyit).
3. Warna baling-baling adalah biru kehitam-hitaman.
4. Penggunaan lambang : lihat Manual Tata Persuratan.

Arti Simbolis Lambang Departemen Pekerjaan Umum

1. Baling-baling

   Menggambarkan D I N A M I K A Berdaun 3 yang merupakan segitiga berdiri tegak lurus menggambarkan STABILITAS Secara keseluruhan menggambarkan DINAMIKA YANG STABIL dan STABILITAS YANG DINAMIS.
2. Bagian daun baling-baling yang mengarah keatas

   Melambangkan PENCIPTAAN RUANG.
3. Bagian lengkungnya dari daun baling-baling

   Memberikan perlindungan untuk ruang kerja tempat tinggal bagi manusia.
4. Bagian daun baling-baling yang mengarah ke kiri dengan bagian lengkungnya yang telungkup

   Menggambarkan penguasaan bumi da alam dan pengusahaan untuk sebesar-besarnya kemakmuran rakyat. Garis Horizontal : bentang jalan / jembatan diatas sungai sebagai usaha untuk pembukaan dan pembinaan daerah.

5. Bagian daun baling-baling yang mengarah ke kanan dengan bagian lengkungnya yang terlentang

   Menggambarkan usaha pengendalian dan Penyaluran untuk dimanfaatkan bagi sebesar-besarnya kemakmuran rakyat.

   Garis Horizontal : bentang jalan / jembatan diatas sungai sebagai usaha untuk pembukaan dan pembinaan daerah.
6. Baling-baling dengan 3 daun ini menggambarkan

   Tiga unsur kekaryaan Departemen Pekerjaan Umum. Tirta, Wisma (Cipta) dan Marga.

   Trilogi Departemen Pekerjaan Umum, Bekerja keras, Bergerak cepat, Bertindak tepat.
7. W A R N A

   Tiga unsur kekaryaan Departemen Pekerjaan Umum. Tirta, Wisma (Cipta) dan Marga.

   Warna kuning sebagai warna dasar melambangkan keagungan yang mengandung arti KeTuhanan Yang Maha Esa, Kedewasaan dan Kemakmuran.

   Warna biru kehitam-hitaman, mengandung arti Keadilan Sosial, Keteguhan hati, Kesetiaan pada tugas dan ketegasan bertindak.
   Warna yang berbentuk dari warna dasar dan likisan baling-baling membentuk huruf-huruf P.U.
8. Lambang P.U.

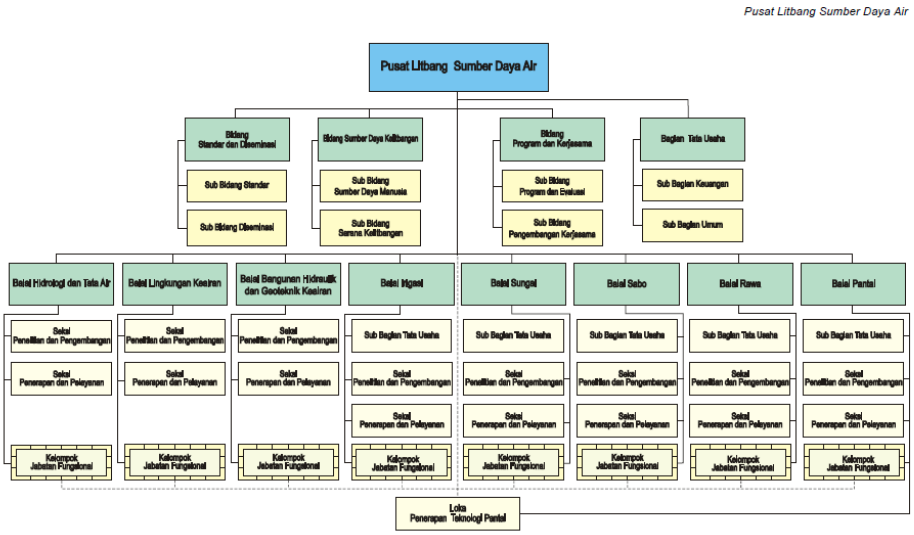
   Menggambarkan fungsi dan peranan Departemen Pekerjaan Umum dalam pembangunan dan Pembinaan prasarana guna memanfaatkan bumi dan air serta kekayaan alam bagi kemakmuran rakyat, berlandaskan Pancasila.

2.1.3 Badan Hukum Instansi

Sebagaimana tercantum dalam Peraturan Menteri Pekerjaan Umum Nomor : 08/PRT/M/2010 tanggal 8 Juli 2010 Tentang Organisasi dan Tata Kerja Kementrian Pekerjaan Umum, menurut pasal 837 sampai dengan pasal 855.

2.1.4 Struktur Organisasi dan Job Description

Struktur organisasi Pusat Penelitian dan Pengembangan Sumber Daya Air dapat dilihat pada Gambar II.2.



Gambar II.2 Struktur organisasi Pusat Penelitian dan Pengembangan Sumber Daya Air Bandung

Tugas :

Sesuai dengan Peraturan Menteri PU Nomor 08/PRT/M/2010 tentang Organisasi Dan Tata Kerja Kementerian Pekerjaan Umum, Kementerian Pekerjaan Umum mempunyai tugas : menyelenggarakan urusan di bidang pekerjaan umum dalam pemerintahan untuk membantu Presiden dalam menyelenggarakan pemerintahan negara.

Fungsi :

1. Perumusan, penetapan dan pelaksanaan kebijakan di bidang pekerjaan umum.
2. Pengelolaan barang milik / kekayaan negara yang menjadi tanggung jawab Kementerian Pekerjaan Umum.
3. Pengawasan atas pelaksanaan tugas di lingkungan Kementerian Pekerjaan Umum.
4. Pelaksanaan bimbingan teknis dan supervisi atas pelaksanaan urusan Kementerian Pekerjaan Umum di daerah.
5. Pelaksanaan kegiatan teknis yang berskala nasional [11] [12].

2.2 Landasan Teori

2.2.1 Dokumen

Dokumen adalah sebuah tulisan yang memuat informasi. Biasanya dokumen ditulis dikertas dan informasinya ditulis memakai tinta baik memakai tangan atau memakai media elektronik (seperti printer). Namun pada zaman sekarang, media elektronik seperti komputer juga dapat digunakan untuk menyimpan dan menampilkan dokumen. menurut kamus besar bahasa Indonesia, dokumen merupakan surat tertulis atau tercetak yang dapat dipakai atau dijadikan sebagai bukti keterangan [3] [14]. Sehingga dokumen merupakan bukti asli yang tercetak memakai tinta maupun tersimpan melalui media elektronik yang digunakan untuk mendukung suatu kebenaran atau keaslian suatu keterangan atau hasil penelitian [15]. Setiap penelitian yang dilakukan oleh Pusat Penelitian dan Pengembangan Sumber Daya Air didokumentasikan dalam media kertas maupun elektronik.

2.2.2 Abstrak

Abstrak adalah representasi dari isi dokumen yang singkat dan tepat. Abstrak merupakan bentuk ringkas dari isi suatu dokumen yang terdiri atas bagian-bagian penting dari suatu tulisan, dan mendeskripsikan isi dan cakupan dari tulisan. Dengan adanya abstrak pada dokumen dapat memudahkan mendapatkan informasi suatu dokumen tanpa harus membaca isi dari keseluruhan dokumen.

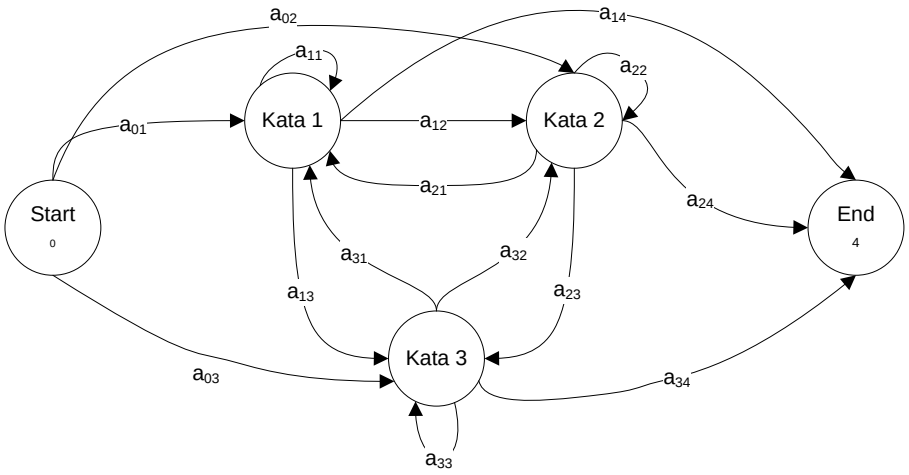
2.2.3 Kata kunci

Kata kunci merupakan kata atau konsep dengan keistimewaan, dengan ini berarti kata apapun yang digunakan sebagai kunci dan kode atau digunakan untuk menghubungkan kekata lain atau informasi lain. Kata kunci suatu dokumen mewakili isi dari dokumen tersebut. Kata kunci diperlukan pada saat proses pencarian dokumen. Sehingga memilih atau menentukan kata kunci suatu dokumen sangatlah penting [1]. Kata kunci yang paling baik adalah kata kunci yang terdiri dari rangkaian kata benda atau kata sifat [2].

Penentuan kata kunci menggunakan rantai markov atau markov chain. Proses markov merupakan proses stokastik masa lalu tidak mempunyai pengaruh pada masa yang akan datang bila masa sekarang diketahui. markov chain berisi

probabilitas transisi dimana kata pada state berpengaruh pada kata sebelumnya. [5] [16]

Kata kunci yang dihasilkan terdiri dari gabungan 2 kata yaitu pasang kata yang terdapat didalam abstrak. Penentuan kata kunci atau pasangan kata yang dipilih adalah berdasarkan nilai dari pasangan kata yang terdapat pada abstrak. Keterkaitan kata satu dengan lainnya dapat dilihat pada gambar gambar II.3 dengan menggunakan markov chain. kata 1, kata 2 dan kata 3 adalah contoh kata yang terdapat pada abstrak. nilai a_{ij} adalah probabilitas transisi yang didapatkan dari algoritma *Baum Welch*. a_{0j} merupakan *state* awal (phi) pada algoritma *Baum Welch*.



Gambar II.3 Rangkaian kata dalam markov chain **[16]**

Probabilitas pasangan = $Phi_i.b_i(O_i).a_{ij}.b_j(O_j)$ (II.1) [16]

Contoh penentuan kata kunci :

Pasang kata yang terdapat pada abstrak adalah pasangan kata 1 dan kata2, kata 2 dan kata 3, kata 3 dan kata2. Maka nilai yang dihasilkan setiap pasangan adalah

Probabilitas pasangan kata 1 dan kata 2

$= Phi_{kata1}.b_{kata1}(O_{kata1}).a_{kata1kata2}.b_{kata2}(O_{kata2})$

Probabilitas pasangan kata 2 dan kata 3

$= Phi_{kata2}.b_{kata2}(O_{kata2}).a_{kata2kata3}.b_{kata3}(O_{kata3})$

Probabilitas pasangan kata 3 dan kata 2

$= Phi_{kata3}.b_{kata3}(O_{kata3}).a_{kata3kata2}.b_{kata3}(O_{kata2})$

2.2.4 Algoritma

Algoritma merupakan salah satu cabang ilmu komputer yang membahas prosedur penyelesaian suatu permasalahan [17]. Algoritma adalah urutan langkah-langkah dalam menentukan suatu masalah. Algoritma juga dapat didefinisikan dengan deretan langkah komputasi yang mentransformasikan masukan (*input*) menjadi keluaran (*output*) [18]. Jadi algortima adalah serangkaian urutan langkah-langkah atau prosedur untuk menyelesaikan suatu masalah dengan memproses nilai masukan menjadi nilai keluaran.

Algoritma dapat ditulis dalam bentuk *pseudocode* ataupun *flowchart*. *Pseudocode* berasal dari kata *pseudo* yang artinya mirip atau menyerupai dan *code* berarti kode sehingga *pseudocode* adalah kode yang menyerupai kode program sebenarnya [17]. Sedangkan *flowchart* merupakan bentuk gambar atau diagram yang mempunyai aliran satu atau dua arah secara sekuensial.

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 [17] [7].

2.2.4.1 Kompleksitas waktu asimptotik

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.

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

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

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

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

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

$= O(n)$

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 :

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

algoritma polinomial algoritma eksponensial

Penjelasan masing-masing kelompok algoritma adalah sebagai berikut :

Tabel II.1 Kelompok Algoritma dan penjelasannya

Kelompok Algoritma	**Penjelasan**

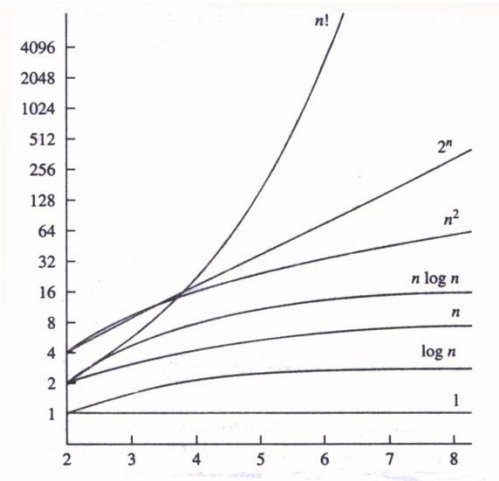
$O(1)$	Kompleksitas $O(1)$ berarti waktu pelaksanaan algoritma adalah tetap, tidak bergantung pada ukuran masukan. Contohnya prosedur tukar di bawah ini:
	
	procedure tukar(**var** a:**integer**; **var** b:**integer**);
	var
	temp:**integer**;
	begin
	temp:=a;
	a:=b;
	b:=temp;
	end;
	
	Di sini jumlah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

Kelompok Algoritma	Penjelasan
$O(\log n)$	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).
$O(n^2)$	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 semul.

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 semul.
$O(2^n)$	Algoritma yang tergolong kelompok ini mencari solusi persoalan secara "*brute force*", misalnya pada algoritma mencari sirkuit Hamilton (lihat Bab 9). 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* - lihat bab 9). Bila n = 5, maka waktu pelaksanaan algoritma adalah 120. Bila n dijadikan dua kali semula, maka waktu pelaksanaan algoritma menjadi faktorial dari 2n.

Nilai masing-masing fungsi untuk setiap bermacam-macam nilai *n*

log n	*n*	*n log n*	n^2	n^3	2^n	*n*!
0	1	0	1	1	2	1
1	2	2	4	8	4	2
2	4	8	16	64	16	24
3	9	24	64	512	256	362880
4	16	64	256	4096	65536	20922789888000
5	32	160	1024	32768	4294967296	(terlalu besar)



Gambar II.4 Grafik nilai masing-masing fungsi untuk setiap bermacam-macam n

Sebuah masalah yang mempunyai algoritma dengan kompleksitas *polynomial* dengan kasus-terburuk dianggap mempunyai algoritma yang "bagus"; yang artinya masalah tersebut mempunyai algoritma yang mangkus, dengan catatan *polynomial* tersebut berderajat rendah. Jika polinomnya berderajat tinggi, waktu yang dibutuhkan untuk mengeksekusi algoritma tersebut panjang. [7] [8]

2.2.5 Algoritma *Baum Welch*

Algoritma *Baum Welch* merupakan salah satu algortima yang digunakan untuk menyelesaikan suatu masalah dalam *Hidden Markov Model*. Hidden Markov Model merupakan perluasan dari proses Markov yaitu proses stokastik masa lalu tidak mempunyai pengaruh pada masa yang akan datang bila masa sekarang diketahui [19] [5]. Dinamai Algoritma *Baum Welch* berasal dari nama Leonard E. Baum dan Lloyd R. Welch [3]. Permasalahan pada Algoritma *Baum*

Welch adalah bagaimana menghasilkan nilai parameter model A, B dan π yang optimal sehingga mendapatkan hasil rangkaian dari rangkaian pengamatan atau observasi. Algoritma *Baum Welch* menggunakan parameter-parameter dari HMM untuk dihitung sehingga mendapatkan model yang optimal atau terbaik. Mengoptimalkan nilai parameter yang artinya memaksimalkan nilai probabilitas dari rangkaian pengamatan P(O|λ), dengan diberikannya model λ(A, B, π). Tidak ada pendekatan analitik untuk permasalahan ini, akan tetapi terdapat prosedur iterative dengan menggunakan Algoritma *Baum Welch* [19] [20].

Adapun penjelasan dari parameter-parameternya sebagai berikut:

1. $A=\{a_{ij}\}$, himpunan nilai probabilitas transisi yaitu kemungkinan perpindahan dari *state* i ke *state* j.
2. $B=\{b_j(o_k)\}$, himpunan nilai probabilitas emisi yaitu kemungkinan simbol pengamatan pada *state* j.
3. $\pi=\{\pi_i\}$, himpunan nilai perpindahan *state* awal.

Adapun parameter lainnya yaitu N dan M. N adalah jumlah *state* dalam model dan M adalah jumlah simbol pengamatan.

Langkah-langkah Algoritma *Baum Welch* adalah sebagai berikut :

1. Inisialisasi nilai parameter N, M, A, B dan π.
2. Menghitung probabilitas maju (*forward probability*) dan probabilitas mundur (*backward probability*). Hasil dari probabilitas maju dan mundur akan digunakan dalam perhitungan menentukan nilai probabilitas transisi dan emisi.

Untuk menghitung probabilitas maju (*forward probability*) digunakan prosedur forward.

Prosedur Forward

Inisialisasi

$$\propto_1 (i) = \pi_i b_i(O_i) \qquad 1 \leq i \leq N \qquad \text{(II.2)}$$

Induksi

$$\propto_t (j) = \left[\sum_{i=1}^{N} \propto_{t-1} (i) a_{ij}\right] b_j(O_t) \qquad 2 \leq t \leq T \; ; \; 1 \leq i \leq N \qquad \text{(II.3)}$$

Untuk menghitung probabilitas mundur (*backward probability*) digunakan prosedur forward.

Prosedur Backward

Inisialisasi

$$\beta_T(i) = \frac{1}{N} \qquad 1 \leq i \leq N \qquad \text{(II.4)}$$

Induksi

$$\beta_t(i) = \left[\sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)\right] \qquad t=T-1,\ldots,1 \; ; \; 1 \leq i \leq N \qquad \text{(II.5)}$$

3. Menghitung kembali probabilitas transisi, probabilitas emisi dan *state* awal yang baru.

$$\pi_bar_i = \gamma_1(i) \qquad (t=1) \qquad \text{(II.6)}$$

$$a_bar_{ij} = \frac{\sum_{t=1}^{T-1} \varepsilon_{t(i,j)}}{\sum_{t=1}^{T-1} \gamma_{t(i)}} \qquad \text{(II.7)}$$

$$b_bar_j(k) = \frac{\sum_{t=1}^{T} \gamma_{t(j)}}{\sum_{t=1}^{T} \gamma_{t(j)}} \qquad \text{(II.8)}$$

(untuk pembilangan pada saat $O_t=V_k$)

Nilai $\gamma_{t(i)}$dan $\varepsilon_{t(i,j)}$ didapat dari :

$$\gamma_t(i) = \frac{\alpha_t(i)\, \beta_t(i)}{\mathrm{P(O|\lambda)}} = \frac{\alpha_t(i)\, \beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i).\beta_t(i)} \qquad \text{(II.9)}$$

$$\varepsilon_{t(i,j)} = \frac{\alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(i)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(i)} \qquad \text{(II.10)}$$

4. Mengulang kembali langkah 2 hingga parameter-parameter dari HMM konvergen (berubah secara tidak signifikan) dengan membandingkan P(O|λ) yang lama dengan yang baru. Jika P(O|λ) baru lebih besar dari P(O|λ) lama maka telah konvergen.

$\max[Q(\lambda|\lambda_{bar})] = P(O|\lambda_bar) \geq P(O|\lambda)$

$Q(\lambda|\lambda_bar) = \sum_{all\ S} \frac{P(O,S|\lambda)}{P(O|\lambda)} logP(O,S|\lambda_bar)$

dapat ditulis kembali,

$$Q(\lambda|\lambda_{bar}) = Q_{a_i}(\lambda_{bar}, a_{bar_i}) + Q_{b_j}\left(\lambda_{bar}, b_{bar_j}\right) \quad \text{(II.11)}$$

dengan,

$$Q_{a_i}(\lambda_{bar}, a_{bar_i}) = \sum_i \sum_j \sum_t \frac{P(O,S_{t-1}=i,S_t=j|\lambda)}{P(O|\lambda)} loga_bar_{ij} \quad \text{(II.12)}$$

$$Q_{b_j}\left(\lambda_{bar}, b_{bar_j}\right) = \sum_i \sum_j \sum_{t\epsilon O_t=v_k} \frac{P(O,S_t=j|\lambda)}{P(O|\lambda)} logb_bar_j(k) \quad \text{(II.13)}$$

5. Didapat nilai probabilitas transisi (A), probabilitas emisi (B) dan *state* awal (π) yang optimal [19] [20] [21].

Perhitungan pada langkah inisialisasi, nilai probabilitas transisi menggunakan smoothing bigram karena apabila terdapat probabilitas pasang kata yang 0 maka nilainya akan dihampiri oleh probabilitas smoothing bigram. smoothing bigram yang digunakan adalah *Jelinec-Mercer smoothing* karena pada penelitian yang dilakukan oleh Yudi Wibisono menyebutkan bahwa *Jelinec-Mercer smoothing* dapat meningkatkan kinerja sistem dengan nilai $\lambda = 0.1$. Rumus *Jelinec-Mercer smoothing* adalah sebagai berikut:

$$P(C_i|C_{i-1}) = \left(\lambda * \acute{P}(C_i|C_{i-1})\right) + \left((\lambda - 1) * \acute{P}(C_i)\right) \quad \text{(II.14)}$$

Keterangan:

$\acute{P}(C_i|C_{i-1}) = \frac{count(C_{i-1}C_i)}{count(C_i)}$, kemungkinan kemunculan pasang katga

$\acute{P}(C_i) = \frac{count(C_i)}{jumlah_kata}$, kemungkinan kemunculan suatu kata

$\lambda = 0.1$, konstanta antara 0 sampai 1

Nilai probabilitas transisi adalah $A=\{a_{ij}\}=\{P(C_j|C_i)\}=\{P(C_i|C_{I.1})\}$. Probabilitas transisi merupakan kemungkinan atau probabilitas suatu kata dipengaruhi oleh kata sebelumnya. C_i adalah kata pada *state* i dan $C_{I.1}$ atau C_j adalah kata sebelumnya. $count(C_{i-1}C_i)$ adalah jumlah pasangan kata yang muncul pada abstrak, $count(C_i)$ adalah jumlah kata pada *state* i yang muncul pada abstrak dan $jumlah_kata$ adalah jumlah seluruh kata pada abstrak. [16] [22]

2.2.5.1 *Flowchart* Algortitma *Baum Welch*

Flowchart merupakan bentuk gambar atau diagram yang mempunyai aliran satu atau dua arah secara sekuensial. Gambar II.3 merupakan gambar *flowchart* dari Algoritma *Baum Welch*.

Nilai inputan pada Algoritma *Baum Welch* adalah jumlah kata pada abstrak tanpa mengalami pengulangan kata yang telah dihitung pada proses diluar algoritma *Baum Welch*. Jumlah kata dimasukkan pada nilai N pada Algoritma *Baum Welch*. Pada algoritma *Baum Welch* terdapat parameter M yaitu panjang observasi. Dikarenakan nilai N dan M adalah sama maka untuk parameter M dihilangkan dan digantikan dengan parameter N saja.

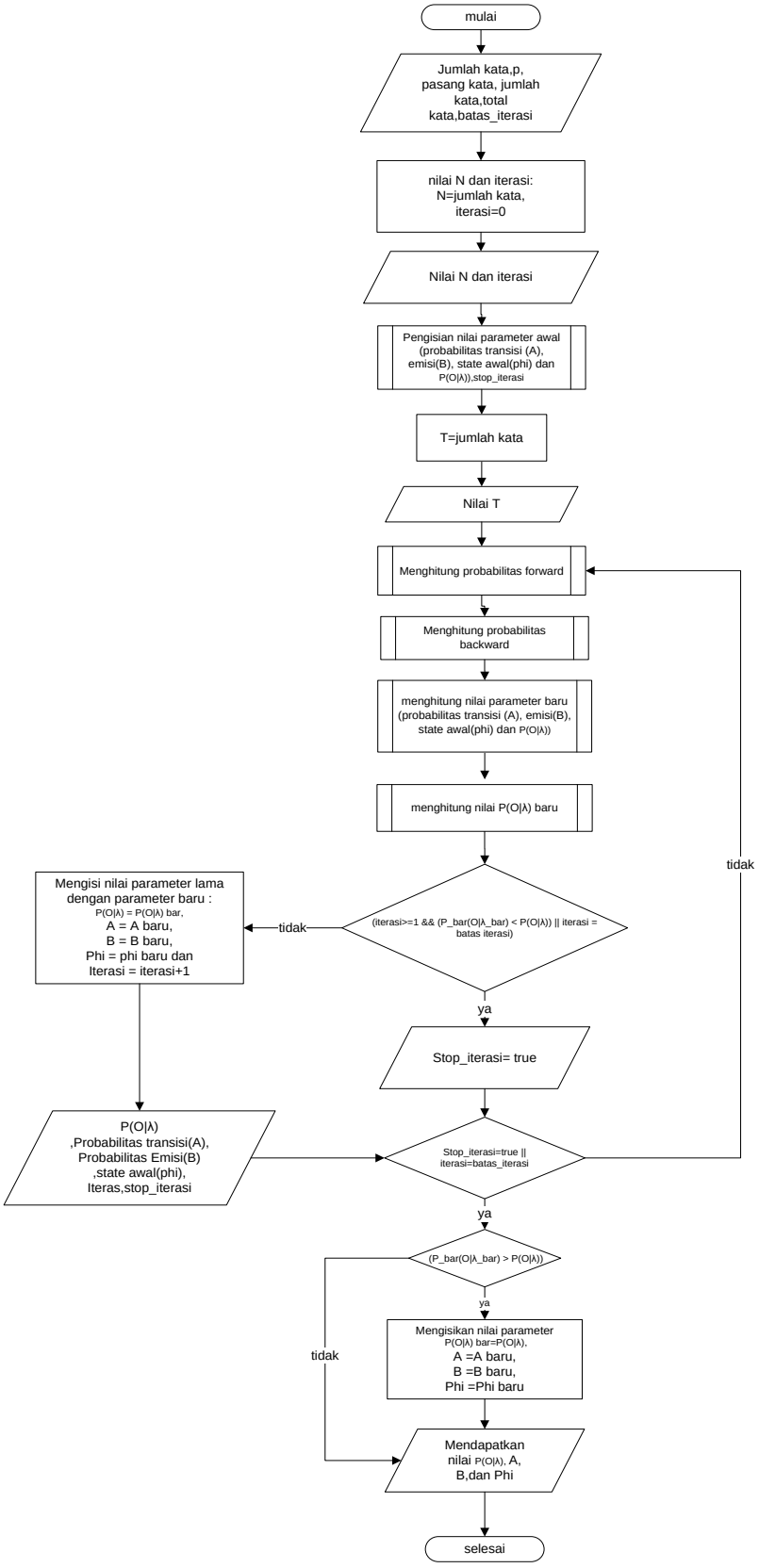
Selanjutnya masuk keproses inisialisasi parameter baru yaitu himpunan probabilitas transisi (A), himpunan probabilitas emisi (B), himpunan *state* awal (Phi) dan P(O|λ) awal yang nantinya akan diisikan dengan P(O|λ) yang baru. Nilai dari Parameter baru ini akan digunakan dalam proses selanjutnya yaitu proses perhitungan probabilitas forward dan backward.

Setelah menghitung probabilitas forward dan backward maka hasil dari kedua proses ini akan digunakan dalam menentukan parameter yang baru. Hasil dari perhitungan parameter baru digunakan kembali untuk mendapatkan nilai P(O|λ). Nilai P(O|λ) akan dijadikan tolak ukur apakah seluruh parameter telah bernilai optimal.

Pada iterasi pertama P(O|λ) lama akan diisikan dengan P(O|λ) baru dan parameter lama digantikan dengan parameter baru. Proses perhitungan kembali nilai probabilitas forward dan backward untuk mendapatkan nilai parameter yang baru. Barulah pada iterasi lebih dari 1 nilai P(O|λ) akan dibandingkan. Jika P(O|λ) baru lebih kecil dari P(O|λ) lama dan iterasi lebih dari 1 atau iterasi belum mencapai batas iterasi maka dilakukan perulangan mencari parameter baru. Jika nilai P(O|λ) baru lebih besar dari P(O|λ) lama maka iterasi berhenti dan mendapatkan nilai parameter yang baru atau telah optimal.

Hasil dari prosedur Algoritma *Baum Welch* adalah nilai probabilitas transisi, probabilitas emisi dan *state* awal. probabilitas transisi merupakan kemungkinan atau probabilitas perpindahan *state* dari *state* I ke *state* j. Probabilitas emisi

merupakan kemungkinan atau probabilitas pengamatan pada *state* j. *State* awal merupakan kemungkinan suatu kata pada keadaan di*state* awal.



Gambar II.5 *Flowchart* algoritma *baum welch*

Pada *flowchart* Algoritma *Baum Welch* diatas masih terdapat proses yang masih dapat diuraikan lagi. Penjelasan dari setiap proses atau prosedur yang terdapat pada Algoritma *Baum Welch* akan dijelaskan pada bab 3 subbab 3.1.1.2.

2.2.6 Perancangan Terstruktur

Dalam tahap perancangan suatu sistem diperlukan adanya teknik-teknik penyusunan sistem untuk menganalisa dan mendokumentasikan data yang mengalir di dalam sistem tersebut. Tools yang digunakan untuk memodelkan adalah diagram konteks dan *Data Flow Diagram* (DFD).

2.2.6.1. Diagram konteks

Diagram konteks adalah arus data yang berfungsi untuk menggambarkan keterkaitan aliran-aliran data antara sistem dengan bagian-bagian luar (kesatuan luar). Kesatuan luar ini merupakan sumber arus data atau tujuan data yang berhubungan dengan sistem informasi tersebut. Diagram konteks bisa disebut dengan *"*Model sistem pokok (*fundamental system model*) mewakili keseluruhan elemen *software* dengan masukan (*input*) dan keluaran (*output*) yang diidentifikasi dengan anak panah masuk dan keluar memperlihatkan sumber data*"*.

2.2.6.2. DFD

Data Flow Diagram digunakan untuk menggambarkan suatu sistem yang telah ada atau sistem baru yang akan dikembangkan secara logika tanpa mempertimbangkan lingkungan fisik dimana data tersebut mengalir atau lingkungan fisik dimana data tersebut akan disimpan. *Data Flow Diagram* juga digunakan pada metodologi pengembangan sistem yang terstruktur [23].

2.2.7 DELPHI

Delphi adalah sebuah IDE Compiler untuk bahasa pemrograman Pascal dan lingkungan pengembangan perangkat lunak. Produk ini dikembangkan oleh CodeGear sebagai divisi pengembangan perangkat lunak milik Embarcadero, divisi tersebut sebelumnya adalah milik Borland. Dengan menggunakan Free Pascal yang merupakan proyek opensource, bahasa ini dapat pula digunakan untuk membuat program yang berjalan di sistem operasi Mac OS X dan Windows CE.

Umumnya Delphi lebih banyak digunakan untuk pengembangan aplikasi desktop dan enterprise berbasis database, tapi sebagai perangkat pengembangan yang bersifat general-purpose ia juga mampu dan digunakan dalam berbagai jenis proyek pengembangan software.

Perbedaan fitur yang utama antara Delphi, Kylix dengan IDE-IDE yang lain adalah keberadaan bahasanya (Bahasa pemrograman delphi), VCL/CLX (*Visual Component Library*), Penekanan konektifitas database yang sangat baik, dan banyaknya komponen-komponen pihak ketiga yang mendukungnya.

Adapun sejumlah keuntungan Embarcadero Delphi, antara lain: Komunitas pengguna yang besar pada Usenet maupun web, Dapat mengkompilasi menjadi *single executable* (aplikasi portable), memudahkan distribusi dan meminimalisir masalah yang terkait dengan *versioning*, banyaknya dukungan dari pihak ketiga terhadap VCL (biasanya tersedia berikut source codenya) ataupun tools pendukung lainnya (dokumentasi, *tool debugging*), optimasi kompiler yang cukup cepat, mendukung multiple platform dari source code yang sama dan untuk yang dikelola oleh embarcadero, delphi dapat dijalankan pada *multiflatform* yaitu *windows*, *linux*, *android*, IOS [6] [24].