APLIKASI PROGRAM DINAMIS DALAM ALGORITMA COCKE-YOUNGER -KASAMI (CYK)

Inas Luthfi [1)] – NIM 13506019

1) Jurusan Teknik Informatika ITB,
Jalan Ganesha 10 Bandung Indonesia 40132
email: if16019@students.if.itb.ac.id

ABSTRAK

Algoritma Cocke-Younger-Kasami (CYK) adalah algoritma untuk menentukan apakah suatu untai (*string*) dapat diterima oleh suatu Bahasa-Bebas-Konteks (*Context Free Grammar – CFG)[1].* CFG yang diterima oleh algoritma CYK yang diterangkan dalam makalah ini adalah CFG dalam bentuk norma *Chomsky Normal Form* (CNF). Apabila suatu untai dapat diterima, dipaparkan tahapan algoritma CYK untuk menyelesaikan permasalahan ini. Algoritma yang akan dibahas merupakan metode yang umum digunakan dalam teori otomata dan bahasa formal terutama di bidang desain kompiler bahasa pemrograman.

Proses *parsing* untai dengan algoritma CYK memanfaatkan struktur data sebuah array dua dimensi dan merupakan aplikasi Pemrograman Dinamis (Program Dinamis) karena proses *parsing* memanfaatkan hasil *parsing* sebelumnya untuk memutuskan apakah proses yang sedang berlangsung dapat diterima maupun tidak. Dalam makalah ini akan dibahas sebuah contoh pembentukan array dengan CYK yang merupakan algoritma terdiri dari banyak iterasi kolom-baris, kemudian digambarkan pohon parsing yang dapat dibentuk dari array tersebut.

Kata kunci: Cocke-Younger-Kasami, CYK, CFG, CNF Program Dinamis

1. PENDAHULUAN

1.1 Bahasa Formal

Dalam teori Otomata dan Bahasa Formal sekaligus desain kompiler, terdapat pendekatan untuk memetakan struktur suatu bahasa menjadi tata bahasa yang lebih formal. Sejak tahun 1950-an para ahli bahasa telah mempelajari struktur bahasa natural manusia seperti bahasa Inggris dan memetakan tata bahasanya secara formal. Setalah itu pada tahun 1960-an dimana teknik komputasi dan komputer mulai banyak dikembangkan, muncullah bahasa pemrograman untuk menjembatani antara programmer dengan mesin komputer agar mempermudah programmer membuat sebuah program yang memiliki fungsionalitas tertentu.

Pada dekade tersebut, muncul banyak metode untuk mendesain bahasa pemrograman beserta kompilernya, sehingga pada dekade-dekade tersebut, penelitian tentang bahasa formal banyak dilakukan dan kuliah tentang teori otomata dan bahasa formal di bidang ilmu komputer dianggap sangat penting.

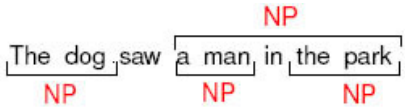
1.2 *Context Free Grammar* dan *Chomsky Normal Form*

Terinspirasi dari bahasa natural manusia, ilmuwan-ilmuwan ilmu komputer yang mengembangkan bahasa pemrograman, turut serta memberikan tata bahasa (pemrograman) secara formal. Tata bahasa ini diciptakan secara bebas-konteks dan disebut CFG (*Context Free Grammar).* Hasilnya, dengan pendekatan formal ini, kompiler suatu bahasa pemrograman dapat dibuat lebih mudah dan menghindari ambiguitas ketika parsing bahasa tersebut. Contoh desain CFG untuk parser semisal : B -> BB | (B) | e untuk mengenali bahasa dengan hanya tanda kurung {'(',')'} sebagai terminal-nya. Proses parsing adalah proses pembacaan untai dalam bahasa sesuai CFG tertentu, proses ini harus mematuhi aturan produksi dalam CFG tersebut.

Secara formal, CFG didefinisikan[2]:

CFG G=(V,T,P,S)
Dimana **V** adalah daftar variabel produksi
T, adalah daftar simbol atau terminalyang dipakai dalam CFG
P, adalah aturan produksi CFG
S, adalah variabel start aturan produksi

CFG dapat 'dinormalkan' dengan pola tersendiri supaya tidak ambigu dan lebih sederhana, meskipun normalisasi CFG kadang membuat aturan produksi menjadi lebih banyak dari sebelumnya. Teknik normalisasi yang digunakan dalam makalah ini adalah CNF (*Chomsky Normal Form).*



Gambar 1. Gambaran parsing bahasa natural (Inggris)

Contoh desain CNF dari bahasa CFG, semisal CFG berikut:

S -> aA | bB (1)
A-> Baalba
B -> bAAlab

CFG (1) tersebut ekivalen dengan CFG dibawah ini, dimana symbol terminal memiliki variabel produksi tersendiri:

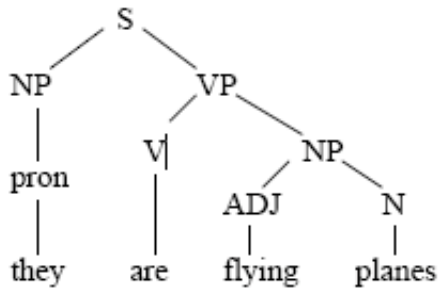
S -> DA | EB (2)
A -> BDD | ED
B -> EAA | DE
D -> a
E -> b

CNF yang dihasilkan dari CFG (2) diatas ialah:

S -> DA | EB (3)
A -> BF | ED
B -> EH | DE
F -> DD
H -> AA
D -> a
E -> b

Setelah terbentuk CFG yang telah dinormalkan secara CNF, dalam implementasi parsing, terdapat algoritma yang berguna untuk menentukan apakah suatu untai 'valid', atau dapat diciptakan dari aturan-aturan CFG yang ada. Salah satu algoritma yang dapat dipakai adalah Algoritma Cocke-Younger-Kasami (CYK). Algoritma ini menyelesaikan masalah analisa kembali sebuah sub-untai yang sama karena seharusnya analisa sub-untai independen terhadap parsing sub-untai yang diparsing setelahnya.

Dengan Program Dinamis, independensi yang diinginkan dapat dicapai ketika parsing. Algoritma CYK termasuk dalam bidang Program Dinamis karena algoritma ini membangun tabel status dua dimensi ketika parsing dimana penentuan parsing selanjutnya diturunkan atau dihasilkan dari parsing sebelumnya, hingga akhir untai. Selain untuk mengetahui validitas untai dalam suatu CFG, algoritma CYK yang dimodifikasi dapat dipergunakan pula untuk membangun pohon parsing.



Gambar 2. Pohon parsing yang terbentuk dari sebuah bahasa natural

2. METODE

2.1 Program Dinamis

Program Dinamis adalah metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan langkah (*step)* atau tahapan (*stage)* sedemikian sehingga solusi dari persoalan dapat dipandang dari serangkaian keputusan yang saling berkaitan [3].

Dalam Program Dinamis, sebuah persoalan dapat diselesaikan dengan metode yang memiliki ciri:

1. Terdapat sejumlah berhingga pilihan yang mungkin
2. Solusi pada setiap tahap dibangun dari hasil solusi tahap sebelumnya
3. Menggunakan persyaratan optimasi dan kendala untuk membatasi sejumlah pilihan yang harus dipertimbangkan setiap tahap
4. Jika solusi total optimal, maka bagian solusi sampai tahap ke-k juga optimal

Algoritma yang membentuk solusi secara bertahap selain Program Dinamis adalah algoritma Greedy, namun hal yang membedakan dengan Program Dinamis adalah daam Program Dinamis rangkaian keputusan yang pernah dihasilkan tidak hanya satu buah seperti layaknya Greedy.[3]

Pendekatan awal algoritma Program Dinamis untuk pengecekan validitas untai dalam suatu CFG pertama secara sistematis mengisi array atau tabel solusi dari permasalahan yang lebih kecil (*memorization)* setelah semua solusi permasalahan seleai diakumulasi dalam tabel, selesaikan permasalahan sebenarnya dari mengkomposisi solusi yang sudah ada.

Dalam hal parsing untai, permasalahan lebih kecil yang dimaksud adalah setiap simbol dalam untai, dan tabel yang digunakan untuk mencatat solusi dinamakan *chart*. *Chart* berisi *constituent* (upapohon) yang telah ditemukan sebelumnya, yang di-indeks sesuai indeks simbol didalam untai[4]. Proses pembentukan *Chart* memiliki algoritma tersendiri, dalam makalah ini, algoritma yang akan dibahas adalah algoritma CYK.

2.2 Algoritma Cocke-Younger-Kasami (CYK)

Algoritma CYK menggunakan tabel dua dimensi untuk menyimpan hasil keputusan permasalahan yang lebih kecil terlebih dahulu. Sisi Program Dinamis dari algoritma ini terletak pada pembangunan array dua dimensi atau tabel saat memparsing sebuah untai, kemudian ketika parsing untai dilakukan dalam iterasi selanjutnya, algoritma ini akan memanfaatkan array atau tabel yang telah dibangun sebelumnya. Dari tabel yang telah terbentuk, untai yang diparsing dapat diketahui apakah valid, dalam artian CFG tersebut dapat memproduksi untai tersebut melalui aturan-aturan yang ada.

Berikut ini adalah persyaratan yang dibentuk dengan mengapikasikan CYK:

- **Input: untai dengan n simbol**
- **Output: valid/ tak valid**
- **Struktur data: tabel n x n**
- **Baris dengan indeks 0 sampai n-1 (atau 1-n dengan modifikasi)**
- **Kolom dengan indeks 1 sampai n**
- **Sel [i,j] simbol yang yang termasuk dalam untai input**

Siapkan tabel n x n dimana n adalah panjang untai yang akan dicek validitasnya, kemudian dalam diagnal tabel tersebut, isi tiap sel dengan tiap simbol dalam untai. Sel yag diisi mulai dari [i,j] awal.

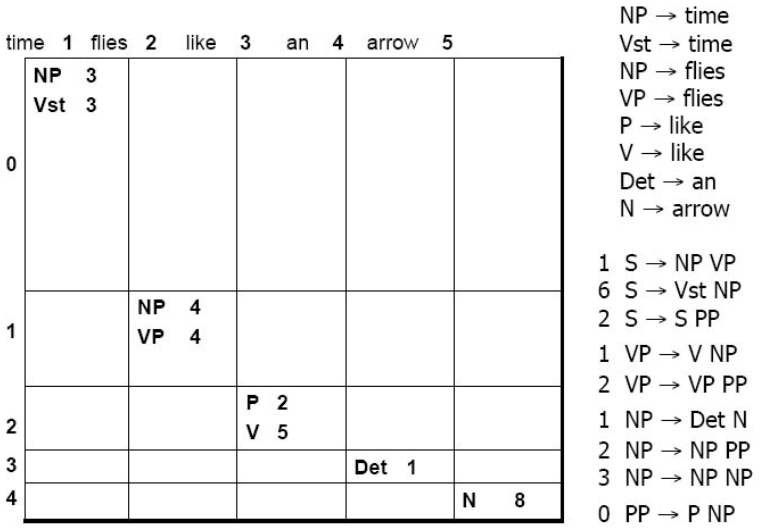
Setelah persyaratan pembangunan algoritma telah dipenuhi maka pseudo-code CYK seperti berikut:

```
for i := 1 to n
        Add to [i-1,i] all (part-of-speech)
categories for the ith word
        for width := 2 to n
         for start := 0 to n-width
          Define end := start + width
           for mid := start+1 to end-1
            for every constituent X in [start,mid]
             for every constituent Y in [mid,end]
              for all ways of combining X and Y (if
any)
                Add the resulting constituent to
[start,end]
```

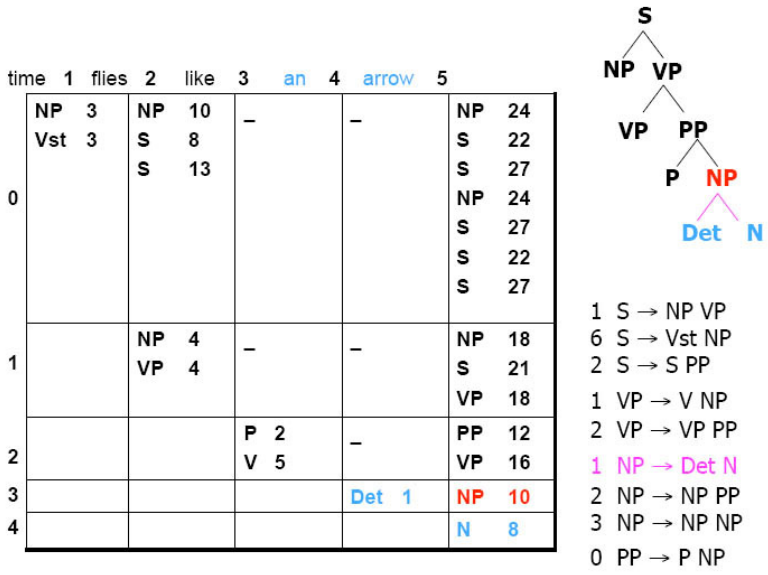
Pseudo-code ini bermaksud untuk mengisi tiap sel setelah diagonal tengah dengan variabel produksi yang mungkin dari CFG yang diberikan oleh persoalan dengan mengecek symbol atas dan bawah (lihat gambar) . Setiap kali variabel produksi dapat dikembalikan, maka simbol dalam untai berkurang satu. Apabila tidak ada hasil produksi, maka sel diberikan penanda khusus, atau dikosongkan.

Apabila untai masih berisi simbol setelah iterasi selesai sampai pada sel batas, maka untai tersebut tidak valid.

Contoh permasalahan yang dipecahkan:



Gambar 3. Algoritma CYK dalam mengecek validitas bahasa natural "time flies like an arrow" – tahap awal



Gambar 4. Algoritma CYK dalam mengecek validitas bahasa natural "time flies like an arrow" – selesai

2.3 Algoritma Cocke-Younger-Kasami (CYK) Dalam Teknik Kompilasi

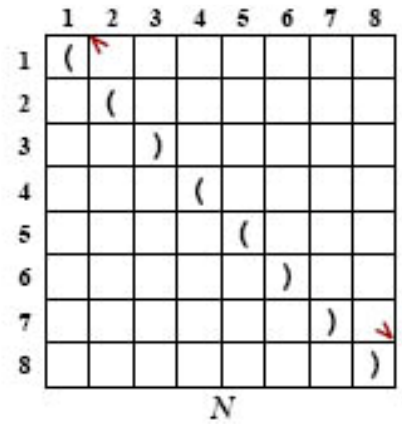
Penerapan algoritma CYK ini dalam teknik kompilasi suatu bahasa tak jauh berbeda, perbedaan yang paling mendasar adalah simbol yang terdapat dalam untai merupakan token-token dalam bahasa pemrograman yang spesifik, sehingga koleksi simbol di CFG yang akan diproses sangat bervariasi. Contohnya dalam bahasa C, terdapat simbol-simbol {'(',')','[',']','{'} dan seterusnya.

Untuk menggambarkan algoritma CYK dalam teknik kompilasi, kita akan membahas satu contoh yang hanya mengecek rekursi simbol '(', dan ')'.

Diberikan CFG G=(V,T,P,S) dimana
V = {S} T = {(,)} dan produknya (1)
P:
S **->** SS
S **->** (S)
S **->** **e**
Konversi ke CNF dahulu:
V = {S} (2)
T = {(,)}
P:
S -> SS
S -> (S1
S1 -> S)
S -> ()

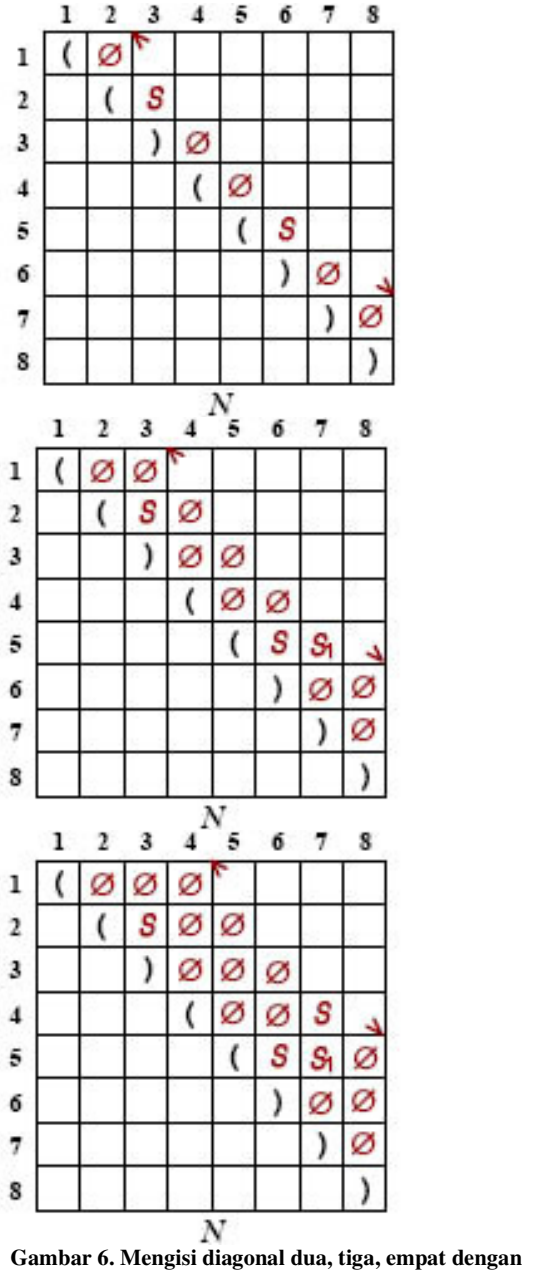
Kita aplikasikan algoritma CYK untuk untai ((()(())), langkah-demi langkah dalam kasus kita (dalam gambar, kolom dan baris dimodifikasi agar mulai dari indeks 1 semua).

Di tahap awal isi diagonal tabel dengan simbol-simbol untai:

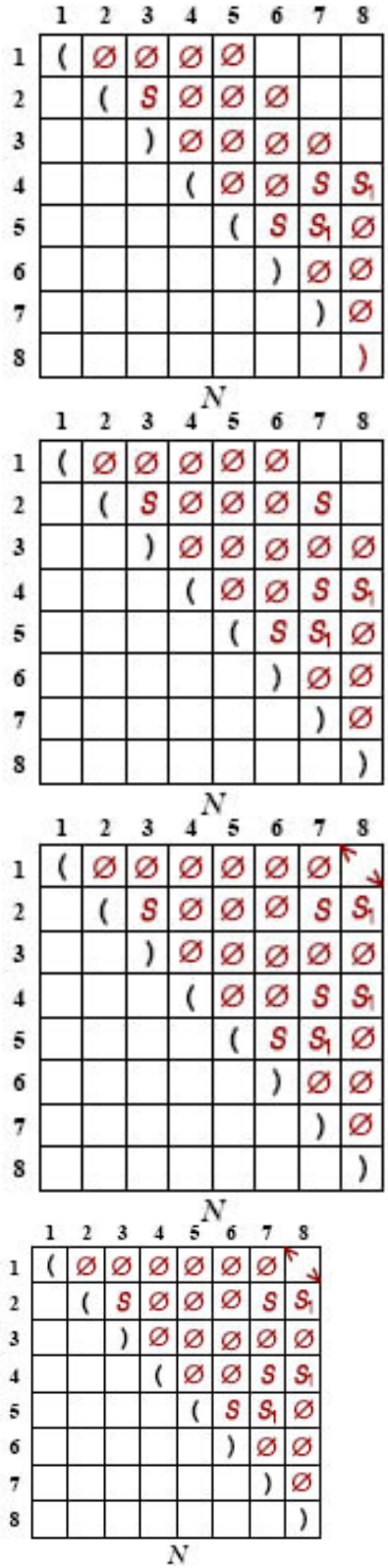


Gambar 5. Mengisi diagonal dengan simbol dari untai

Selanjutnya adalah mengiterasi hingga habis sel paling pojok kanan-atas :



Gambar 6. Mengisi diagonal dua, tiga, empat dengan Hasil Produksi CFG



Gambar7. Mengisi diagonal sisa Hasil Produksi CFG

Hal yang perlu diperhatikan sebelum mengisi sel diagonal dengan variabel poduksi adalah memperhatikan simbol di kolom dan baris yang sama dengan posisi [i,j] sel sekarang. Cek semua kombinasi yang mungkin dengan variabel produksinya. Disinilah peran CNF untuk menghilangkan kemabiguitasan pengambilan kombinasi.

3. KESIMPULAN

CFG adalah tata bahasa yang digunakan untuk formalisasi suatu bahasa baik bahasa natural maupun bahasa pemrograman. Untuk menormalkan suatu CFG dipergunakan metode CNF. CFG dan CNF dipelajari dalam bidang teori otomata dan bahasa formal secara lebih mendalam.

Permasalah pengecekan 'validitas' sebuah untai simbol dalam CFG tertentu dapat diselesaikan secara Program Dinamis. Salah satu algortima yang dapat digunakan untuk permasalahan ini adalah algoritma Cocke-Younger-Kasami (CYK) yag merupakah implementasi pengembangan lebih lanjut dari Program Dinamis. CYK merupakan penerapan Program Dinamis yang cukup mudah diterapkan meski memiliki kompleksitas kasus terburuk $O(n3)$, dimana n adalah panjang untai yang diparsing.

Sisi Program Dinamis dari algoritma ini terletak pada pembangunan array dua dimensi atau tabel saat memparsing sebuah untai. Dari tabel yang telah terbentuk, untai yang diparsing dapat diketahui apakah valid, dalam artian CFG tersebut dapat memproduksi untai tersebut melalui aturan-aturan yang ada.

CYK merupakan algoritma yang cukup efisien dalam hal mengenali CFG apapun dan merupakan algoritma dasar yang diperkenalkan dalam pemrograman kompiler suatu bahasa pemrograman.

REFERENSI

[1] Andrew McCallum, "Chart Parsing", Lecture of University of Massachusetts Amherst: Introduction to Natural Language Processing. 2007

[2] Lecture of CYK algorithm http://www.ecst.csuchico.edu/~juliano/Computing/ Tanggal akses : 18 Mei 2008 pukul 12:00

[3] Munir, Rinaldi, *Matematika Diskrit,* Program Studi Informatika, Institut Teknologi Bandung. 2007.

[4] Nitin Gupta, "Parsing Natural Languages", Paper from Department of Computer Science and Engg, IIT Bombay , 2005.

[5] English Wikipedia 2008 http://en.wikipedia.org/wiki/CYK Tanggal akses: 19 Mei 2008 pukul 13:00.

[6] Bonnie Webber,"Chart Parsing: The CYK Algorithm", Lecture of School of Informatics University of Edinburgh, 2007