

LAPORAN DESKRIPSI CFG dan RANCANGAN FINITE AUTOMATA

LAPORAN

Diajukan untuk memenuhi tugas pada mata kuliah Teori Bahasa dan Automata



Kelas : IF- 43 - 06

Anggota :

Faiz Rofi Hencya (1301190230)

Fadhlan Mochamad Daffa R. (1301194172)

Firra Millaty Suryadi (1301194314)

**PROGRAM STUDI S1 INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
BANDUNG
2021**

DAFTAR ISI

DAFTAR ISI	1
BAB 1 PENDAHULUAN	2
BAB 2 DASAR TEORI	2
2.1 Context Free Grammar (CFG)	2
2.2 Finite Automata.....	2
BAB 3 IMPLEMENTASI.....	3
3.1 Context Free Grammar (CFG).....	3
3.2 Finite Automata.....	4
3.3 Program Lexical Analyzer	5
BAB 4 ANALISIS	10
4.1 Valid Lexical Analyzer.....	10
4.2 Tidak valid lexical analyzer.....	10

BAB 1 PENDAHULUAN

Pembuatan laporan ini dimaksudkan untuk memenuhi tugas besar pada mata kuliah Teori Bahasa dan Automata. Pada laporan ini kami akan mendefinisikan Context Free Grammar (CFG) yang merepresentasikan bahasa sederhana di dalamnya. Kemudian mengimplementasikan CFG tersebut ke dalam rancangan Finite Automata dan setelahnya membuat program lexical analyzer. Program lexical analyzer ini nantinya akan mengidentifikasi apakah sebuah lexical/ token/ kata/ valid sesuai simbol terminal yang telah di definisikan.

Untuk bahasa sederhana yang di gunakan adalah bahasa Palembang. Bahasa Palembang, bahasa Melayu Palembang atau Bahasa Musi adalah sebuah bahasa atau kelompok dialek yang dipertuturkan oleh masyarakat di sebagian wilayah Sumatra Selatan dengan penutur asli berjumlah sekitar 3,1 juta orang. Sebagai bagian dari rumpun bahasa Melayu, bahasa ini berhubungan dekat dengan bahasa Jambi, bahasa Minangkabau, bahasa Banjar, serta bahasa Indonesia. Di antara beragam bahasa yang dipertuturkan di Sumatra Selatan, bahasa Palembang (dialek kota) juga berfungsi sebagai bahasa pemersatu atau *lingua franca*. Bahasa Palembang merupakan bahasa aglutinatif seperti banyak bahasa Austronesia yang lain.

BAB 2 DASAR TEORI

2.1 Context Free Grammar (CFG)

CFG atau Context Free Grammar adalah suatu notasi formal untuk menyatakan definisi rekursif dari suatu bahasa. Grammar terdiri atas satu atau lebih variable yang mewakili kelas-kelas untai dimana terdapat aturan yang menyatakan bagaimana tiap untai dibangun. Aturan ini akan menghasilkan alfabet, untai lain yang telah diketahui atau keduanya.

2.2 Finite Automata

Finite automata merupakan sebuah model komputasi dengan jumlah yang sangat terbatas (Model komputasional yang paling sederhana). Finite automat juga disebut sebagai Finite State Machine (FSM). Finite Automata biasa digunakan pada aplikasi yang membutuhkan teknik pengenalan pola. Finite automata juga disebut sebagai mesin abstrak yang terdiri dari finite number of state. Dimana finite automata harus memiliki initial state dan minimal satu accepted state. Mesin akan menerima input stream berupa symbol/ alphabet yang datang secara sekuensial. Mesin juga akan berubah dari state satu ke state lain berdasarkan simbol input dan current state.

BAB 3 IMPLEMENTASI

3.1 Context Free Grammar (CFG)

Berikut bentuk CFG dengan merepresentasikan bahasa Palembang didalamnya

Grammar :

$\langle S \rangle ::= \langle \text{noun} \rangle \langle \text{verb} \rangle \langle \text{noun} \rangle$

$\langle \text{noun} \rangle ::= \text{mak} \mid \text{abah} \mid \text{mamang} \mid \text{televisi} \mid \text{sungi} \mid \text{bakso} \mid \text{telok}$

$\langle \text{verb} \rangle ::= \text{meli} \mid \text{majo} \mid \text{nyingok} \mid \text{ngenjuk}$

Penjelasan :

- Simbol non-terminal:

S (starting symbol), noun, verb

- Simbol terminal :

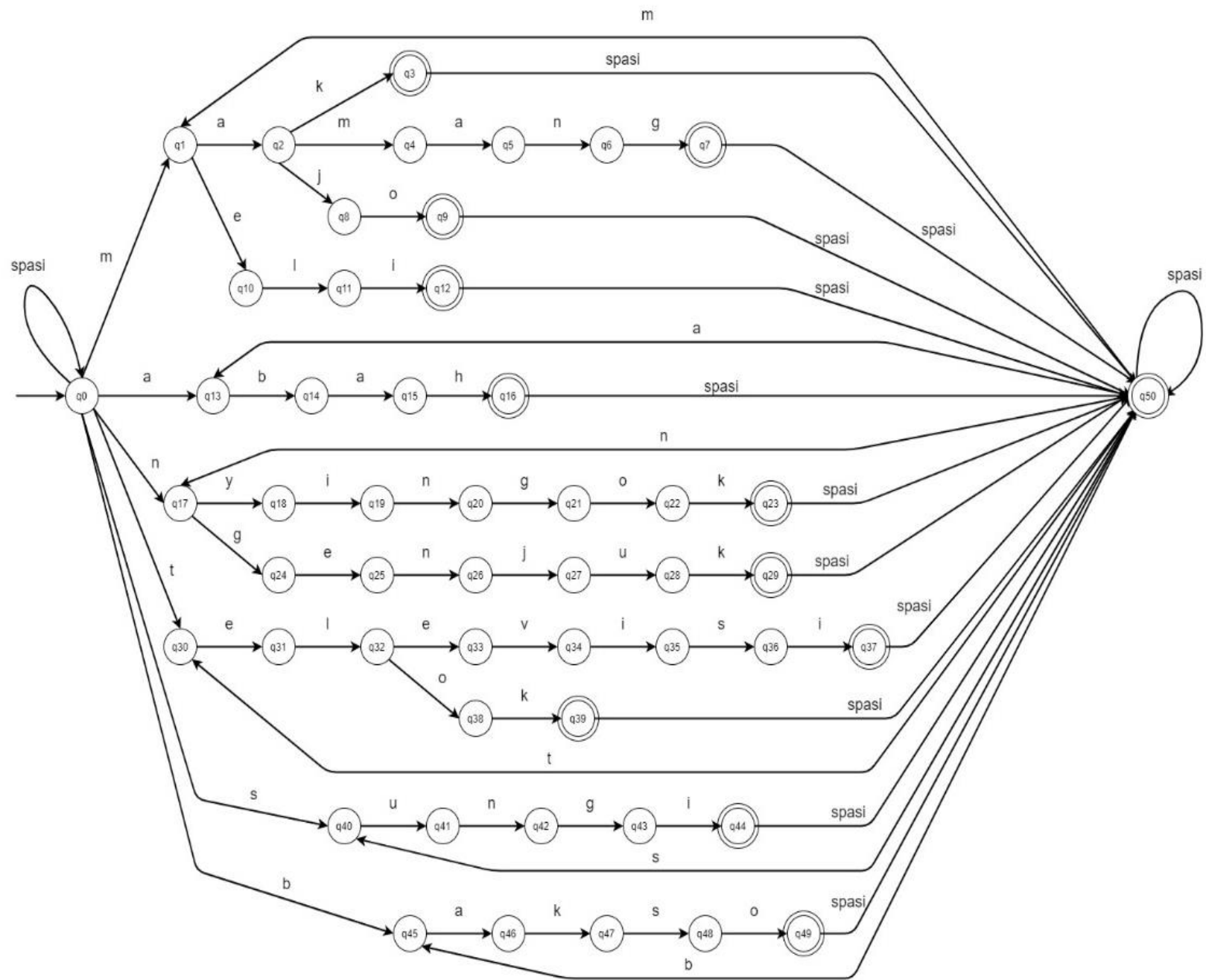
Mak, abah, mamang, televisi, sungi, bakso, telok, meli, majo, nyingok, ngenjuk

Untuk arti dari Bahasa Palembang yang digunakan adalah sebagai berikut:

- | | |
|------------|------------|
| - mak | = ibu |
| - abah | = ayah |
| - mamang | = paman |
| - televisi | = televisi |
| - sungi | = sungai |
| - bakso | = bakso |
| - telok | = telur |
| - meli | = membeli |
| - majo | = makan |
| - nyingok | = melihat |
| - ngenjuk | = memberi |

3.2 Finite Automata

Berikut bentuk Finite Automata berdasarkan CFG yang telah dibuat:



3.3 Program Lexical Analyzer

Berikut program lexical analyzer yang dibuat berdasarkan CFG dan Finite Automata yang telah dibuat :

No	Program	Keterangan
1	<pre># input input_sentence = input("Masukan kalimat yang ingin anda cek :") string_check = input_sentence.lower() + "#"</pre>	Menerima inputan dari user yang disimpan dengan bentuk lowercase dan ditambahkan dengan “#”
2	<pre># initialization alphabet_list = list(string.ascii_lowercase) state_list = ['q0', 'q1', 'q2', 'q3', 'q4', 'q5', 'q6', 'q7', 'q8', 'q9', 'q10', 'q11', 'q12', 'q13', 'q14', 'q15', 'q16', 'q17', 'q18', 'q19', 'q20', 'q21', 'q22', 'q23', 'q24', 'q25', 'q26', 'q27', 'q28', 'q29', 'q30', 'q31', 'q32', 'q33', 'q34', 'q35', 'q36', 'q37', 'q38', 'q39', 'q40', 'q41', 'q42', 'q43', 'q44', 'q45', 'q46', 'q47', 'q48', 'q49', 'q50'] transition_table = {} for state in state_list: for alphabet in alphabet_list: transition_table[(state,alphabet)] = 'error' transition_table[(state, '#')] = 'error' transition_table[(state, ' ')] = 'error'</pre>	Inisialisasi dari alphabet list, state list dan transition table. Dan juga melakukan inisialisasi untuk semua transition table menjadi “error”
3	<pre># space before input string transition_table['q0', ' '] = 'q0'</pre>	Transition table jika terjadi input spasi sebelum menginputkan kalimat
4	<pre># transition for new token transition_table[('q50', 'm')] = 'q1' transition_table[('q50', 'a')] = 'q13' transition_table[('q50', 'n')] = 'q17' transition_table[('q50', 't')] = 'q30' transition_table[('q50', 's')] = 'q40' transition_table[('q50', 'b')] = 'q45'</pre>	Transition table untuk kalimat baru selanjutnya setelah berhasil mengidentifikasi sebuah kalimat

5	<pre># update the transition table for the following token: mak transition_table[('q0', 'm')] = 'q1' transition_table[('q1', 'a')] = 'q2' transition_table[('q2', 'k')] = 'q3' transition_table[('q3', ' ')] = 'q50' transition_table[('q3', '#')] = 'accept' transition_table[('q50', ' ')] = 'q50' transition_table[('q50', '#')] = 'accept'</pre>	Transition table untuk mengenali kalimat “mak”
6	<pre># update the transition table for the following token: majo transition_table[('q0', 'm')] = 'q1' transition_table[('q1', 'a')] = 'q2' transition_table[('q2', 'j')] = 'q8' transition_table[('q8', 'o')] = 'q9' transition_table[('q9', ' ')] = 'q50' transition_table[('q9', '#')] = 'accept' transition_table[('q50', ' ')] = 'q50' transition_table[('q50', '#')] = 'accept'</pre>	Transition table untuk mengenali kalimat “majo”
7	<pre># update the transition table for the following token: mamang transition_table[('q0', 'm')] = 'q1' transition_table[('q1', 'a')] = 'q2' transition_table[('q2', 'm')] = 'q4' transition_table[('q4', 'a')] = 'q5' transition_table[('q5', 'n')] = 'q6' transition_table[('q6', 'g')] = 'q7' transition_table[('q7', ' ')] = 'q50' transition_table[('q7', '#')] = 'accept' transition_table[('q50', ' ')] = 'q50' transition_table[('q50', '#')] = 'accept'</pre>	Transition table untuk mengenali kalimat “mamang”
8	<pre># update the transition table for the following token: meli transition_table[('q0', 'm')] = 'q1' transition_table[('q1', 'e')] = 'q10' transition_table[('q10', 'l')] = 'q11' transition_table[('q11', 'i')] = 'q12' transition_table[('q12', ' ')] = 'q50' transition_table[('q12', '#')] = 'accept' transition_table[('q50', ' ')] = 'q50' transition_table[('q50', '#')] = 'accept'</pre>	Transition table untuk mengenali kalimat “meli”
9	<pre># update the transition table for the following token: abah transition_table[('q0', 'a')] = 'q13' transition_table[('q13', 'b')] = 'q14' transition_table[('q14', 'a')] = 'q15' transition_table[('q15', 'h')] = 'q16' transition_table[('q16', ' ')] = 'q50' transition_table[('q16', '#')] = 'accept' transition_table[('q50', ' ')] = 'q50' transition_table[('q50', '#')] = 'accept'</pre>	Transition table untuk mengenali kalimat “abah”

10	<pre># update the transition table for the following token: nyingok transition_table[('q0', 'n')] = 'q17' transition_table[('q17', 'y')] = 'q18' transition_table[('q18', 'i')] = 'q19' transition_table[('q19', 'n')] = 'q20' transition_table[('q20', 'g')] = 'q21' transition_table[('q21', 'o')] = 'q22' transition_table[('q22', 'k')] = 'q23' transition_table[('q23', ' ')] = 'q50' transition_table[('q23', '#')] = 'accept' transition_table[('q50', ' ')] = 'q50' transition_table[('q50', '#')] = 'accept'</pre>	Transition table untuk mengenali kalimat “nyingok”
11	<pre># update the transition table for the following token: ngenjuk transition_table[('q0', 'n')] = 'q17' transition_table[('q17', 'g')] = 'q24' transition_table[('q24', 'e')] = 'q25' transition_table[('q25', 'n')] = 'q26' transition_table[('q26', 'j')] = 'q27' transition_table[('q27', 'u')] = 'q28' transition_table[('q28', 'k')] = 'q29' transition_table[('q29', ' ')] = 'q50' transition_table[('q29', '#')] = 'accept' transition_table[('q50', ' ')] = 'q50' transition_table[('q50', '#')] = 'accept'</pre>	Transition table untuk mengenali kalimat “ngenjuk”
12	<pre># update the transition table for the following token: televisi transition_table[('q0', 't')] = 'q30' transition_table[('q30', 'e')] = 'q31' transition_table[('q31', 'l')] = 'q32' transition_table[('q32', 'e')] = 'q33' transition_table[('q33', 'v')] = 'q34' transition_table[('q34', 'i')] = 'q35' transition_table[('q35', 's')] = 'q36' transition_table[('q36', 'i')] = 'q37' transition_table[('q37', ' ')] = 'q50' transition_table[('q37', '#')] = 'accept' transition_table[('q50', ' ')] = 'q50' transition_table[('q50', '#')] = 'accept'</pre>	Transition table untuk mengenali kalimat “televisi”
13	<pre># update the transition table for the following token: telok transition_table[('q0', 't')] = 'q30' transition_table[('q30', 'e')] = 'q31' transition_table[('q31', 'l')] = 'q32' transition_table[('q32', 'o')] = 'q38' transition_table[('q38', 'k')] = 'q39' transition_table[('q39', ' ')] = 'q50' transition_table[('q39', '#')] = 'accept' transition_table[('q50', ' ')] = 'q50' transition_table[('q50', '#')] = 'accept'</pre>	Transition table untuk mengenali kalimat “telok”

14	<pre># update the transition table for the following token: sungi transition_table[('q0', 's')] = 'q40' transition_table[('q40', 'u')] = 'q41' transition_table[('q41', 'n')] = 'q42' transition_table[('q42', 'g')] = 'q43' transition_table[('q43', 'i')] = 'q44' transition_table[('q44', ' ')] = 'q50' transition_table[('q44', '#')] = 'accept' transition_table[('q50', ' ')] = 'q50' transition_table[('q50', '#')] = 'accept'</pre>	Transition table untuk mengenali kalimat “sungi”
15	<pre># update the transition table for the following token: bakso transition_table[('q0', 'b')] = 'q45' transition_table[('q45', 'a')] = 'q46' transition_table[('q46', 'k')] = 'q47' transition_table[('q47', 's')] = 'q48' transition_table[('q48', 'o')] = 'q49' transition_table[('q49', ' ')] = 'q50' transition_table[('q49', '#')] = 'accept' transition_table[('q50', ' ')] = 'q50' transition_table[('q50', '#')] = 'accept'</pre>	Transition table untuk mengenali kalimat “bakso”
16	<pre># lexical analysis idx_char = 0 state = 'q0' current_token = '' while state != 'accept': current_char = string_check[idx_char] current_token += current_char state = transition_table[(state,current_char)] if state == 'q3': print('current token: ',current_token,' valid') current_token = '' if state == 'q7': print('current token: ',current_token,' valid') current_token = '' if state == 'q9': print('current token: ',current_token,' valid') current_token = '' if state == 'q12': print('current token: ',current_token,' valid') current_token = '' if state == 'q16': print('current token: ',current_token,' valid') current_token = '' if state == 'q23': print('current token: ',current_token,' valid') current_token = '' if state == 'q29': print('current token: ',current_token,' valid') current_token = '' if state == 'q37': print('current token: ',current_token,' valid') current_token = ''</pre>	Main code dari lexical analysis dimana akan melakukan perulangan selama state nya adalah “accept” dan akan mengeluarkan “error” jika state adalah “error”

	<pre> if state == 'q39': print('current token: ', current_token, ' valid') current_token = '' if state == 'q44': print('current token: ', current_token, ' valid') current_token = '' if state == 'q49': print('current token: ', current_token, ' valid') current_token = '' if state == 'error': print('error') break; idx_char = idx_char + 1 </pre>	
	<pre> # conclusion if state == 'accept': print('semua token di input: ', input_sentence, ' valid') </pre>	<p>Kesimpulan dari hasil lexical analysis dimana akan mengeluarkan hasil jika state tetap “accept”</p>

Berikut merupakan petunjuk untuk menjalankan program lexical analyzer :

1. User menjalankan program di cmd/VSC
2. User menginputkan kalimat yang ingin di cek
3. Jika kalimat terdeteksi(ada) di dalam bahasa Palembang yang disediakan maka program akan mengeluarkan pesan berupa semua kalimat valid yang diinputkan oleh user.
4. Dan jika kalimat tidak terdeteksi(tidak ada) di dalam bahasa Palembang yang disediakan Maka program akan mengeluarkan pesan error

BAB 4 ANALISIS

4.1 Valid Lexical Analyzer

Program akan menghasilkan output yang valid jika inputan atau kata yang dimasukkan sesuai dengan daftar kata atau token yang telah ada didalam program seperti contoh jika kita menginputkan kata “mak abah televisi sungi bakso telok mamang meli majo nyingok ngenjuk” maka akan menghasilkan output valid pada semua token atau inputan yang dimasukkan.

Dapat dilihat seperti gambar dibawah :

```
E:\materi kulyah\SEMESTER 4\Teori Bahasa Automata\Tubes>C:/Users/user/AppData/Local/Microsoft/WindowsApps/python3.9.exe "e:/materi kulyah/SEMESTER 4/Teori Bahasa Automata/Tubes/lexical_analyzer.py"
Masukan kalimat yang ingin anda cek :mak abah televisi sungi bakso telok mamang meli majo nyingok ngenjuk
current token: mak valid
current token: abah valid
current token: televisi valid
current token: sungi valid
current token: bakso valid
current token: telok valid
current token: mamang valid
current token: meli valid
current token: majo valid
current token: nyingok valid
current token: ngenjuk valid
semua token di input: mak abah televisi sungi bakso telok mamang meli majo nyingok ngenjuk valid
```

4.2 Tidak valid lexical analyzer

Program akan menghasilkan output yang **tidak valid** jika inputan atau kata yang dimasukkan tidak sesuai dengan daftar kata atau token yang telah ada didalam program seperti contoh jika kita menginputkan kata “kakak”, “ibuk”, “ayah”, “mengambil”, dan “sungai” maka akan menghasilkan output error pada semua token atau inputan yang dimasukkan.

Dapat dilihat seperti gambar di bawah :

```
E:\materi kulyah\SEMESTER 4\Teori Bahasa Automata\Tubes>C:/Users/user/AppData/Local/Micros
oft/WindowsApps/python3.9.exe "e:/materi kulyah/SEMESTER 4/Teori Bahasa Automata/Tubes/lex
ical_analyzer.py"
Masukan kalimat yang ingin anda cek :kagak
error

E:\materi kulyah\SEMESTER 4\Teori Bahasa Automata\Tubes>C:/Users/user/AppData/Local/Micros
oft/WindowsApps/python3.9.exe "e:/materi kulyah/SEMESTER 4/Teori Bahasa Automata/Tubes/lex
ical_analyzer.py"
Masukan kalimat yang ingin anda cek :ibuk
error

E:\materi kulyah\SEMESTER 4\Teori Bahasa Automata\Tubes>C:/Users/user/AppData/Local/Micros
oft/WindowsApps/python3.9.exe "e:/materi kulyah/SEMESTER 4/Teori Bahasa Automata/Tubes/lex
ical_analyzer.py"
Masukan kalimat yang ingin anda cek :ayah
error

E:\materi kulyah\SEMESTER 4\Teori Bahasa Automata\Tubes>C:/Users/user/AppData/Local/Micros
oft/WindowsApps/python3.9.exe "e:/materi kulyah/SEMESTER 4/Teori Bahasa Automata/Tubes/lex
ical_analyzer.py"
Masukan kalimat yang ingin anda cek :mengambil
error

E:\materi kulyah\SEMESTER 4\Teori Bahasa Automata\Tubes>C:/Users/user/AppData/Local/Micros
oft/WindowsApps/python3.9.exe "e:/materi kulyah/SEMESTER 4/Teori Bahasa Automata/Tubes/lex
ical_analyzer.py"
Masukan kalimat yang ingin anda cek :sungai
error
```