LAPORAN TUGAS BESAR

IF2124 Teori Bahasa Formal dan Automata

HTML Checker dengan Pushdown Automata (PDA)

Dipersiapkan oleh:

Kelompok “Ngangongangong”:
Aurelius Justin Philo Fanjaya (13522020)
Dhidit Abdi Aziz (13522040)

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

DAFTAR ISI

1 Landasan Teori..... 2
2 Hasil Pushdown Automata (PDA)..... 4
2.1 Implementasi PDA pada python..... 4
3 Implementasi dan Pengujian..... 12
3.1 Alur Program..... 12
3.2 Struktur Data Program..... 12
3.3 Fungsi dan Prosedur..... 13
3.4 Tokenisasi..... 14
3.5 Antarmuka..... 17
3.6 Analisis Pengujian..... 17
4 Daftar Pustaka..... 22
5 Lampiran..... 22

1 Landasan Teori

HTML (Hypertext Markup Language) adalah bahasa markup yang digunakan untuk membuat struktur dan tampilan konten web. HTML adalah salah satu bahasa utama yang digunakan dalam pengembangan web dan digunakan untuk menggambarkan bagaimana elemen-elemen konten, seperti teks, gambar, tautan, dan media, akan ditampilkan di browser web. Setiap dokumen HTML dimulai dengan elemen <html>, lalu diikuti dengan <head> (untuk metadata dan tautan ke file eksternal) dan <body> (untuk konten yang akan ditampilkan)

HTML menggunakan elemen-elemen *(tags)* untuk mengelompokkan dan mengatur konten. Contohnya, <p> digunakan untuk paragraf teks, <h1> hingga <h6> digunakan untuk judul, <a> untuk tautan, <img> untuk gambar, dan sebagainya. Elemen HTML sering memiliki atribut yang memberikan informasi tambahan tentang elemen tersebut. Contohnya adalah atribut src untuk gambar, href untuk tautan, dan class untuk memberikan elemen kelas CSS.

Sama seperti bahasa pada umumnya, HTML juga memiliki sintaks tersendiri dalam penulisannya yang dapat menimbulkan error jika tidak dipenuhi. Meskipun web browser modern seperti Chrome dan Firefox cenderung tidak menghiraukan error pada HTML memastikan bahwa HTML benar dan terbentuk dengan baik masih penting untuk beberapa alasan seperti *Search Engine Optimization (SEO),* aksesibilitas, *maintenance* yang lebih baik, kecepatan render, dan profesionalisme.

Dibutuhkan sebuah program pendeteksi *error* untuk HTML. Oleh sebab itu, kami membuat sebuah program yang dapat memeriksa kebenaran HTML dari segi nama *tag* yang digunakan serta *attribute* yang dimilikinya. Pada tugas pemrograman ini, kami menggunakan konsep Pushdown Automata (PDA) dalam mencapai hal tersebut yang diimplementasikan dalam bahasa Python**.**

Push Down Automata (PDA) merupakan mesin otomata dari bahasa bebas konteks. PDA digambarkan sebagai tempat penyimpanan tidak terbatas berupa stack/tumpukan. Stack adalah kumpulan elemen sejenis yang disimpan atau disusun dengan cara ditumpuk. Prinsip stack adalah Last In First Out (LIFO) yaitu elemen yang masuk dahulu keluar akhir. PDA adalah pasangan 7 tuple yaitu himpunan n state; alfabet input; alfabet/simbol stack; state awal; simbol awal stack; himpunan state penerima; dan fungsi transisi. Terdapat sedikit perbedaan antara PDA dengan finite automata. Pertama, push down automata menggunakan bahasa konteks bebas sedangkan finite automata menggunakan bahasa reguler. Terakhir, pushdown automata mempunyai tambahan stack untuk menyimpan query yang panjang sedangkan finite automata tidak punya.

Prinsip yang dimiliki PDA khususnya keberadaan stack sebagai tempat penyimpanan sementara sangatlah bermanfaat dalam pengimplementasian program kami. Pengecekan tag per tag HTML dapat dilakukan dengan membandingkan simbol yang sedang dibaca dengan simbol yang tersimpan pada stack. Dengan demikian, dapat dibuat beragam state untuk merepresentasikan setiap kondisi yang mungkin terjadi sehingga program kami mampu mengcover kasus pengecekan yang sangat luas.

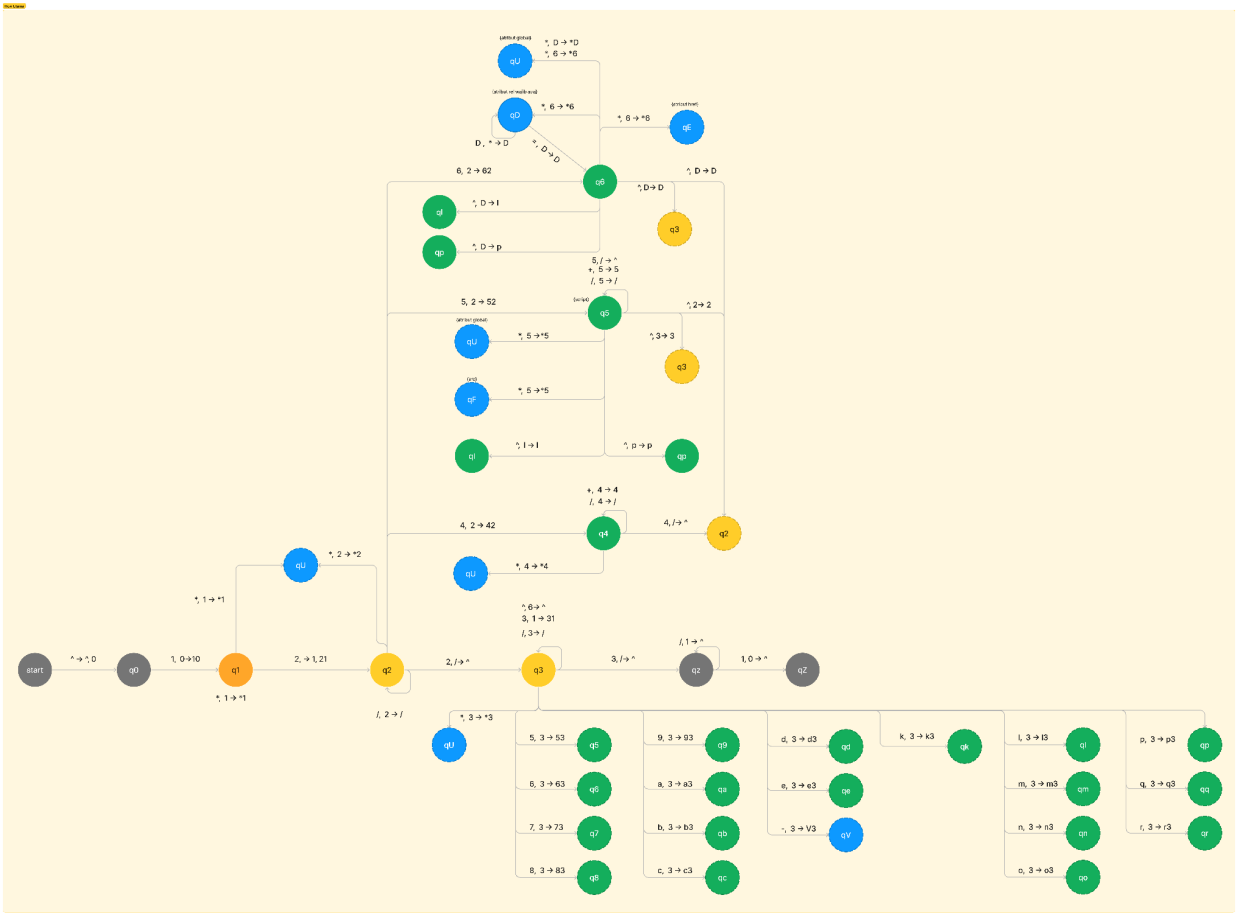
2 Hasil Pushdown Automata (PDA)

2.1 Implementasi PDA pada python

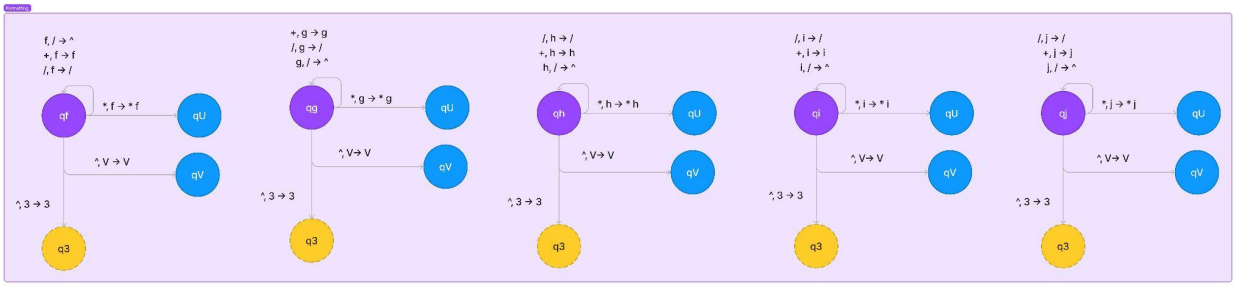
PDA yang kami implementasikan pada bahasa python menggunakan konsep rekursif untuk memanfaatkan kemampuan *backtracking* pada PDA. PDA menerima input config, yaitu konfigurasi yang berisi *total states, input symbols, stack symbols, starting state, starting stack, accepting state,* dan *productions.* Selain itu, PDA juga menerima string yang akan dicek diterima atau tidak oleh PDA dengan konfigurasi config tadi.

Basis dari PDA ini adalah jika panjang input = 0 (input habis terbaca) dan state terdapat dalam accepting_state, maka input diterima (return True). Jika tidak, PDA akan melakukan rekursif pada setiap kemungkinan next state sesuai dengan productions pada config. Khusus untuk implementasi PDA untuk menyelesaikan bonus, yaitu PDABonus, PDA juga menerima dan melakukan return kondisi ketika rekursi dengan depth terdalam / sisa length input paling pendek.

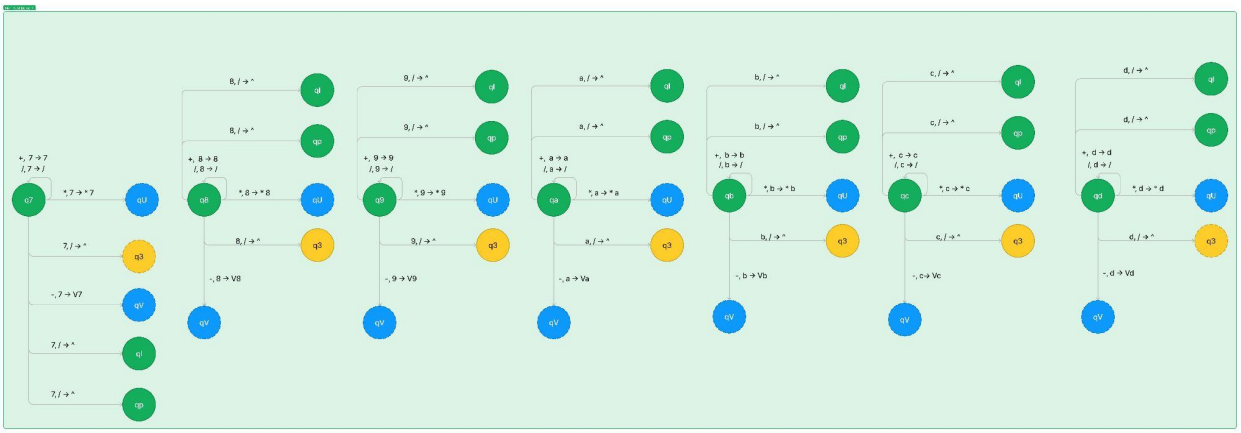
2.2 PDA State Diagram



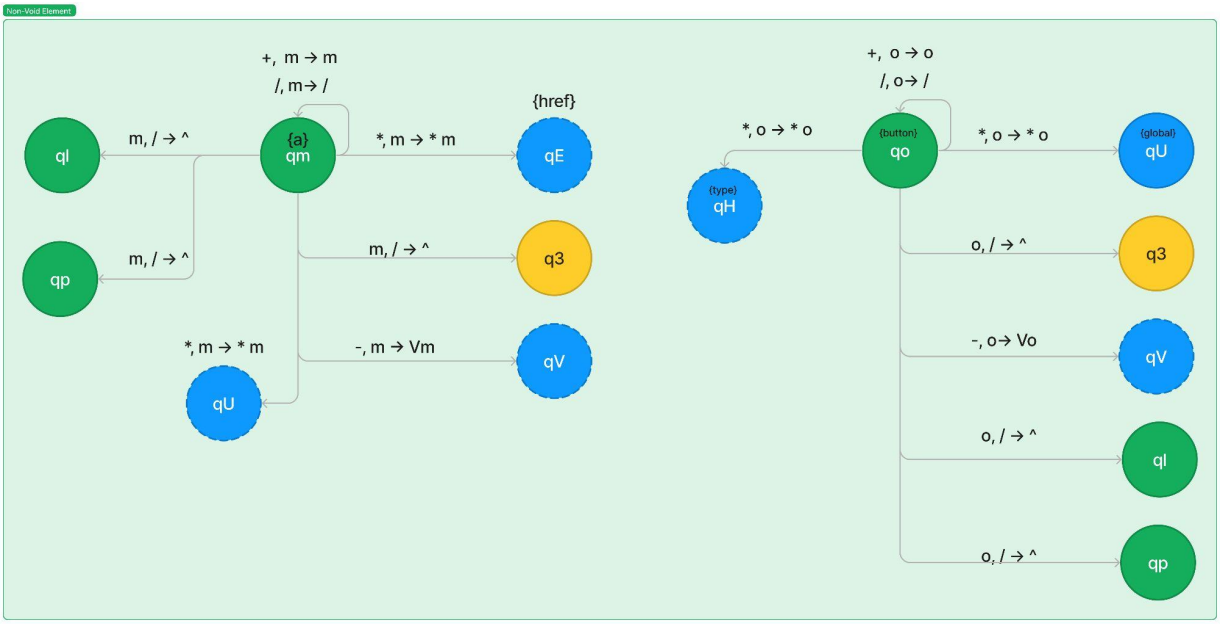
Gambar 2.1 Diagram Utama PDA



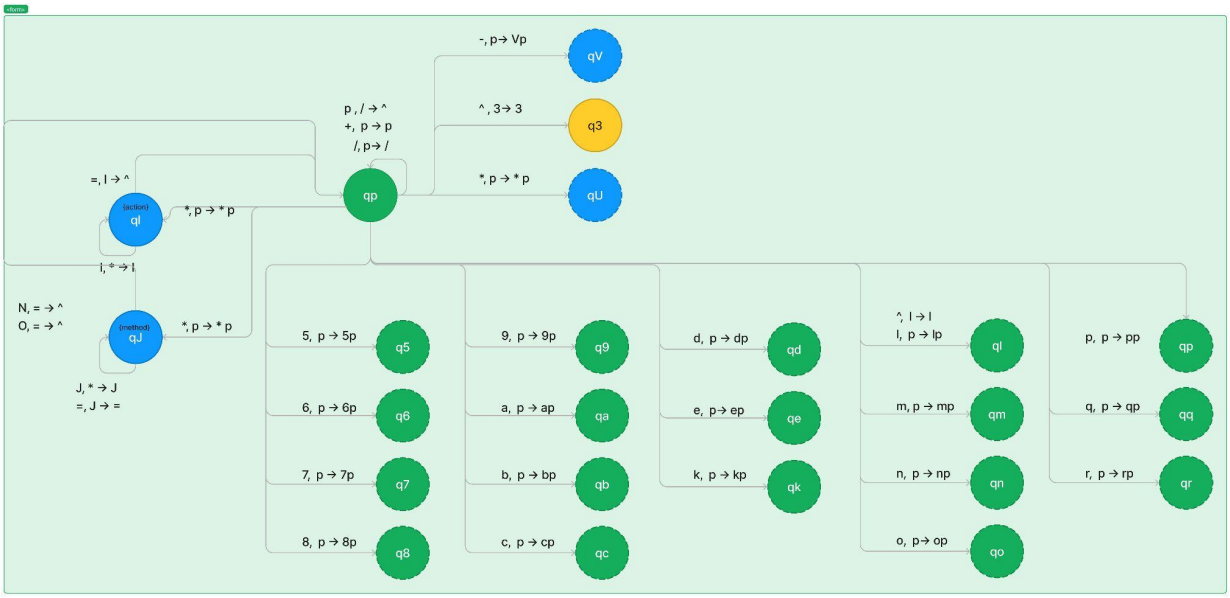
Gambar 2.2 Diagram Formatting Element



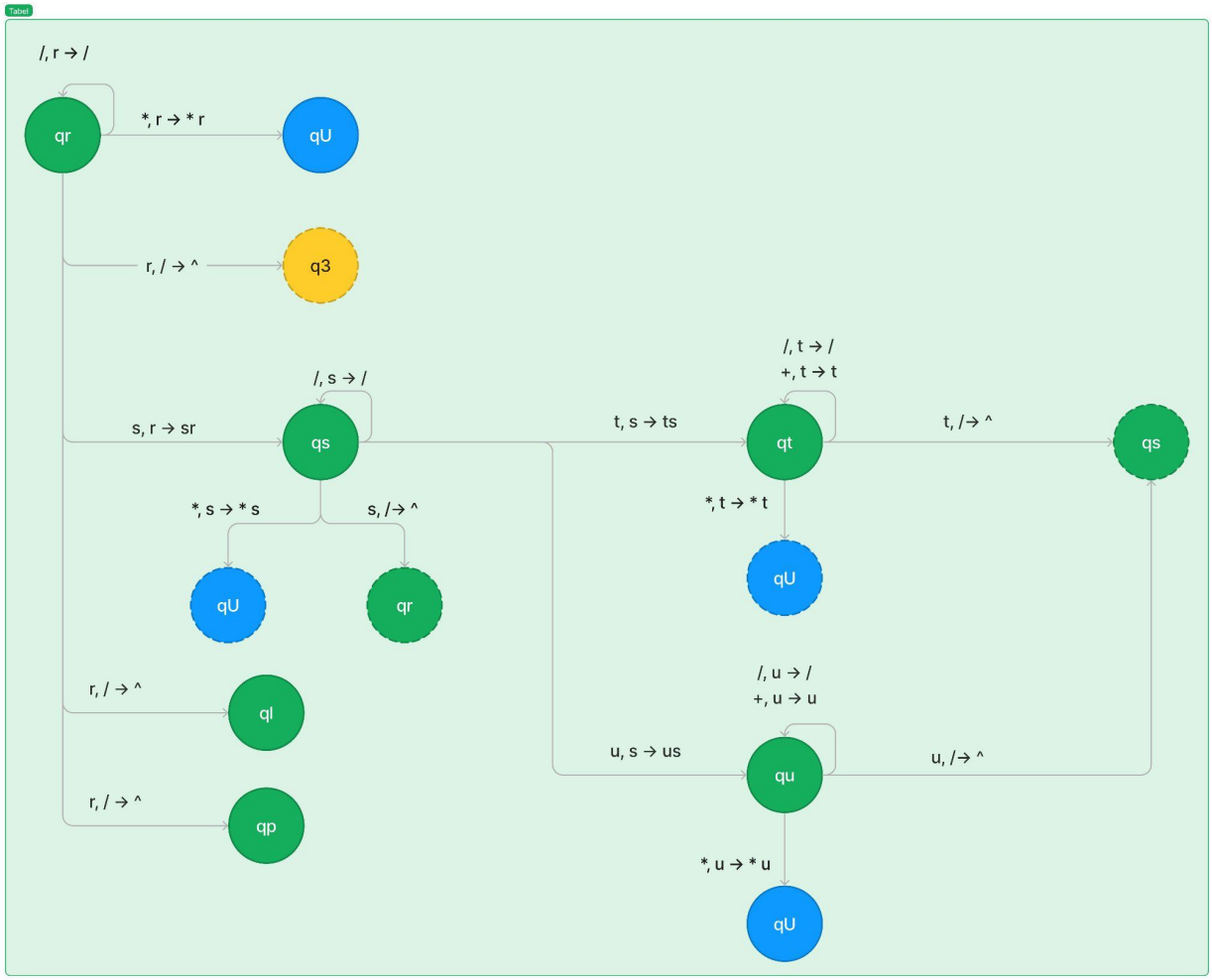
Gambar 2.3 Non-Void Element Tag



Gambar 2.4 Non-Void Element Tag

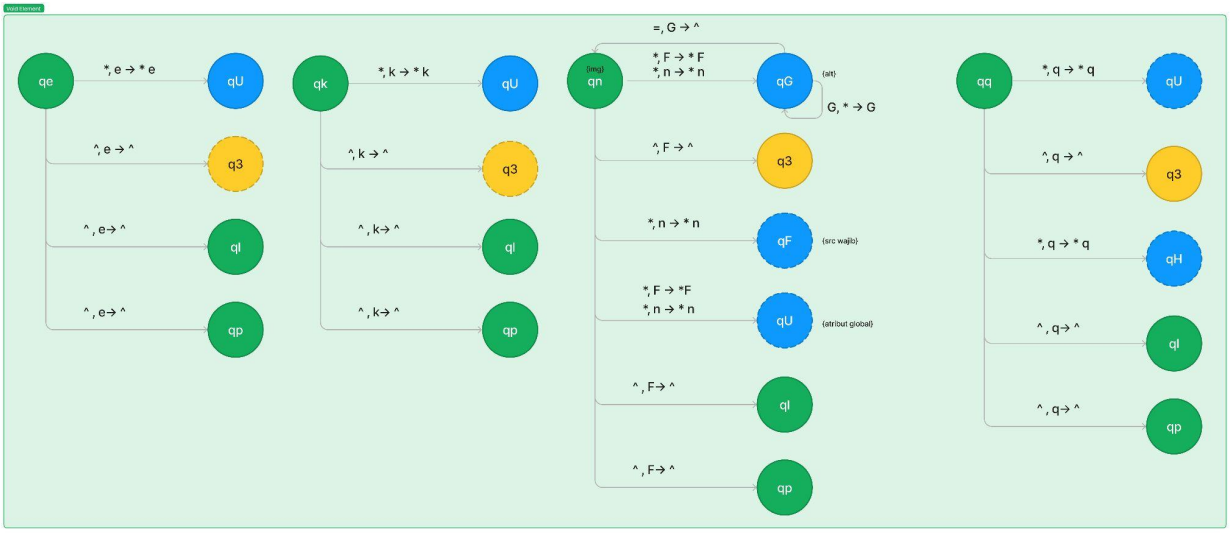


Gambar 2.5 Diagram Tag Formulir

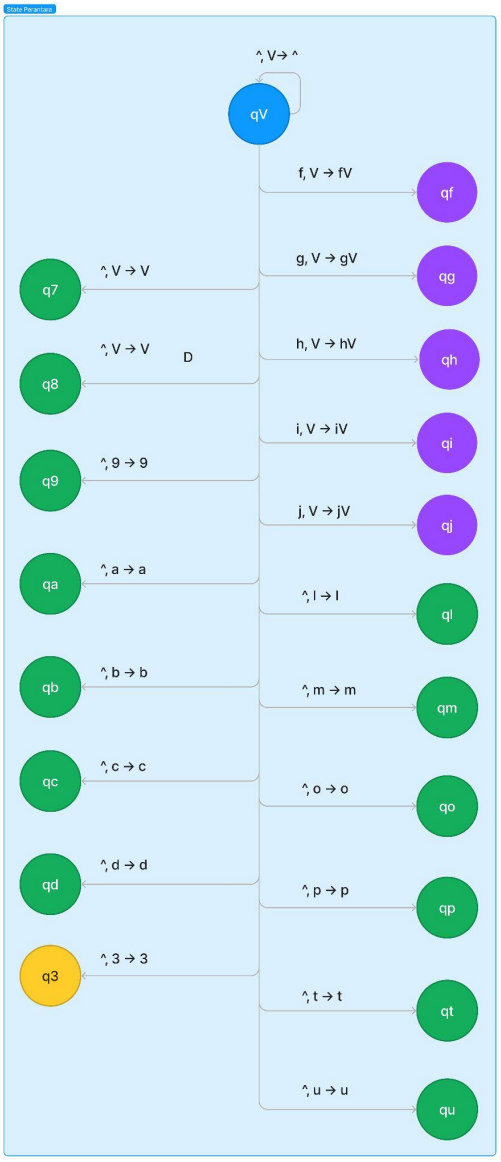


Gambar 2.6 Diagram Tag Tabel

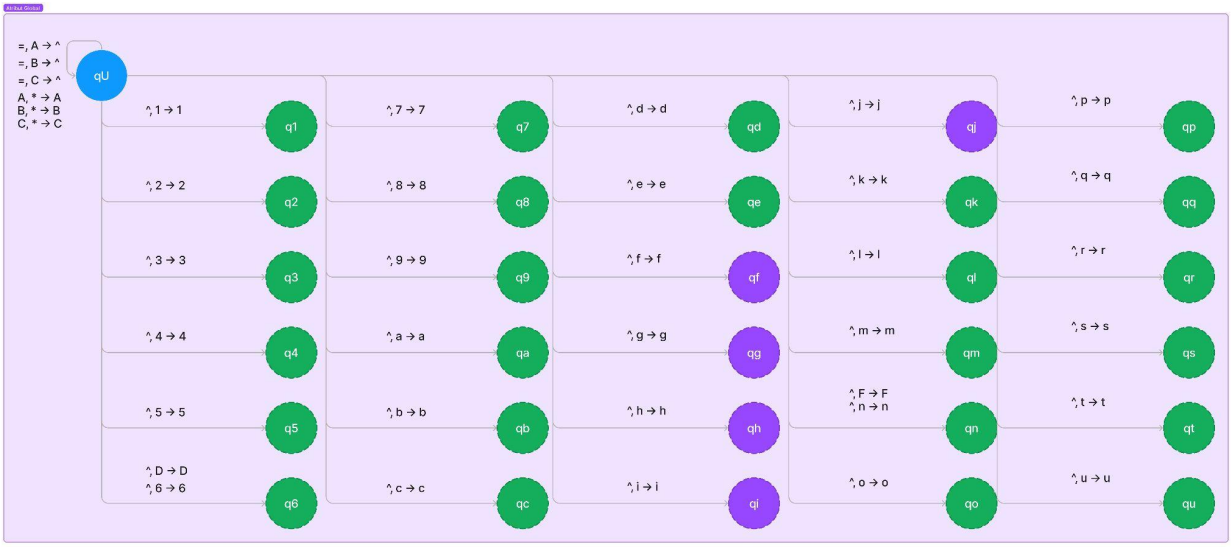
Gambar 2.7 Diagram Tag Div



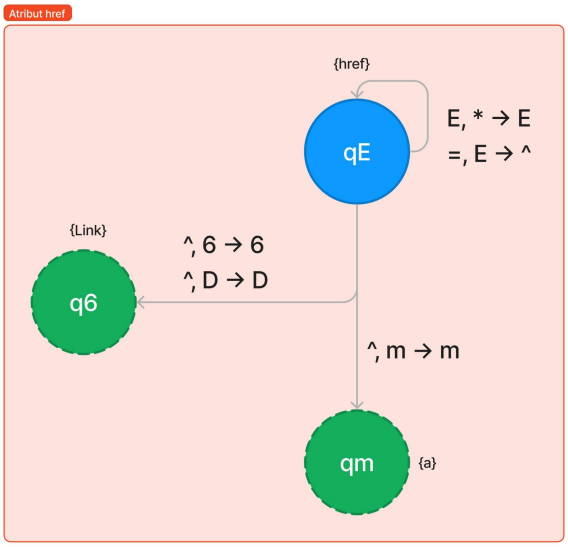
Gambar 2.8. Diagram Void Element



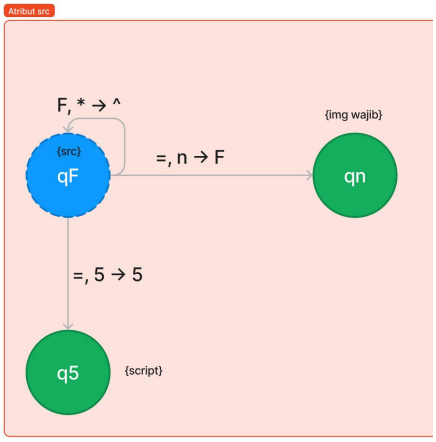
Gambar 2.9 Diagram State Perantara



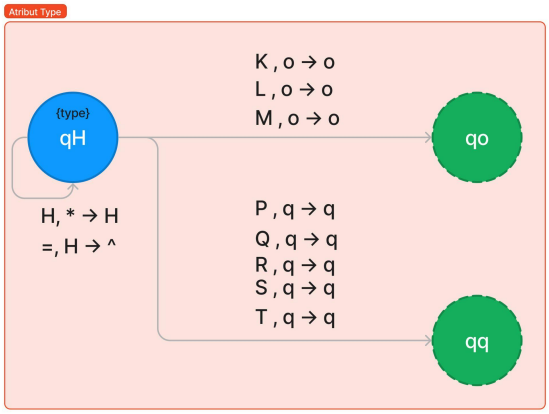
2.10 Diagram Atribut Global



2.11 Diagram Atribut Href



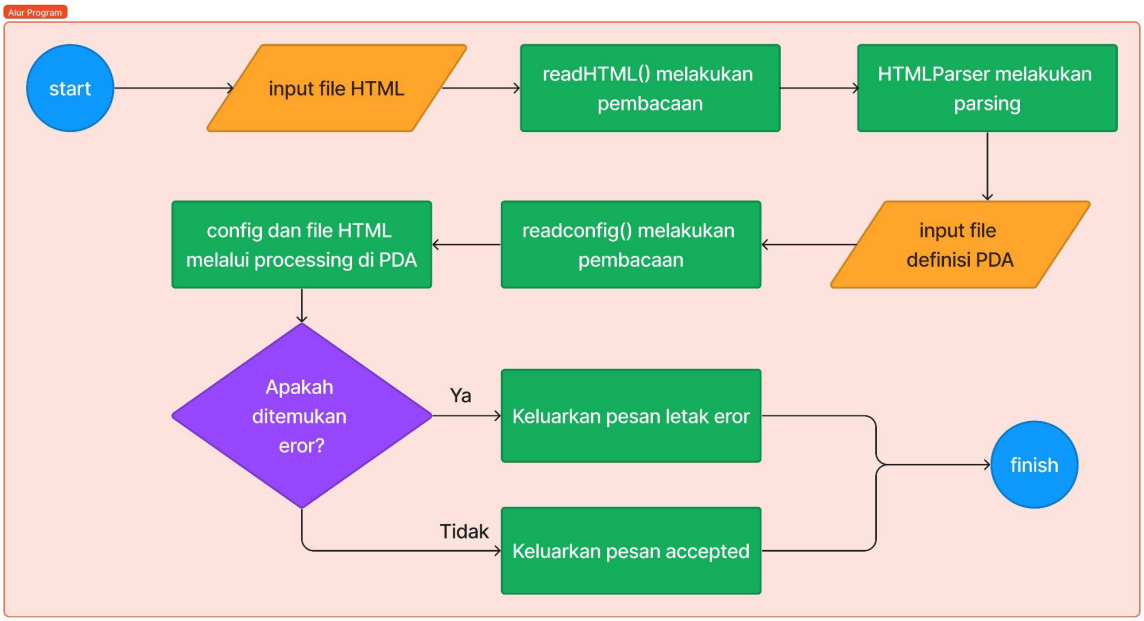
2.12 Diagram Atribut Src



2.13 Diagram Atribut Type

3 Implementasi dan Pengujian

3.1 Alur Program



Gambar 3.1 Alur Program

Pertama-tama program akan meminta user untuk menginput nama file config dan html yang ingin dicek. File html tersebut akan dibaca dan dilakukan parsing melalui readHTML dan htmlParser. Selanjutnya, file config berupa definisi state-state PDA yang sebelumnya diinput akan dibaca melalui readconfig. Kedua file ini akan diproses bersama-sama pada main dan melalui Push Down Automata. Terakhir, akan dikirimkan pesan error beserta letak kesalahannya apabila ditemukan error. Apabila tidak ditemukan error, maka dikirimkan pesan “accepted”.

3.2 Struktur Data Program

1. **Stack**

   Struktur data *Stack* digunakan untuk mengimplementasikan stack pada PDA. Stack dapat di-*pop* dan di-*push* sesuai dengan transition function pada config file.

2. **Dictionary**

   Struktur data *Dictionary* digunakan untuk melakukan tokenisasi tag, attribute, dan nilai attribute html. Misalnya, tag <html> di-*mapping* menjadi token ‘1’ untuk selanjutnya diproses oleh PDA.

   Selain itu, struktur data *Dictionary* juga dipakai untuk menyimpan *total states, input symbols, stack symbols, starting state, starting stack, accepting state,* dan *productions* pada config PDA.

3. **Class MyHTMLParser()**

MyHTMLParser() adalah class dari objek python yang dapat melakukan parsing terhadap input string file html. MyHTMLParser() diperoleh dari *library* html.parser. Class ini dapat meng-*handle* pemrosesan start tag, end tag, data, comment, dan lain-lain. MyHTMLParser() digunakan dalam program kami untuk melakukan tokenisasi dari file html menjadi string hasil tokenisasi.

3.3 Fungsi dan Prosedur

3.3.1 Fungsi

1. readHTML(path)

   Parameter:

   1. Path: path menuju file HTML yang akan dibaca.

   Return Value:

   1. String: string hasil tokenisasi HTML menjadi string yang dapat diinput ke PDA.

   Penjelasan:

   Fungsi readHTML() menerima 1 parameter yaitu *path* yang merupakan string path menuju file '.html' yang akan dibaca. readHTML() kemudian mengecek apakah file ada, dan jika ada akan melakukan 'read' pada file tersebut dan dimasukkan ke dalam variabel data. Data akan di-*pass* kepada method .feed() pada objek 'parser' dengan class MyHTMLParser. Hasil Tokenisasi oleh parser akan di return sebagai string.

2. readConfig(path)

   Parameter:

   1. Path: path menuju file config yang akan dibaca.

   Return Value:

   1. config: dictionary hasil pembacaan file config.

   Penjelasan:

   Fungsi readConfig() menerima satu parameter yaitu path menuju file config yang akan dibaca. Jika path valid, maka fungsi akan melakukan pembacaan menggunakan open() dan tiap line akan diproses menjadi config.

   config sendiri merupakan dictionary dengan 7 key-value pair. Key pertama adalah 'total_states' yaitu semua state dalam PDA. 'input_symbols' merupakan semua input yang mungkin pada PDA, stack_symbols adalah simbol-simbol dari stack. 'starting_state' adalah state awal ketika PDA dijalankan. 'starting_stack' adalah kondisi stack ketika PDA dipanggil. 'accepting _state' adalah himpunan semua final state/accepting state. Production adalah Transition function dari PDA yang tadi dibuat.

3. PDA(state, input, stack, config, error)

   Parameter:

   1. state: currentState dari PDA
   2. input: input yang harus dibaca PDA
   3. stack: currentStack dari PDA
   4. config: konfigurasi yang berisi (total states, input symbols, stack symbols, starting state, starting stack, accepting state, dan productions).

5. error: kondisi ketika terjadi fail/reject pada rekursi terdalam (length input, state, input, stack)

Return Value:

1. True/False
2. error

Penjelasan:

Fungsi PDA() menerima 5 parameter, yaitu state, input, stack, config, dan error. Error adalah tuple berisi 4 elemen yang merupakan panjang input, state, input, dan stack. Return value error digunakan untuk mengetahui letak kesalahan terakhir / terdalam ketika melakukan rekursi. Error akan di *update* saat rekursi jika panjang input < panjang input parameter error.

Fungsi PDA() mengembalikan boolean (True/False) serta variabel 'error'. Boolean (True/False) menandakan input diterima bila 'True', dan input ditolak bila 'False'.

3.3.2 Prosedur

Program tidak menggunakan prosedur.

3.4 Tokenisasi

Tokenisasi dilakukan menggunakan *library* **html.parser** dan **re** (regular expression) dengan *handle* kasus sebagai berikut.

3.4.1 Start Tag

Start Tag (<html>, <head>, <body>, dll) akan langsung ditokenisasi menjadi token dengan nilai tertentu sesuai tabel 3.1. Misalnya, tag "<html>" akan diubah menjadi token '1'. Jika nama tag tidak valid, misalnya "<htmq>", akan diubah menjadi *arbitrary token* '!'.

3.4.2 End Tag

End Tag (</html>, </head>, </body>, dll) akan langsung ditokenisasi menjadi token dengan nilai tertentu sesuai tabel 3.1. Misalnya, tag "</html>" akan diubah menjadi token '/1'. Jika nama tag tidak valid, misalnya "<htmq>", akan diubah menjadi *arbitrary token* '/!'.

3.4.3 Attribute

Attribute (id, style, class, type, src, dll) akan ditokenisasi menjadi token dengan nilai tertentu sesuai tabel 3.1. Misalnya, attribute "id" akan diubah menjadi token '*A'. Jika nama attribute tidak valid, misalnya "clas", akan diubah menjadi *arbitrary token* '*!'. Kemudian, dicek apakah attribute diikuti oleh ="(value)" atau ='(value)'. Jika benar diikuti, maka akan ditambahkan token '='.

Kasus khusus untuk attribute type dan method, nilai (value) juga akan ditambahkan menjadi token sesuai tabel 3.1 (nomor 41 - 50). Jika tidak sesuai, maka akan diubah menjadi arbitrary token '!'.

3.4.4 Content

Konten di dalam sebuah tag "*<tag>(konten)</tag>*" akan diubah menjadi token '+'.

3.4.5 Comment

Comment akan di-*handle* dengan cara tidak memasukkannya ke string hasil tokenisasi. Hal ini dilakukan karena comment dapat berada di mana saja asalkan di luar <> dan tidak memengaruhi PDA (state dan stack akan tetap jika bertemu *comment*).

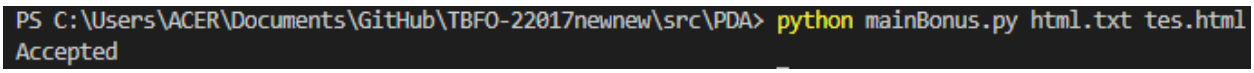
Tabel 3.1 Kamus Simbol Push Down Automata

No	Token	Makna	Keterangan
1	1	html	tag
2	2	head	
3	3	body	
4	4	title	
5	5	script	
6	6	link	
7	7	h1	
8	8	h2	
9	9	h3	
10	a	h4	
11	b	h5	
12	c	h6	
13	d	p	
14	e	br	
15	f	em	
16	g	b	
17	h	abbr	
18	i	strong	
19	j	small	
20	k	hr	
21	l	div	
22	m	a	

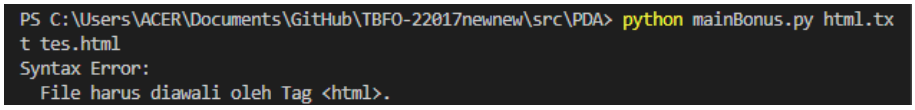
23	n	img	
24	o	button	
25	p	form	
26	q	input	
27	r	table	
28	s	row table	
29	t	tablehead	
30	u	kolom tabel	
31	A	id	atribut
32	B	class	
33	C	style	
34	D	rel	
35	E	href	
36	F	src	
37	G	alt	
38	H	type	
39	I	action	
40	J	method	
41	K	submit	nilai atribut
42	L	reset	
43	M	button	
44	N	GET	
45	O	POST	
46	P	text	
47	Q	password	
48	R	email	
49	S	number	
50	T	checkbox	
51	*	tanda sebelum atribut	lainnya
52	-	tanda sebelum formatting element	
53	=	tanda ketika atribut memiliki nilai	
54	/	tanda ketika tag akan ditutup	

3.5 Antarmuka

Program kami menggunakan antarmuka Command Line Interface seperti pada Gambar 3.2



Gambar 3.2 Tampilan Ketika File HTML Tidak Memiliki Eror



Gambar 3.3 Tampilan Ketika File HTML Memiliki Eror

3.6 Analisis Pengujian

Tabel 3.2 Test Case 1: Head muncul setelah body

```
<html>
  <body>
    <h1>Hello, World!</h1>
    <p>This is a simple webpage.</p>
  </body>
  <head>
    <title>Simple Webpage</title>
  </head>
</html>
```

Tabel 3.3 Test Case 2: Tag pertama bukan html

```
<hmif>
  <head>
    <title>Simple Webpage</title>
  </head>
  <body>
    <h1>Hello, World!</h1>
    <p>This is a simple webpage.</p>
  </body>
</hmif>
```

Tabel 3.4 Test Case 3: Tidak punya head

```
<html>
  <body>
    <h1>Hello, World!</h1>
    <p>This is a simple webpage.</p>
  </body>
</html>
```

Tabel 3.5 Test Case 4: Accepted

```
<html>
  <head>
    <title>Simple Webpage</title>
  </head>
  <body>
    <h1>Hello, World!</h1>
    <h2>Welcome to my page</h2>
    <img src=”./welcome.jpeg”>
    <p>This is a <em>simple</em> webpage.</p>

    <div id=”footer” class=”footer”> This is the end of the page </div>
  </body>
</html>
```

Tabel 3.6 Test Case 5: Accepted

```
<html>
  <head>
    <title>Simple Webpage</title>
  </head>
  <body>
    <!-- Bagian utama web -->
    <h1>Hello, World!</h1>
    <h2>Welcome to my page</h2>
    <hr>
    <img src="./welcome.jpeg" alt="Welcome Banner">
    <p>This is a <em>simple</em> webpage.</p>

    <!-- Custom element -->
    <div id=”footer” class=”footer”> This is the end of the page </div>
  </body>
</html>
```

Tabel 3.7 Test Case 6: Atribut wajib src pada img tidak ada

```
<html>
  <head>
    <title>Simple Webpage</title>
  </head>
  <body>
    <!-- Bagian utama web -->
    <h1>Hello, World!</h1>
    <h2>Welcome to my page</h2>
    <img alt="Welcome Banner">
    <p>This is a <em>simple</em> webpage.</p>

    <!-- Custom element -->
```

```
    <div id="footer" class="footer"> This is the end of the page </div>
  </body>
</html>
```

Tabel 3.8 Test Case 7: Accepted

```
<html>
<head>
  <title>Simple Webpage</title>

</head>
<body>

<h2>HTML Forms</h2>

<form action="/action_page.php" method="POST">
  <h5 class="label">First name:</h5><br>
  <input type="text" id="fname"><br>
  <h5 class="label">Last name:</h5><br>
  <input type="text" id="lname"><br><br>
  <button type="submit">Submit</button>
</form>

<p>If you click the "Submit" button, the form-data will be sent to a page
called "/action_page.php".</p>

</body>
</html>
```

Tabel 3.9 Test Case 8: Tidak termasuk value yang diperbolehkan (POST, GET)

```
<html>
<head>
  <title>Simple Webpage</title>

</head>
<body>

<h2>HTML Forms</h2>

<form action="/action_page.php" method="TEMBAK">
  <div id="label">First name:</div><br>
  <input type="text" id="fname"><br>
  <div id="label">Last name:</div><br>
  <input type="text" id="lname"><br><br>
  <button type="submit">Submit</button>
</form>

<p>If you click the "Submit" button, the form-data will be sent to a page
called "/action_page.php".</p>

</body>
</html>
```

Tabel 3.10 Test Case 9: Accepted

```
<html>
<head>
  <title>Simple Webpage</title>
  <script>
    document.getElementById("demo").innerHTML = "Hello JavaScript!";
  </script>
</head>
<body>

<h1>The script element</h1>

<p id="demo"></p>

</body>
</html>
```

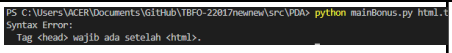
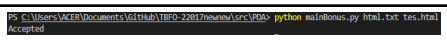
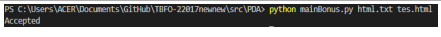
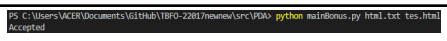
Tabel 3.11 Test Case 10: <p> bukan void element, wajib ditutup

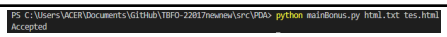
```
<html>
<head>
  <title>Simple Webpage</title>
  <script>
    document.getElementById("demo").innerHTML = "Hello JavaScript!";
  </script>
</head>
<body>

<h1>The script element</h1>

<p id="demo">

</body>
</html>
```

Tabel 3.12 Test Case 11: Accepted

```
<html>
    <head>
        <title>Simple Webpage</title>
        <script>
            document.getElementById("demo").innerHTML = "Hello JavaScript!";
        </script>
    </head>
    <body>
        <h1>The script element</h1>
        <a>Not going anywhere</a><br>
        <a href="https://www.google.co.id/">Might send you somewhere</a>
```

```
        <p id="demo"></p>
    </body>
</html>
```

Nomor Test Case	Hasil yang Keluar	Sesuai yang Diharapkan?	Analisis
1		Ya	Seharusnya tag head muncul setelah tag html tetapi malah muncul setelah body
2		Ya	Seharusnya file diawali <html> tetapi malah diawali <hmif>
3		Ya	Seharusnya terdapat tag head setelah tag html
4		Ya	Program berhasil menghandle: 1. Nesting head dan body di dalam tag html 2. Nesting h1, h2, img, p, dan div di dalam tag body 3. Nesting text di tag h1, h2, p, em, dan div 4. Atribut src pada tag img 5. Nesting formatting element di dalam tag p 6. Atribut id pada div .
5		Ya	Program berhasil menghandle: 1. Nesting head dan body di dalam tag html 2. Komentar 3. Nesting h1, h2, h3, img, p, dan div di dalam body 4. Nesting formatting elemen di dalam p 5. Atribut src dan id
6		Ya	Seharusnya terdapat atribut src pada tag img
7		Ya	Program berhasil menghandle: 1. Nesting head dan body di dalam tag html 2. Komentar 3. Nesting h2, form, dan p di dalam body 4. Nesting h5, input, dan button dalam form 5. Atribut type untuk button
8		Ya	Nilai atribut method harusnya terbatas antara POST atau GET saja
9		Ya	Program berhasil menghandle: 1. Nesting head dan body di dalam html

Nomor Test Case	Hasil yang Keluar	Sesuai yang Diharapkan?	Analisis
			2. Nesting title dan script di dalam head 3. Nesting h1 dan p di dalam body 4. Atribut id 5. Nesting text di dalam title dan h1
10		Ya	Seharusnya tag p ditutup dengan p bukan dengan body
11		Ya	Program berhasil menghandle: 1. Nesting head dan body di dalam html 2. Nesting title dan script dalam head 3. Nesting h1, a, dan p di dalam body 4. Nesting text di dalam title, script, h1, dan a 5. Atribut href pada a 6. Atribut id pada p

4 Daftar Pustaka

- https://docs.python.org/3/library/html.parser.html
- https://docs.python.org/3/library/re.html

5 Lampiran

Link Diagram:
https://www.figma.com/file/Uq6mBpGz1JMFgCCofURLtr/PDA-Tubes-TBFO-Ngangonga ngong?type=whiteboard&node-id=0%3A1&t=SIaTMvngNlsTY7Aw-1

Link Repository Github:
https://github.com/dhiabziz/TBFO-22017.git

Pembagian Tugas:

NIM	Nama	Tugas
13522020	Aurelius Justin Philo Fanjaya	1. Membuat tokenisasi 2. Membuat program PDA 3. Membuat pembaca config 4. Membuat pembaca HTML 5. Membuat bonus

		6. Membuat laporan
13522040	Dhidit Abdi Aziz	1. Membuat diagram state-state PDA 2. Membuat file external definisi PDA 3. Membuat laporan
13522017	Kresna Adi Prayogo	-