BAB III

METODE DAN PERANCANGAN APLIKASI

3.1 Metode Penelitian

Penelitian ini akan dilakukan dengan metode studi pustaka dan pada akhir penyelesaian program, maka akan melakukan metode percobaan untuk menguji tingkat akurasi dari program. Mempelajari tentang Kimia Organik, jenis-jenis reaksi untuk menentukan hasil reaksi, dan bagaimana proses suatu reaksi terjadi untuk nantinya diproyeksikan ke dalam gambar tiga dimensi dengan membaca buku-buku, artikel, *browsing* internet dan berbagai sumber lainnya.

Pengembangan sistem ini adalah berbasis *website* yang diharapkan lebih berpotensi untuk dipakai banyak orang karena jika seseorang mendapat akses internet, orang tersebut dapat mengakses *website* ini tanpa harus melakukan *downloading* dan sebagainya. Pengembangan sistem ini memakai metode spiral, yaitu metode perbaikan dari model *waterfall* dan *prototype*.

1. Studi Pustaka

Kumpulan data yang nantinya akan dipakai, diambil dari buku, *paper*, jurnal, makalah dan media penelitian lain yang telah dilakukan sebelumnya yang berkaitan dengan ikatan kimia, senyawa kimia, kimia organik, tabel periodik dan algoritma prim.

2. Perancangan dan implementasi program

Pada tahap perancangan dilakukan analisis terhadap prosedur atau langkah-langkah yang harus dilaksanakan agar program ini bisa terimplementasi atau dapat dijalankan. Selain menganalisa prosedur, dilakukan juga perancangan untuk

tampilan dari *website*, apa saja halaman yang diperlukan dari *website* dan bagaimana cara kerja dari *website* ini. Rancangan tersebut kemudian didokumentasikan ke dalam diagram alir.

3. Uji coba atau *testing* dan *debugging*

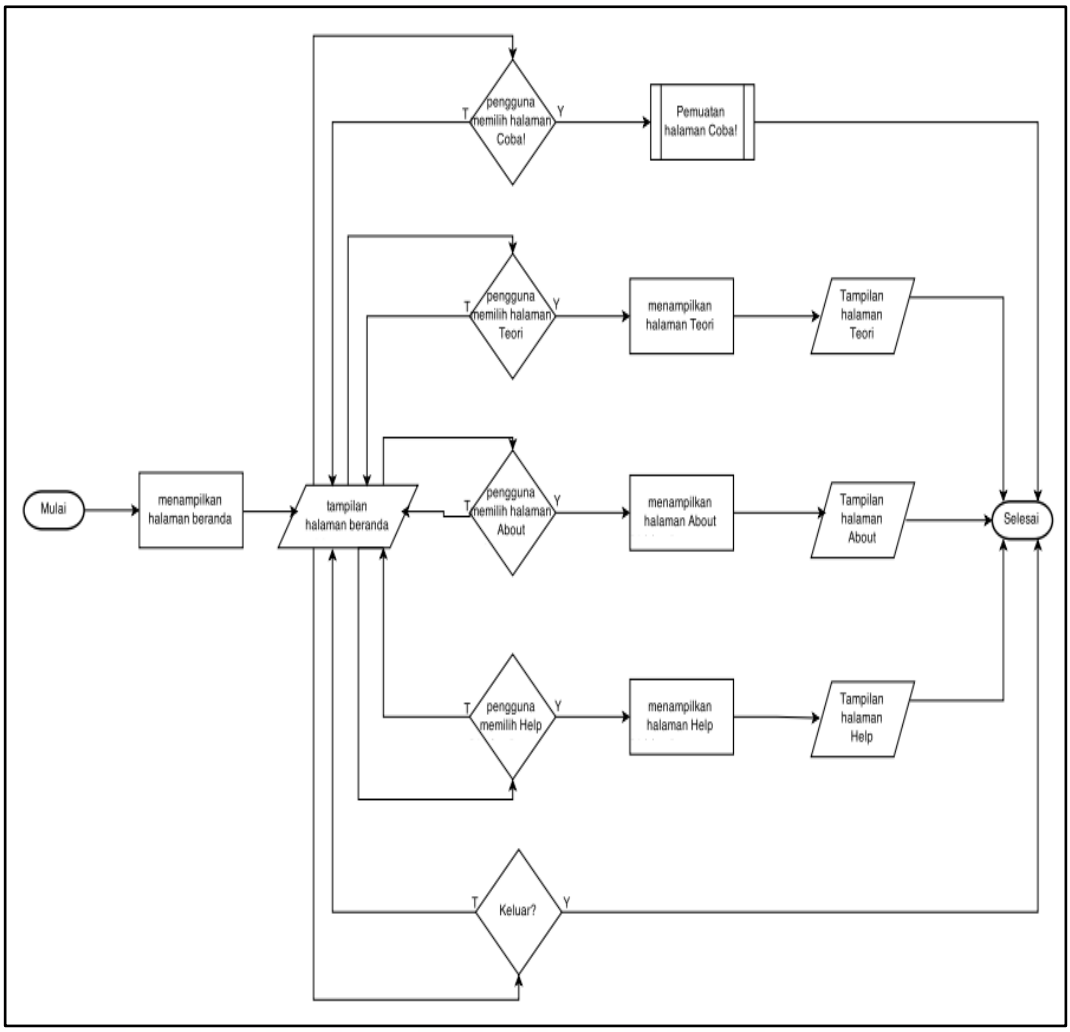
Selama proses pengembangan *website* dilakukan serangkaian uji coba sistem apakah sistem berjalan sesuai dengan rancangan atau tidak. Jika ditemui bug atau kesalahan dari sistem maka langsung dilakukan perbaikan. Setelah *website* telah berhasil dikembangkan, dilakukan publikasi. Dalam publikasi *website* disediakan pilihan untuk pengguna dapat memberi *feedback* dari *website*. Dengan demikian dari *feedback* yang didapatkan, proses *testing* program juga bisa berjalan.

3.2 Perancangan

Rancangan sistem dituangkan ke dalam diagram alir atau *flowchart*. Antarmuka *website* ini secara garis besar adalah website yang dapat menampilkan halaman awal atau beranda yang berisi pilihan-pilihan yang akan mengarahkan pengguna ke halaman yang diinginkan sesuai dengan pilihannya. Untuk pengimplementasian algoritma prim atau dengan kata lain inti dari *website* ini adalah ada di halaman Coba. Di dalam halaman ini, pengguna akan memasukkan *input*, kemudian masukan tersebut diproses dan kemudian program mampu memberikan *output* atau keluaran sesuai dengan masukan dari pengguna tersebut.

3.2.1 Perancangan Diagram Alir

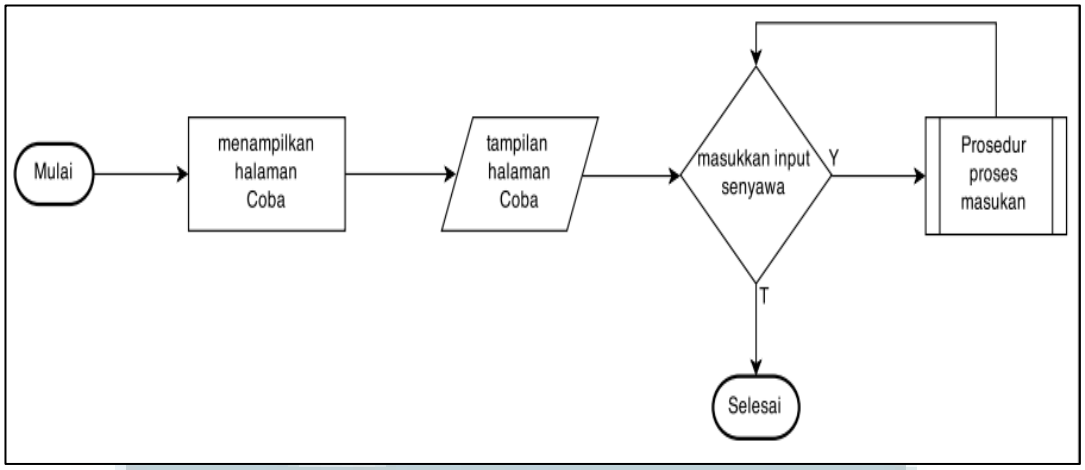
A. Website Secara Keseluruhan



Gambar 3. 1 Diagram Alir *Website* Secara Keseluruhan

Gambar 3.1 menggambarkan diagram alir untuk *website* secara keseluruhan dari mulai buka *website* dengan menampilkan halaman beranda sesuai dengan tampilan yang telah dirancang sebelumnya. Selanjutnya akan ada pemilihan jika pengguna memilih halaman Coba, maka sistem akan melanjutkan ke prosedur memuat halaman tersebut, jika memilih halaman Teori atau halaman About atau halaman Help maka akan melanjutkan dengan menampilkan halaman Teori atau halaman Tentang atau halaman Help.

B. Pemuatan Halaman Coba

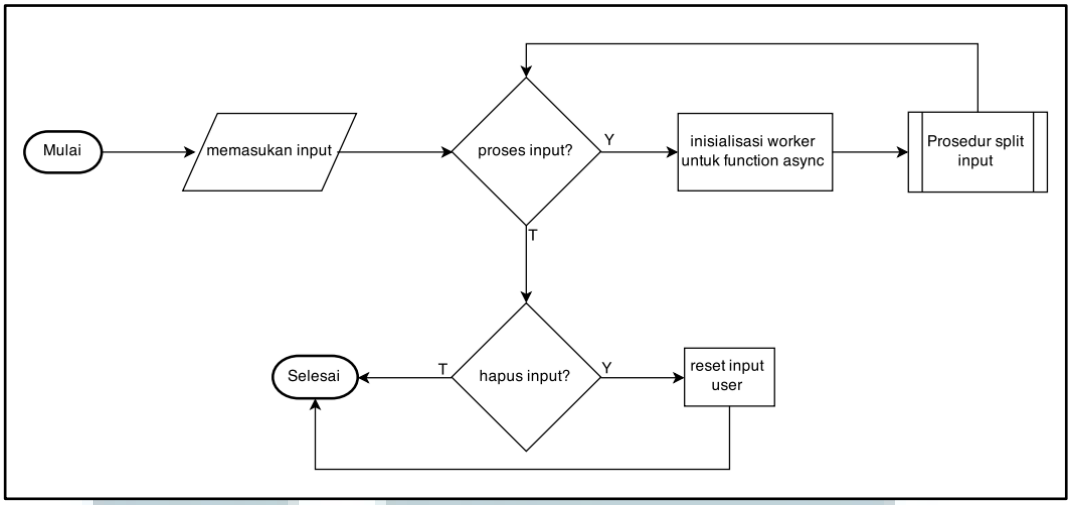


Gambar 3. 2 Diagram Alir Pemuatan Halaman Coba

Di halaman Coba sistem akan menampilkan halaman Coba seperti rancangan antar muka yang telah dibuat untuk halaman ini. Jika pengguna memasukkan *input*-an senyawa maka, senyawa tersebut akan diproses seperti pada Gambar 3.3 dibawah ini.

C. Prosedur Proses Masukan

Di dalam rancangan antar muka untuk halaman Coba, ada dua tombol yang bisa dipilih untuk memproses masukan. Jika pengguna menekan tombol Coba, maka masukan atau *input* dari pengguna akan dilanjutkan ke prosedur atau langkah selanjutnya, yaitu proses *split input*. Jika pengguna menekan tombol Hapus, maka akan dilanjutkan dengan proses *reset* atau atur ulang masukan ke posisi awal, yaitu kosong. Prosedur untuk memproses masukan akan digambarkan ke dalam diagram alir sebagai berikut.



Gambar 3. 3 Diagram Alir Prosedur Proses Masukan

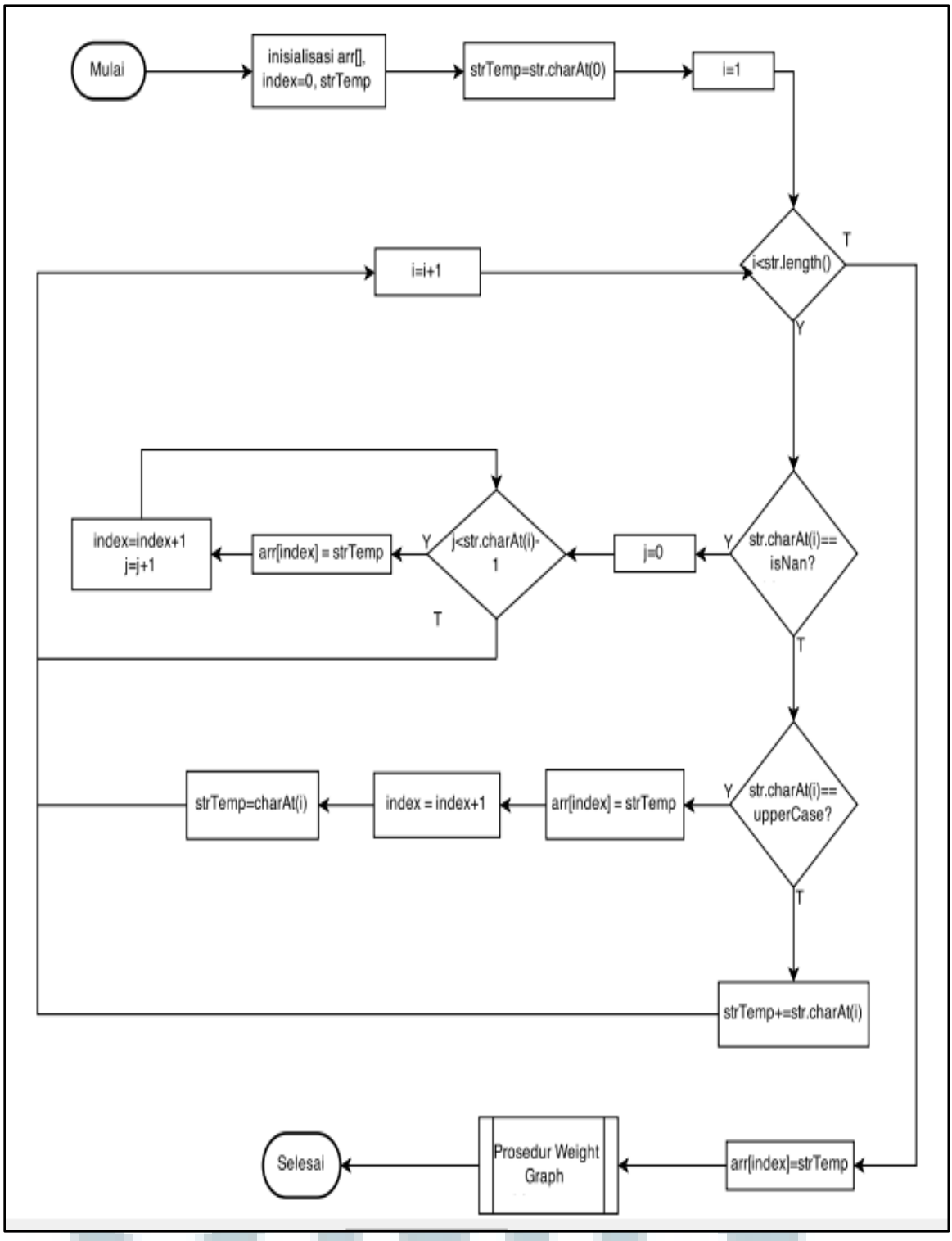
Perancangan untuk proses-proses selanjutnya banyak menggunakan metode rekursif atau perulangan, maka aplikasi ini membuat sebuah penampung *worker*. Penampung ini membuat *function* bisa berjalan secara asynchronous. Dengan kata lain, walaupun nanti proses rekursif berjalan lama, proses utamanya tetap bisa jalan. Jika proses tidak dibuat *asynchronous*, ada kemungkinan terjadi *page unresponsive*.

Proses validasi input akan dilakukan selama berjalannya proses sistem ini. Hal ini disebabkan oleh untuk mengetahui suatu *input* atau masukan dari *user* atau pengguna tidak bisa langsung diketahui secara langsung pada saat *user* memasukkan *input*, tapi untuk mengetahui masukan atau *input* dari *user* itu salah adalah saat masukan di-*split* baru bisa tahu apakah masukan tersebut merupakan suatu senyawa atau tidak. Misalnya, masukan atau input *user* adalah ‘sjs2gh’, maka dalam proses *split input* akan diketahui jika masukan adalah salah.

Kasus lainnya adalah saat mencari *weight* dari unsur, setelah *split input* dilakukan maka akan dilakukan pencarian *weight*, jadi jika unsur yang telah merupakan hasil dari *split* tidak ditemukan, maka data yang dimasukkan tidak *valid*. Misalnya jika masukan dari pengguna menyerupai senyawa tapi sebenarnya bukan senyawa, yaitu 'Ma2Pa'. Ma bukan merupakan unsur begitu pula dengan Pa.

D. Prosedur Split Input

Pada proses ini program akan melakukan pembacaan pada masukan atau *input* dari *user*. Masukan dibaca berdasarkan huruf. Proses pembacaan perhuruf dilakukan karena asumsi masukan atau *input* dari pengguna adalah senyawa. Senyawa berarti terdiri dari beberapa unsur kimia. Senyawa yang dimasukan harus benar sesuai dengan penulisan senyawa kimia yang benar dengan memperhatikan *uppercase* dan *lowercase* dari unsur. Setiap unsur dibedakan dengan huruf besar, misalnya, H, O, Cl, C, Rb, Na, Al, S, dan seterusnya. Dan penulisan angka untuk senyawa yang memiliki lebih dari satu unsur yang sama, misalnya, C2H6, senyawa memiliki dua unsur C dan enam unsur H.

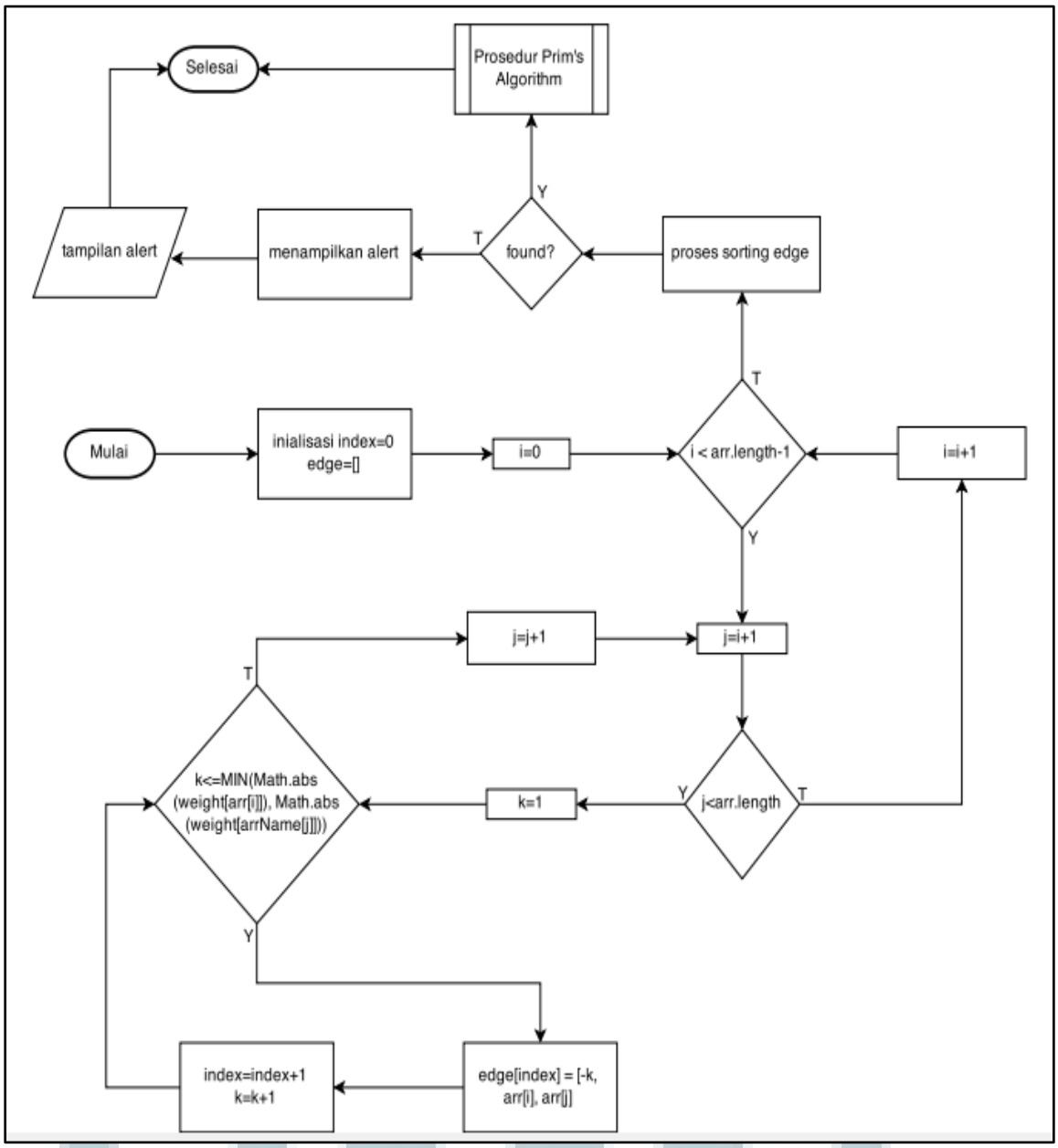


Gambar 3. 4 Diagram Alir Prosedur *Split Input*

Gambar 3.4 menggambarkan diagram alir dari proses untuk memisahkan input atau masukan dari pengguna menjadi perhuruf. Senyawa yang dimasukkan oleh pengguna yang sudah divalidasi atau sudah benar akan diubah menjadi bentuk unsur-unsur yang membentuk senyawa tersebut. Program membentuk *array* yang akan menampung unsur-unsur dari senyawa masukan pengguna. Misalnya, senyawa masukan atau *input* dari pengguna adalah C_2H_6. Senyawa C2H6 dibentuk dari dua unsur C (karbon) dan enam unsur H (hidrogen). Dengan demikian, program akan menyimpan karakter masukan yang ke-0 (str.charAt(0)) dan memeriksa karakter berikutnya. Jika karakter berikutnya adalah angka (isNaN), maka program akan menampung unsur tersebut ke dalam *array* (arr[]) sebanyak angka tersebut (j < str.charAt(i)). Jadi, dengan contoh di atas, C2H6, di dalam *array*, arr[], akan tersimpan dua unsur C. Jika karakter berikutnya, adalah huruf besar (*uppercase*), maka strTemp (penampung string ke-0) akan dimasukkan ke dalam *array split input* yang ke-*index*, yaitu ke-0. Setelah itu, *index* ditambahkan dan penampung berubah menjadi str.charAt(i) dan proses berulang ke pemeriksaan huruf selanjutnya.

Jika karakter berukutnya adalah huruf kecil (*lowercase*), maka strTemp akan menambahkan penampung tersebut dengan str.charAt(i) dan dilanjutkan dengan pembacaan hal ini dilakukan karena pada tabel periodik terdapat unsur yang memiliki elemen huruf besar dan huruf kecil, masing-masing satu, misalnya Ca, Rb, Al, dan lain sebagainya. Di dalam algoritma Prim unsur-unsur yang dibentuk ini adalah *vertex* atau *node* dari graf G.

E. Prosedur Weight Graph



Gambar 3. 5 Diagram Alir Prosedur *Weight Graph*

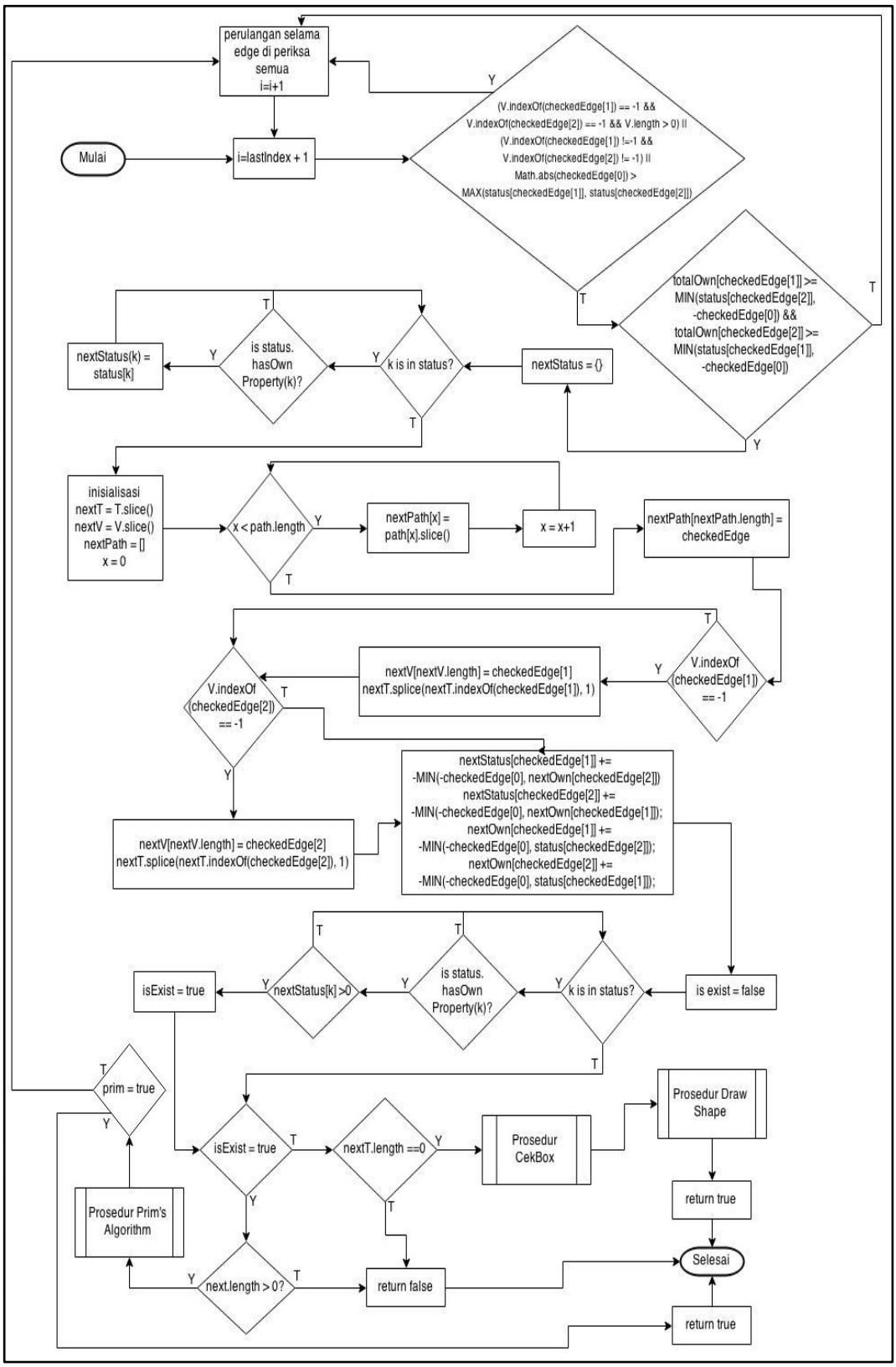
Gambar 3.5 adalah lanjutan dari proses *split input*. *Array* yang dihasilkan oleh prosedur *split input* akan dibandingkan antara arr[i] dan arr[i+1] atau dengan kata lain, dilakukan pemeriksaan antara *array* indeks yang ke-1 dan *array* indeks yang ke- 1+1, yaitu yang ke-2, dan seterusnya. Proses perbandingan antara kedua isi *array* ini dilakukan karena seperti yang dipaparkan sebelumnya, bahwa nilai simpul antara dua *node* atau *vertex* yang bersisian akan dipilih *weight* yang terkecil dari kedua unsur. Disinilah proses tersebut dijalankan. Untuk mencari *weight* dari simpul.

Proses :

(k < = MIN(Math.abs(weight[arrName[i]]), Math.abs(weight[arrName[j]])),

dimaksudkan untuk proses perulangan pengisian *weight* bagi dua *vertex* yang bersisian yang memiliki *weight* lebih dari satu. Misalnya, untuk senyawa C2H4, untuk unsur C bersisian atau berikatan dengan unsur C, bisa saling berikatan 4 (empat), jadi dilakukan perulangan sebanyak 4 kali, dan nantinya akan dilakukan pemilihan pada saat prosedur Prim, untuk membuat senyawa tetap seimbang. Saat proses pemeriksaan sampai pada indeks arr.length dan arr.length-1, misalnya pada *array* yang dibentuk H2O, adalah H0, H1, O0 dilakukan pemeriksaan terakhir pada H1 dan O0, maka akan dilakukan proses *sorting* untuk membuat *weight* terurut untuk nantinya masuk ke dalam proses Prim *Algorithm*. Jika tidak ditemukan, maka akan menampilkan *output* “Periksa Masukan Anda”.

F. Prosedur *Prim's Algorithm*



Gambar 3. 6 Diagram Alir Prosedur *Prim's Algorithm*

Gambar 3.6 menggambarkan diagram alir dari prosedur algoritma Prim. *Function* prim menerima lastIndex yang diberi oleh *function* sebelumnya saat memanggil *function* ini, telah dilakukan proses rekursif sebelumnya sudah ada pengecekan untuk lastIndex. Jadi, misalnya, nilai lastIndex adalah 5 (lima) berarti *array* yang ke-0 sampai ke-5 tidak perlu dilakukan pemeriksaan lagi. Saat mengirim nilai lastIndex pada saat pemanggilan *function*, dikirim nilai -1 yang artinya prim harus memeriksa *array* yang ke-0 sampai ke-n.

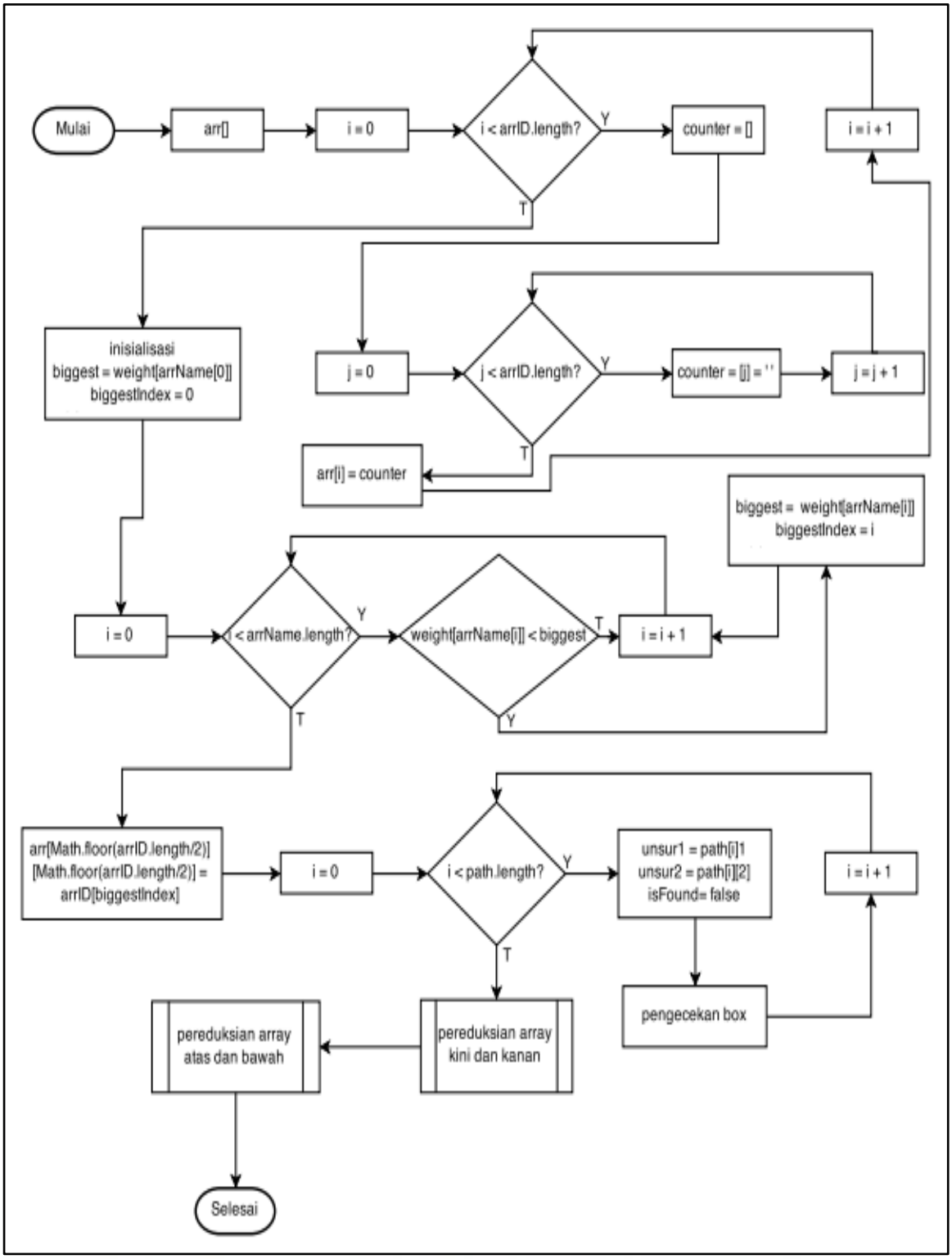
Masuk ke dalam proses pemilihan if. Ada tiga syarat yang harus terpenuhi oleh proses pemilihan tersebut, yaitu pertama jika *node* di jalur ke-*edge* yang diperiksa tidak ada didalam V (daftar dari *node* yang sudah terhubung), maka tidak bisa dihubungkan lagi dengan *node* lain, jadi sistem *continue*, atau yang kedua, jika kedua *node* di *edge* sudah ada di V berarti tidak perlu dihubungkan lagi, jadi sistem *continue*, atau yang ketiga, pengecekan *edge* yang dipakai melewati batas *weight* atau tidak, kalau melewati batas, maka sistem *continue*.

Setelah itu melakukan perulangan *foreach* untuk meng-*copy* status ke nextStatus. status dikirim dari *function* sebelumnya yang memanggil *function* ini yang berisi pasangan *object* dari nama unsur dan *weight* dari unsur tersebut. Variabel-variabel yang mengandung next, yaitu nextStatus, nextT (menampung node yang belum dikunjungi), nextV (menampung node yang sudah visited atau sudah dikunjungi), nextPath (jalur yang sudah diperiksa) adalah variabel-variabel penampung untuk *node* selanjutnya. Data next ditampung karena proses ini adalah proses rekursif, jadi data *node* yang asli tetap disimpan. Proses nextPath[nextPath.length] = checkedEgde adalah proses menampung jalur berikutnya, dan menganggap jalur (checkedEdge) tersebut benar. Misalnya,

nextPath adalah, X0-Y0-Y1, dan checkEdge adalah Y1-X2 dengan proses ini menampung jalur (nextPath) X0-Y0-Y1-X2. Kemudian dilakukanlah dua proses pemeriksaan, yaitu proses memeriksa *path* atau jalur yang dianggap benar tadi apakah berada di dalam unsur yang telah di-*visited* (V) atau belum *visited* (T). Proses selanjutnya adalah proses untuk mengurangi nilai *weight* (berapa yang dibutuhkan) dan nilai *own* (berapa yang dimiliki). Jadi, karena nextStatus dan nextOwn telah ditambahkan dengan *weight* (inisialisasi *weight* adalah – (minus)) dari checkEdge, maka didapatilah nextStatus dan nextOwn bisa berkurang. Selanjutnya dilakukan proses pemeriksaan pada checkedEdge untuk memastikan yang dibutuhkan oleh masing-masing unsur tidak *minus*.

Kemudian proses dilanjutkan dengan memeriksa apakah ada unsur yang masih membutuhkan atom atau tidak. Jika masih, maka bisa prosedur prim dilakukan kembali. Jika semua unsur sudah terpenuhi, maka proses berlanjut pada Prosedur Cek Box.

G. Prosedur Cek Box



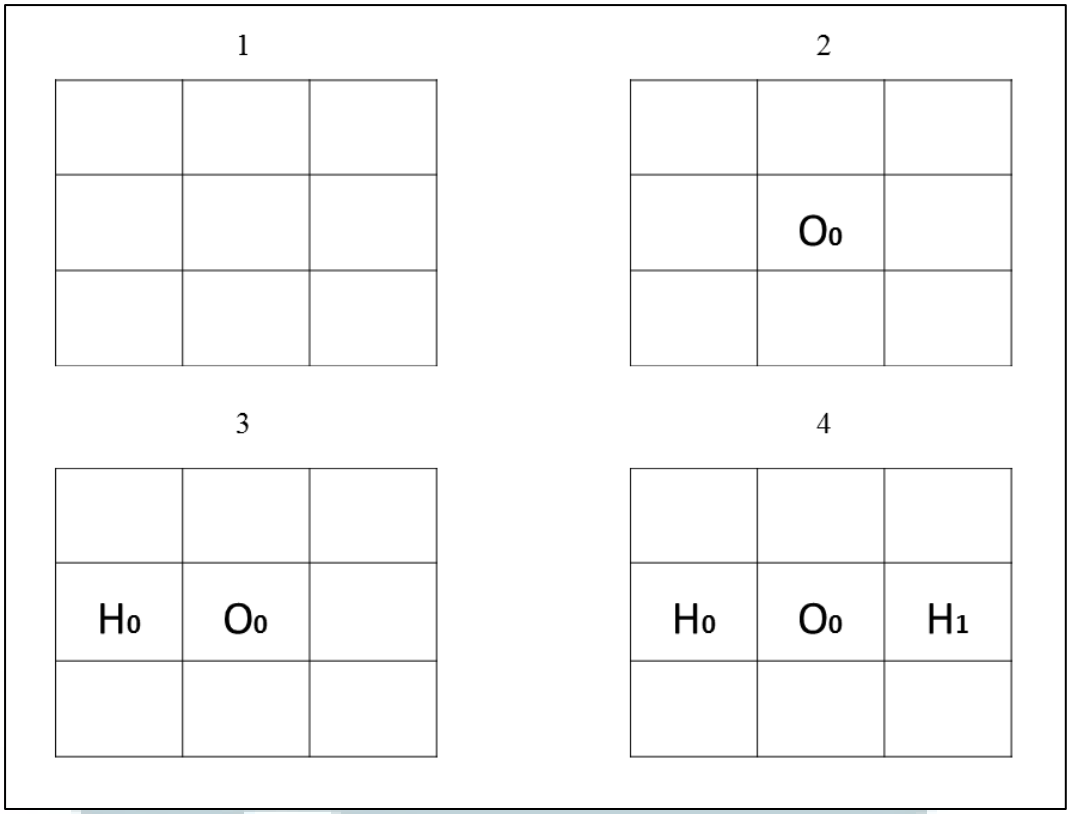
Gambar 3. 7 Diagram Alir Prosedur Cek *Box*

Gambar 3. 7 memberikan penjelasan mengenai diagram alir dari proses cek *box*. Penggambaran visualisasi dari ikatan senyawa kimia menggunakan *canvas* 2D, jadi disini dibuatlah sebuah penampung *array*, arr[], untuk nanti membuat matriks 2D untuk pengisian unsur ke dalam *box*. Ukuran dari matriks 2D itu adalah ukuran panjang dari senyawa.

Misalnya, suatu senyawa X2Y4, yaitu memiliki 6 (enam) unsur, maka panjang matriks adalah 6x6. Hal ini disebabkan karena kemungkinan unsur tersebut membentuk ikatan berbentuk linier sepanjang banyaknya unsur. Dalam contoh kasus ini, maka kemungkinan maksimal pengisian matriks adalah Y-X-Y-X-Y-Y. Jadi, matriks dibuat dalam ukuran arrID.length x arrID.length.

Selanjutnya akan dilakukan pencarian unsur apa yang merupakan prioritas. Maksud dari prioritas adalah unsur yang memiliki banyak koneksi dengan unsur lainnya atau dengan kata lain, manakah *node* yang memiliki simpul terbanyak dengan *node* yang lain. Oleh karena itu di inisialisasi prioritas adalah yang paling *array* yang pertama lalu dilakukan perulangan untuk membandingkan dengan *node* yang lain. Dengan demikian, pada akhir perulangan akan ditemukan ‘biggest’, yaitu *weight* dari *node* yang menjadi prioritas dan pada ‘arr’ indeks ke berapa *node* tersebut berada.

Berikut adalah gambaran dan penjelasan dari proses peletakkan *node* ke dalam matriks. Contoh senyawa yang dipakai adalah senyawa air, H2O.



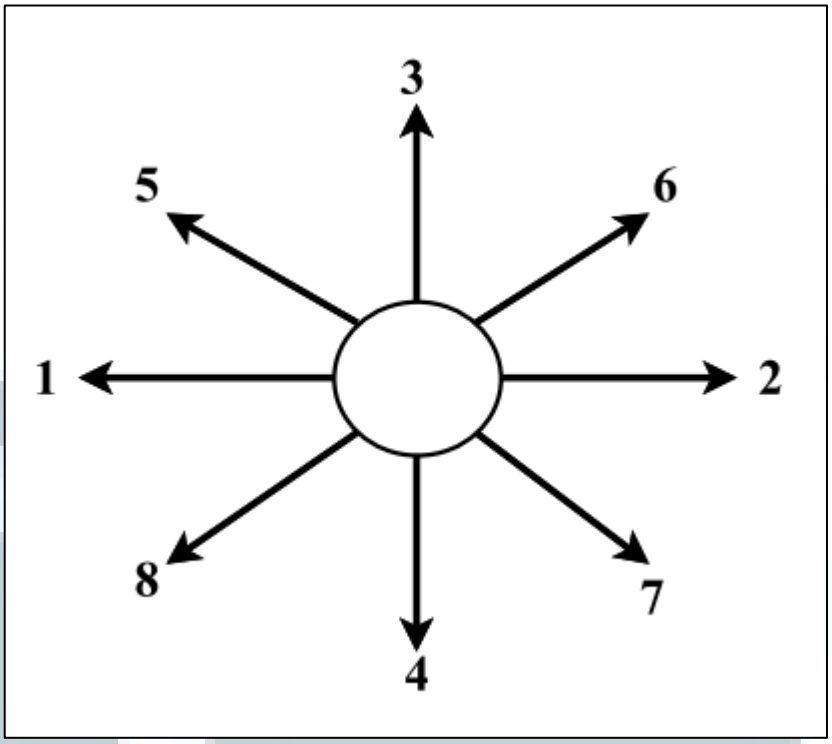
Gambar 3. 8 Gambaran peletakan node pada matriks

1. Pembentukan matriks[arrID.length][arrID.length]. Senyawa H2O, memiliki unsur sebanyak 3 (tiga), jadi matriks yang terbentuk adalah matriks 3x3.
2. arr[Math.floor(arrID.length/2)][Math.floor(arrID.length/2)] = arrID[biggestIndex].

   Proses ini adalah proses meletakkan *node* prioritas ke tengah matriks yang telah dibentuk sebelumnya. Setelah itu dilakukan perulangan untuk mencari tempat didalam matriks. Unsur O (oksigen) adalah prioritas, karena memiliki 2 (dua) simpul.

   arr[Math.floor(3/2)][Math.floor(3/2)] → arr[Math.floor(1.5)][Math.floor(1.5)] → arr[1][1] → posisi tengah dari matriks.

3. Prioritas dari peletakan *node*.



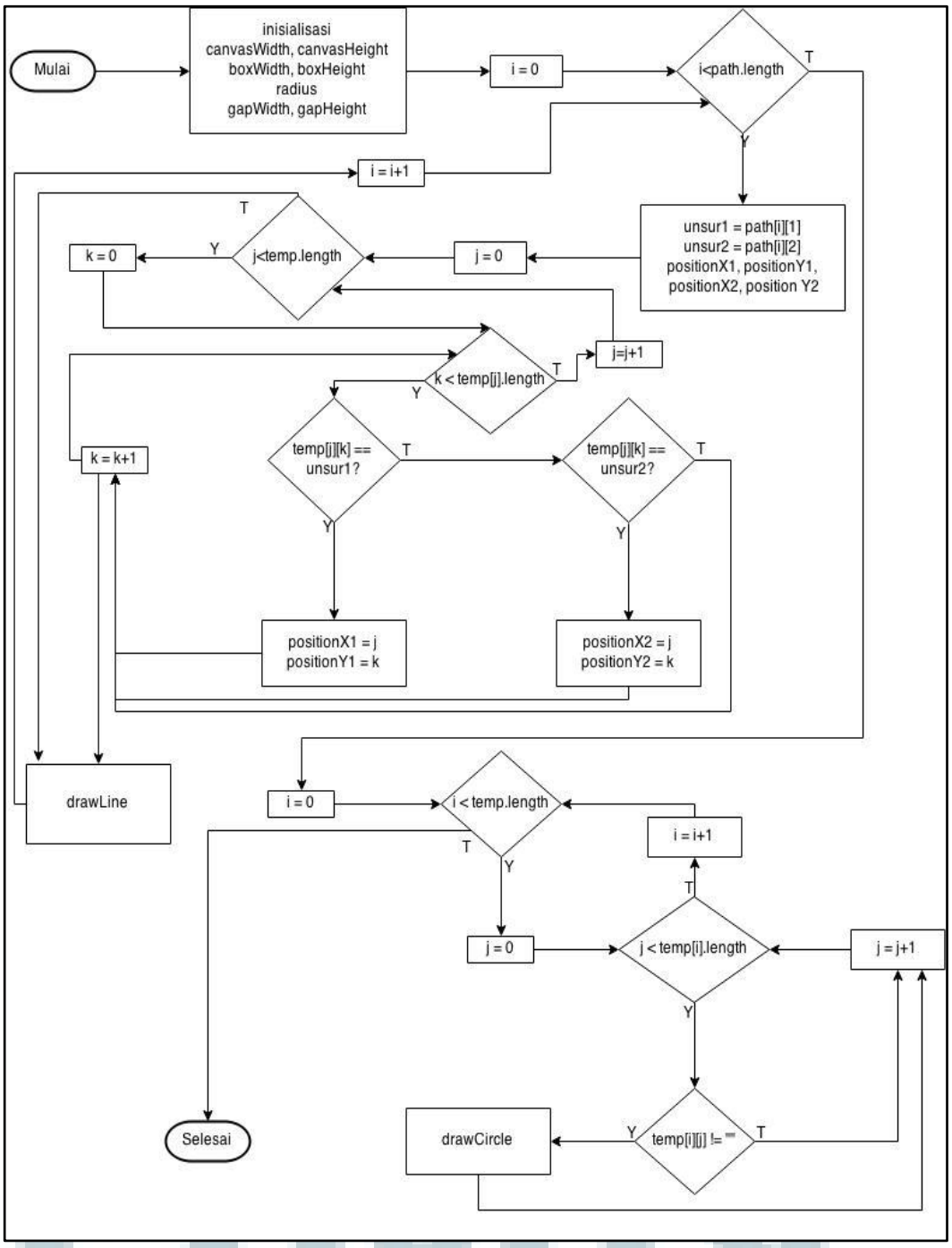
Gambar 3. 9 Prioritas peletakan *node*

Lingkaran adalah *node* yang isFound (sudah terisi). Untuk meletakkan *node* lain yang bersisian dengan *node* tersebut dilakukan pengecekan di seluruh arah yang mungkin. Prioritas peletakan *node* bersisian adalah seperti yang telah digambarkan pada Gambar 3. 9.

4. Lanjutan dari proses nomor 3 (tiga).

Setelah selesai ditemukan, dilakukanlah proses reduksi matriks dari kiri dan kanan serta atas dan bawah sehingga ukuran penggambaran di *canvas* bisa maksimal.

H. Prosedur Draw Shape



Gambar 3. 1 Diagram Alir Prosedur *Draw Shape*

Saat *function* akan dimulai, maka akan dilakukan inisialisasi variabel yang dibutuhkan untuk menggambar lingkaran. Variabel-variabel tersebut adalah canvasWidth, canvasHeight, boxWidth, boxHeight, radius (jari-jari lingkaran), gapWidth (jarak antar lingkaran merentang ke kiri atau kanan), dan gapHeight (jarak antar lingkaran merentang ke atas atau bawah).

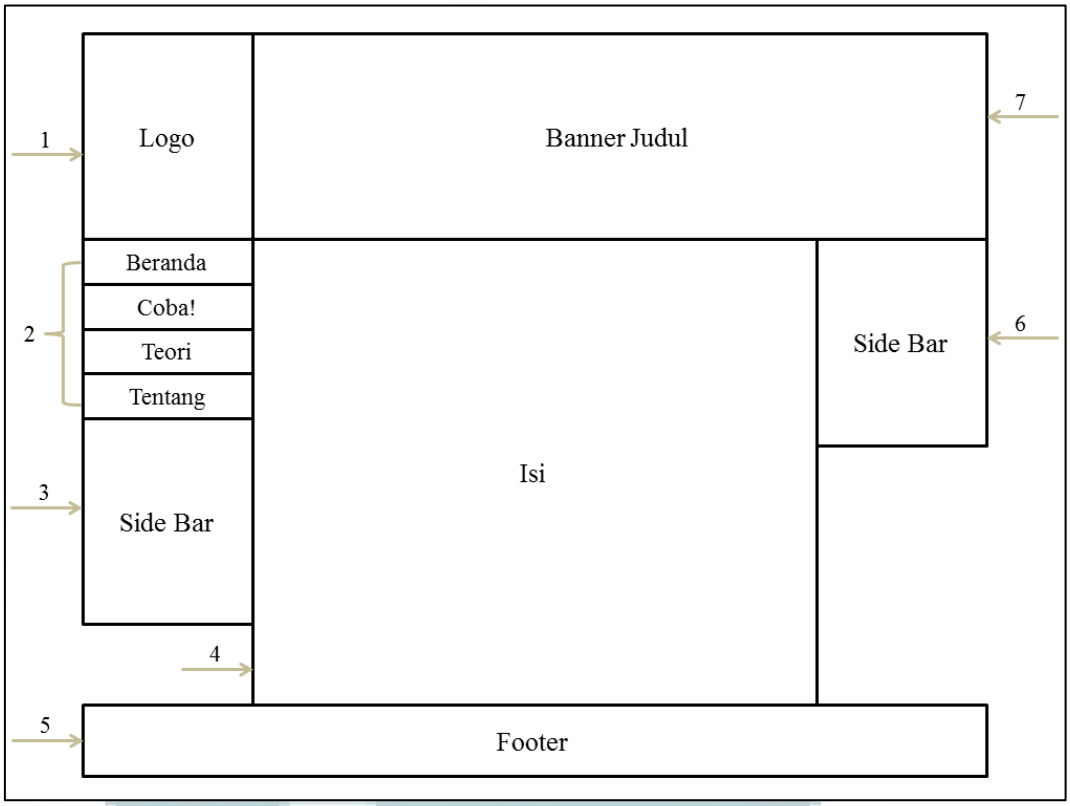
Variabel boxWidth didapat dari canvasWidth / temp.length sedangkan boxHeight didapat dari canvasWidth / temp[0].length. Variabel temp adalah *array* yang terbentuk pada Prosedur Cek Box. Misalnya isi temp adalah, [X,Y,X][-,X,-], jadi boxWidth adalah canvasWidth/2 (dua) dan boxHeight adalah canvasHeight/3 (tiga).

Kemudian dilakukan perulangan untuk cari posisi. Misalnya, path[0] adalah [2, X, Y], maka variabel unsur1 adalah X dan unsur2 adalah Y. Setelah itu dilakukan penggambaran garis setelah posisi didapat.

Proses berlanjut pada pemeriksaan *array* temp, jika temp[i][j] memiliki isi, maka lakukan proses penggambaran lingkaran (drawCircle).

3.2.2 Perancangan Antarmuka

A. Tampilan Halaman Website



Gambar 3. 21 Sketsa Rancangan Tampilan *Website*

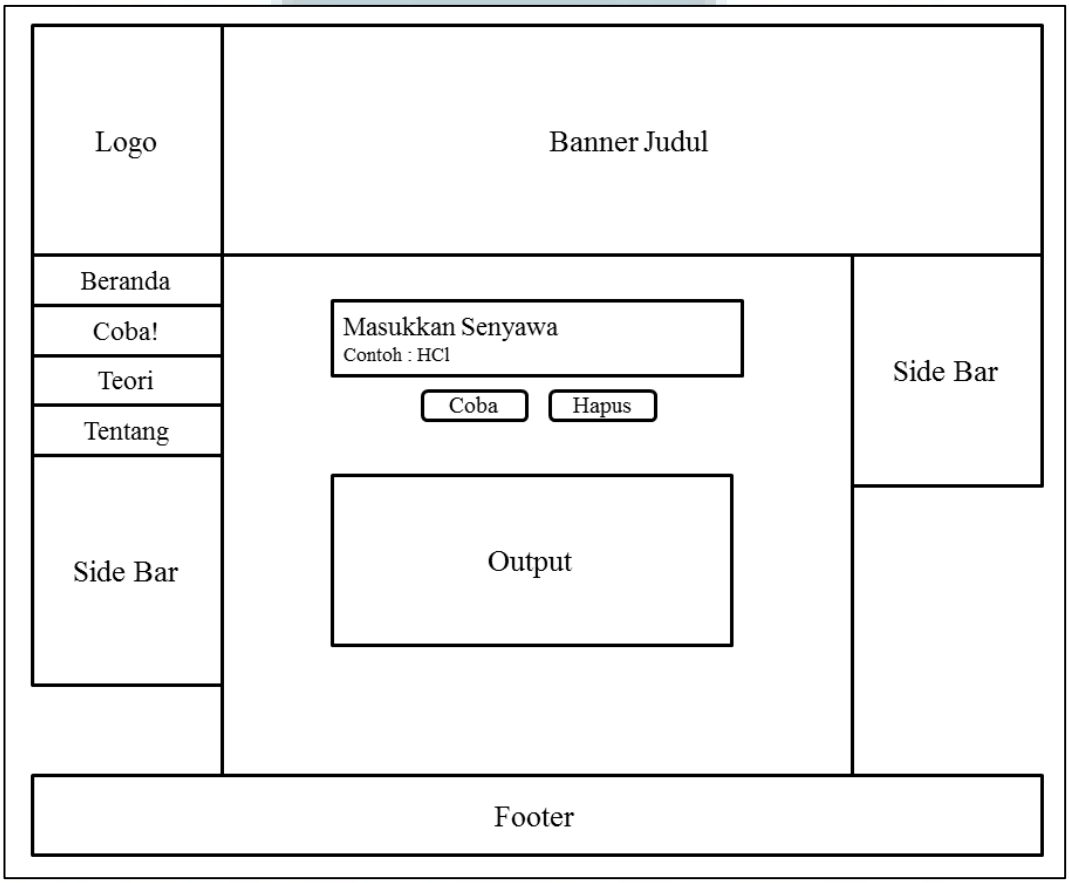
Gambar 3.11 menunjukkan gambar dari rancangan tampilan untuk halaman *website* secara keseluruhan. Tampilan ini memiliki komponen-komponen seperti yang ditunjukkan pada gambar dengan panah yang beserta dengan angka. Penjelasan dari komponen-komponen tersebut akan dijelaskan sebagai berikut.

1. Memuat gambar logo Universitas Multimedia Nusantara
2. Memuat pilihan-pilihan yang disediakan oleh *website* ini.
3. *Side bar* yang berisi *widget*
4. Kolom isi dari halaman yang dipilih. Isi akan berubah untuk halaman berbeda.
5. *Footer* dari *website*

6. *Side bar* yang berisi *widget*

7. Kolom judul yang berisi judul dari laporan hasil akhir.

B. Tampilan Halaman Coba



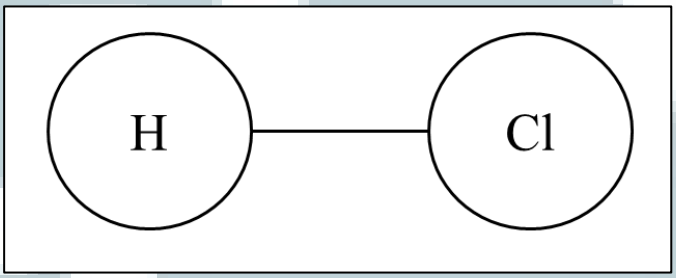
Gambar 3. 13 Sketsa Rancangan Tampilan Halaman Coba

Untuk halaman Coba akan ada kolom *text area* untuk memasukkan *input* dari senyawa yang dicari pengguna. Masukan dari pengguna haruslah benar untuk program bisa berjalan. *Format* dari masukan harus sesuai dengan kaidah penulisan senyawa, yaitu setiap unsur harus diawali dengan huruf besar.

Ada dua *button* yang tersedia di kolom isi halaman Coba, yaitu *button* Coba dan *button* Hapus. Jika menekan tombol Coba maka program akan menampilkan *output* atau hasil berdasarkan masukan dari pengguna sedangkan jika masukan

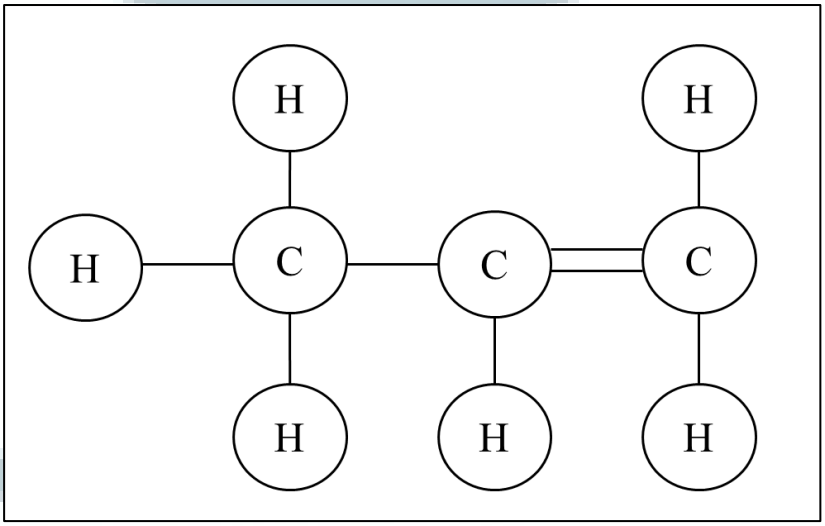
dari pengguna tidak *valid*, maka program akan memberikan pemberitahuan untuk masukan yang salah. Sedangkan, jika menekan tombol Hapus, maka *text area* akan di *reset* atau dihapus untuk pengguna dapat memasukkan masukan atau *input* yang berbeda.

C. Tampilan Kolom *Output*



Gambar 3. 14 Rancangan Tampilan Kolom *Output*

Pada kolom *output*, jika masukan *valid* atau benar, maka akan menampilkan tampilan seperti pada Gambar 3.13. Bentuk bulat, huruf dan bentuk garis yang ditampilkan dibuat memakai *canvas* yang tersedia di dalam HTML 5. Hasil dari tampilan kolom ini akan berbeda untuk masukan yang berbeda dari pengguna.



Gambar 3. 15 Rancangan Tampilan Kolom *Output*