

Struktur Data Non Linear
Laporan UTS 2 (Final)
Senyawa Kimia Hidrokarbon Dengan BST
Kelas B

Dosen Pengampu : Kartono Pinaryanto, S.T., M.Cs.



Oleh :

Antonius Yogi Prihantoro (165314006)
Dyline Melynea F (185314125)

Program Studi Informatika
Fakultas Sains Dan Teknologi
Universitas Sanata Dharma
Yogyakarta
2020

A. Tujuan

1. Mahasiswa mampu membuat implementasi program senyawa kimia dengan konsep pohon biner.
2. Dapat membantu pelajar dalam menentukan yang mana senyawa hidrokarbon.
3. Sebagai dictionary (kamus) dari senyawa-senyawa hidrokarbon

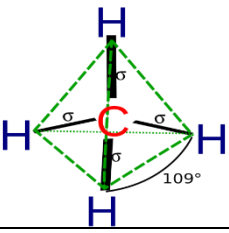
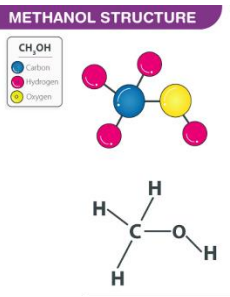
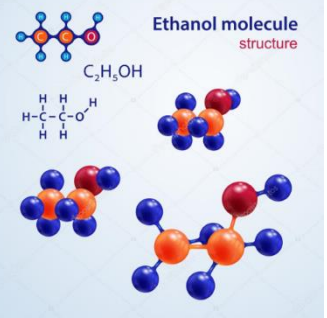
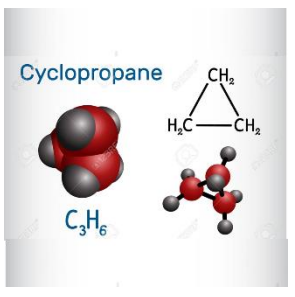
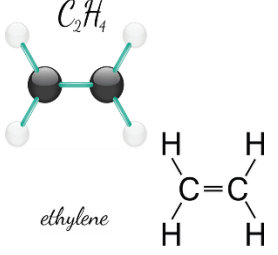
B. Penjelasan Topik

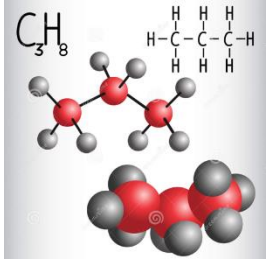
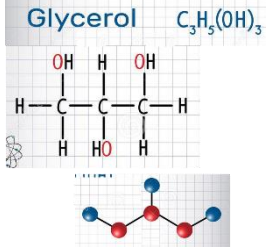
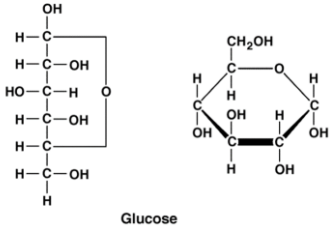

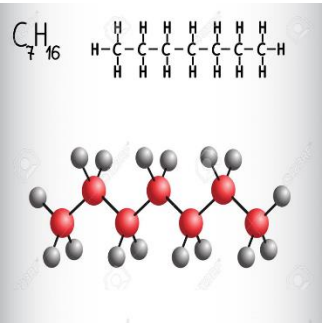
Senyawa merupakan suatu zat tunggal yang masih bisa dijelaskan menjadi 2 unsur maupun lebih. Senyawa juga mempunyai beberapa unsur yang saling berhubungan secara kimiawi, sehingga akan melambangkan senyawa terdiri atas beberapa lambang unsur.

Dalam kehidupan sehari-hari senyawa kimia sering kita temui, seperti senyawa campuran dan senyawa anorganik. Adapun berikut ini kami akan berikan sedikit contoh senyawa dalam kehidupan sehari-hari yang tergolong pada senyawa organik hidrokarbon yang mungkin sering kita dengar beserta rumus dan bentuk strukturnya dan juga kegunaan dari senyawa tersebut.

Program ini mengacu pada massa relatif dari masing-masing senyawa sebagai pembandingnya (nilai data pada tree untuk insert, search, dan delete).

Tabel senyawa untuk contoh input:

Rumus Kimia	Massa Relatif	Nama Senyawa	Kegunaan	Gambar Struktur
CH ₄	16	metana	Salah satu bahan bakar yang penting dalam pembangkitan listrik, dengan cara membakarnya dalam gas turbin atau pemanas uap.	
CH ₃ OH	31	Methanol	Kegunaan utamanya adalah dalam sintesis organik, sebagai bahan bakar, pelarut, dan antibeku. Metanol dapat digunakan sebagai antibeku, pelarut, bahan bakar, dan sebagai campuran untuk etanol.	
C ₂ H ₅ OH	46	Etanol	Digunakan pada: Bahan minuman beralkohol. Bahan industri untuk cairan pembersih (disinfektan)	
C ₃ H ₆	42	Propena	Digunakan dalam proses industri. misalnya, Propena adalah monomer Polypropene yang berkontribusi pada pembuatan Plastik seperti Botol, Film untuk kemasan, tutup Snap-on, dll. Juga digunakan untuk membuat propilena oksida, asam akrilat, alkohol okso dan isopropanol.	
C ₂ H ₄	28	Etena	Senyawa antara pada produksi senyawa kimia lain seperti plastik (polietilena). Etena juga dibentuk secara alami oleh tumbuhan dan berperan sebagai hormon. Ia diketahui terutama merangsang pematangan buah dan pembukaan kuncup bunga.	

C ₃ H ₈	44	Propana	Sebagai bahan bakar gas cair , LP - gas atau LPG.	
C ₃ H ₈ O ₃	92	gliserol	Pembuatan resin sintetis dan ester gums, obat-obatan, kosmetika, dan pasta gigi.	<p>Glycerol C₃H₅(OH)₃</p> 
C ₆ H ₁₂ O ₆	180	glukosa	Terdapat dalam nasi, tepung sebagai sumber energi makanan sehari-hari	 <p>Glucose</p>
C ₇ H ₆ O	106	Benzaldehida	Digunakan dalam kosmetik sebagai denaturant, zat penyedap, dan sebagai pewangi. Saat ini hanya digunakan dalam tujuh produk kosmetik, konsentrasi penggunaan tertinggi yang dilaporkan adalah 0.	<p>Benzaldehyde C₇H₆O</p> 
C ₇ H ₁₆	100	heptana	Penggunaan Heptana (dan banyak isomernya) banyak digunakan di laboratorium sebagai pelarut non-polar. Sebagai cairan, sangat ideal untuk transportasi dan penyimpanan. Dalam tes grease spot, heptana digunakan untuk melarutkan tempat minyak untuk menunjukkan keberadaan senyawa organik sebelumnya pada kertas bernoda.	<p>C₇H₁₆</p> 

C. Penjelasan Program

***yang digunakan hanya kelas MainScreen2, Senyawa, Tree, TreeNode pada package BST**

Class Senyawa

Sebagai wadah dari objek yang diinput, yaitu **Senyawa**. Pada kelas ini terdapat atribut-atribut dari method senyawa beserta konstruktor setter-getternya, dan method **toString()**. Yang perlu diperhatikan pada kelas ini adalah pada method **setMassaMr()** dengan masukannya yaitu rumus molekul. Pada method tersebut terjadi pemanggilan method **hitungMassa()** dengan masukan yang sama dengan method **setMassaMr()**. Pada **hitungMassa()** terdapat determinasi (penentuan) massa atom dari setiap unsur pada suatu molekul berdasarkan karakter yang dipisah dengan spasi.

Contoh:

C4H10 => C 4 H 10, yang di mana nilai-nilai unsur terkait pada hidrokarbon adalah sebagai berikut:

C = 12, H = 1, O = 16, N = 14, P = 31, F = 19, Cl = 35, Br = 80, I = 127.

Sehingga perhitungan Mr pada method **hitungMassa()** menghasilkan $12 \cdot 4 + 1 \cdot 10 = 58$.

Masukan dari method tersebut adalah rumus senyawanya (kode) yang masih dalam format berspasi. Sedangkan untuk method **setMR()** digunakan untuk proses delete/remove.

Pada kelas ini diberikan penanda juga jika data yang diinputkan bukan merupakan unsur-unsur di senyawa hidrokarbon dengan melemparkan jenis exception **NoSuchElementException**, yang kemudian lanjutannya diproses di kelas **MainScreen2** dengan **try-catch**.

Terdapat juga method **compareTo()** untuk membandingkan objek tipe **Senyawa** berdasarkan nilai MRnya yang kemudian digunakan di kelas **Tree** method **insertRecur()**.

Class TreeNode

Merepresentasikan pembentukan suatu objek node pada **TreeNode**. Terdapat atribut **leftChild**, **rightChild**, dan **parent** untuk tipe **TreeNode** berfungsi terhadap node yang dibentuk, atribut **senyawa** tipe **Senyawa** berperan sebagai data yang dibawa suatu node. Kemudian terdapat juga konstruktor tanpa masukan (default) dan yang dengan masukan bertipe **Senyawa**, serta method setter-getter dari atribut-atribut yang sudah dijelaskan sebelumnya.

Class Tree

Merepresentasikan proses pembentukan pada tree.

Pada kelas ini terdapat atribut **root** dan **rootHelper** yang bertipe **TreeNode**, dan **list** bertipe **ArrayList<Senyawa>** yang akan digunakan kemudian untuk traversal, dan **flag** bertipe integer berfungsi sebagai penanda akan scope mana/kondisi mana yang kejadiannya tidak menemukan data yang dicari.

Atribut **root** berperan sebagai node yang terbentuk (berjalan saat itu), **rootHelper** adalah kloningan (duplikat) dari **root** yang digunakan di method **lookup()**. Membutuhkan **rootHelper** agar tidak termodifikasi **root** (kejadian tidak diinginkan) yang sebenarnya setelah keluar hasil pencarian.

Berikut adalah method-method pada kelas **Tree**:

Method	Penjelasan
insert()	<p>Tipe masukan yaitu Senyawa, tipe method : void. Method ini memanggil method insertRecur() dengan masukan atribut tipe Senyawa turunan method ini dan node root untuk mengeksekusi insert objek. Pembuatan 2 method yang berfungsi untuk insert ini bertujuan untuk menggabungkan keseluruhan root setelah berekursif untuk menentukan posisi node terbaru yang masuk.</p>
insertRecur()	<p>Tipe masukan yaitu TreeNode dan Senyawa, tipe method : TreeNode. Terdapat banyak pengondisian di method ini. Dimulai dari apabila root parameter bernilai null, maka dibuatlah node terbaru (paling pertama masuk). Kemudian pengondisian jika atribut kode/formula senyawa untuk atribut s (nilai pada textfield terkait add/insert) sama dengan atribut kode formula senyawa rootnya, maka dimunculkan pemberitahuan bahwa ada potensi terjadinya duplikasi data jika tidak ditangani (mengecek apakah rumus molekulnya sama atau tidak, jika sama maka berpotensi terjadi duplikasi data). Kemudian jika tidak termasuk dalam pengondisian sebelumnya, terdapat else yang berisi serangkaian if-else mengenai penentuan lebih besar nilai MR atribut s atau root.</p> <ol style="list-style-type: none"> 1. Jika MR root lebih besar, maka melakukan rekursif method insertRecur() dengan masukan cabang kiri dari root. 2. Jika MR root lebih kecil, maka kebalikannya (rekursif method insertRecur() mengarah ke cabang kanan) 3. Jika MR root bernilai setara dengan MR s, maka dicek lagi di dalamnya apakah kode/formula senyawanya sama atau tidak. Jika sama maka menampilkan pemberitahuan duplikasi data, namun jika kode/formulanya tidak sama satu sama lain maka lanjut perjalanan rekursif insertRecur() dengan masukan cabang kanan (kanan diasumsikan untuk data terbaru \geq root). <p>Masing-masing ketiga kondisi di atas setelah diset cabang kiri/kanannya, diset pula parentnya dengan masukan root</p>

	(root menjadi parent , currentnya yaitu cabang kiri/kanannya root).
lookup()	<p>Tipe masukan yaitu String & double, tipe method : TreeNode.</p> <p>Method ini memanggil method lookupHelper() dengan masukan atribut tipe String, double yang merupakan turunan method ini dan node rootHelper untuk mengeksekusi pencarian berdasarkan mr.</p>
lookupHelper()	<p>Tipe masukan yaitu String, double, dan TreeNode, tipe method : TreeNode.</p> <p>Method lookupHelper() rekursif untuk mencari posisi pemegang data nilai mr dan yang tidak sama nama senyawanya di antara node-node dalam tree.</p> <p>Terdapat penginisialan variable int flag yang tujuannya untuk menentukan mana yang 0 (artinya data yang dicari ada di pohon), mana yang 1 (data yang dicari tidak ada di pohon).</p> <p>Ada beberapa kondisi</p> <ol style="list-style-type: none"> 1. Jika root==null, tidak ada aksi, namun nilai flag = 0 yang akan berpengaruh di tampilan MainScreen2 2. Jika posisi nilai MR (kunci/key) ketemu (nilainya sama dengan mr node) dengan syarat lainnya yaitu kode/formula senyawa nodenya sama dengan kode/formula pada textfield di MainScreen2 3. Jika nilai MR (kunci/key) lebih besar/sama dengan MR node. Maka pencarian dilanjut ke cabang kanan jika cabang kanan tidak null, namun jika null maka nilai flag dimodifikasi menjadi 1 sehingga menjadi penanda tidak ditemukan data di MainScreen2. 4. Jika nilai MR (kunci/key) lebih kecil dari MR node. Maka pencarian dilanjut ke cabang kiri jika cabang kiri tidak null, namun jika null maka nilai flag dimodifikasi menjadi 1 sehingga menjadi penanda tidak ditemukan data di MainScreen2. 5. Jika tidak memenuhi semua kondisi di atas, maka langsung memberi pemberitahuan "DATA TIDAK DITEMUKAN"
remove()	<p>Tipe masukan yaitu TreeNode & double, tipe method : TreeNode.</p> <p>Method ini memanggil method removeRecur() dengan masukan atribut tipe double turunan method ini dan node root untuk mengeksekusi penghapusan objek.</p> <p>Pembuatan 2 method yang berfungsi untuk delete/remove ini bertujuan untuk menggabungkan keseluruhan root</p>

	setelah berekursif untuk menghilangkan/melangkah node yang ditargetkan untuk dihapus.
removeRecur()	<p>Tipe masukan yaitu double, tipe method : void.</p> <p>Method ini memiliki beberapa kondisi yang di mana kondisi-kondisi percabangan paling luar selain else adalah tahap pencarian, belum penghilangan node/objek.</p> <p>Di else, terdapat pemecahan kondisi jika cabang kanan null atau cabang kiri yang null. Jika cabang kanan = null maka mengembalikan cabang kiri, jika cabang kiri = null maka mengembalikan cabang kanan yang kemudian akan tersusun ketika sudah balik ke method remove(). Selain masuk ke 2 kondisi tersebut artinya cabang kiri dan kanan tidak null, yang kemudian membutuhkan method minValue() untuk mencari suksesor dari node yang dihapus.</p>
minValue()	<p>Tipe masukan yaitu TreeNode, tipe method : double.</p> <p>Method ini bertujuan untuk mencari suksesor (cabang paling kirinya dari cabang kanan root) dari node yang dihapus. Maka dari itu di dalamnya dilakukan while-loop berjalan ke cabang kiri terus sampai dihentikan oleh kondisi sudah tidak ada cabang lagi. Kemudian nilai suksesor itu (datanya/objeknya) yang dikembalikan (returned).</p>
size()	Method size() untuk memanggil sizeHelper() method yang menghitung size tree.
sizeHelper()	<p>Metod sizeHelper() rekursif untuk mengembalikan jumlah kode yang disimpan dalam subtree yang terdapat pada root saat ini.</p> <p>return size subtree yang terdapat di root saat ini</p>
height()	Method height() untuk memanggil heightHelper() method yang menghitung jumlah tingkatan/level dari tree.
heightHelper()	<p>Metod heightHelper() rekursif untuk menghitung tingkatan subtree yang terdapat pada root saat ini.</p> <p>return height subtree yang terdapat di root saat ini</p>
inOrder()	<p>Tipe masukan: tidak ada masukan, tipe method : void.</p> <p>Method ini memanggil method inOrderH() dengan masukan node root untuk mengeksekusi pengunjungan node-node pada tree (traversal).</p>
inOrderH()	<p>Tipe masukan: TreeNode, tipe method : void.</p> <p>Method ini memanggil rekursif method inOrderH() / dirinya sendiri dengan ketentuan in-order mengunjungi cabang kiri-root-cabang kanan. Ketika di root, maka disimpan data yang ada pada node root itu ke ArrayList<Senyawa> untuk dapat ditampilkan di MainScreen2.</p>

Class MainScreen2

Menguji method-method yang terdapat pada class tree. Terdapat tombol-tombol fungsional seperti search button (mencari data senyawa berdasar rumus senyawanya), add button (input keseluruhan data senyawa), seeAll button (melihat senyawa-senyawa yang sudah masuk ke pohon), level button (melihat ada berapa tingkatan pohon senyawa hidrokarbon berdasar inputan), delete button (menghapus data senyawa berdasar data yang sedang dicari).

D. Penjelasan alur BST (analisis berdasar input)

***input** pada menu add diberi spasi di setiap unsur dan indeks unsur (contoh: ch4 -> c h 4)

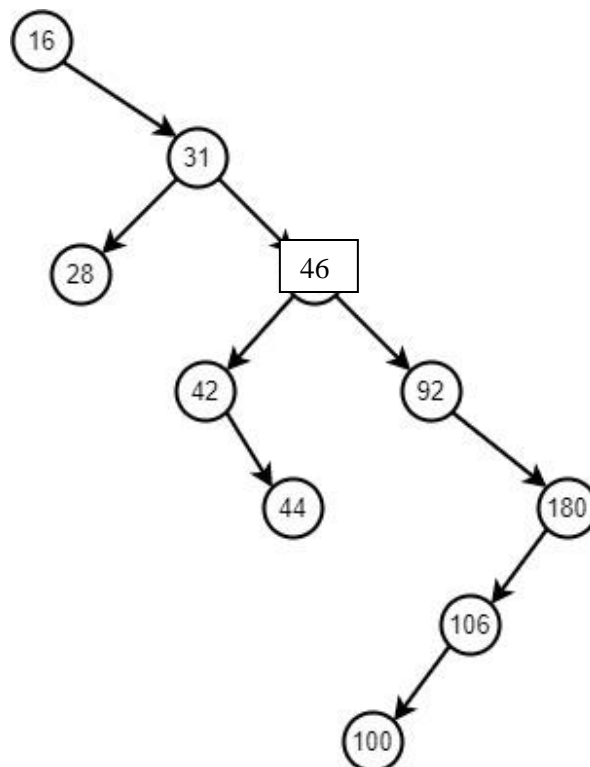
Jika asumsi root dimulai dari senyawa CH₄ (metana) dengan nilai MR yaitu 16, maka tree yang terbentuk adalah seperti ini

a. Insert

Insert:

Pada mulanya, data pertama yang dimasukkan akan menjadi root dari tree yang terbesar jika unsur-unsurnya adalah unsur hidrokarbon, apabila bukan maka akan muncul pemberitahuan tidak dapat insert. Kemudian setelahnya ditetapkan terlebih dahulu ketentuan untuk membagi subtree kanan dan kiri. Jika senyawa baru yang mau dimasukkan lebih kecil nilai MRnya dari nilai MR senyawa rootnya, maka senyawa baru tersebut dimasukkan ke cabang kiri. Sebaliknya, jika senyawa baru yang ingin dimasukkan nilai MRnya lebih besar/sama dengan nilai MRnya dari root, maka senyawa baru tersebut dimasukkan ke cabang kanan.

Masing-masing senyawa baru tersebut kemudian dianggap menjadi root di subtree-nya masing-masing yang nantinya berguna sebagai acuan perbandingan senyawa baru berikut-berikutnya untuk menentukan termasuk ke cabang kiri/kanan. Namun apabila terdapat 2x input rumus senyawa yang sama, maka insert tidak dianggap yang ke2 kalinya. Walaupun perbandingannya sangat bergantung pada nilai MR, tapi tetap disesuaikan dengan nama senyawanya. Karena terdapat kemungkinan berbeda senyawa namun memiliki massa relatif yang sama.



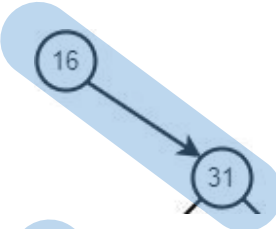
Urutan insert: 16, 31, 46, 42, 28, 44, 92, 180, 106, 100/CH₄, CH₃OH, C₂H₅OH, C₃H₈, C₂H₄, C₃H₈, C₃H₈O₃, C₆H₁₂O₆, C₇H₆O, C₇H₁₆

b. Search

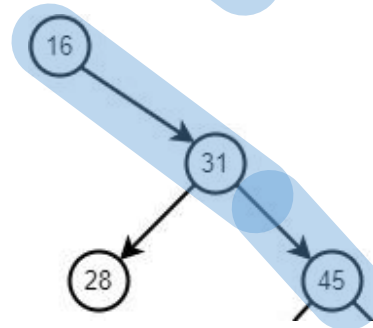
Konsepnya sama dengan insert, yaitu menggerakkan fokus dari root pohon terbesar ke root-root subtree agar dapat mencari suatu data.

Pada program, contoh kasus yang dicari misal yaitu C3H8.

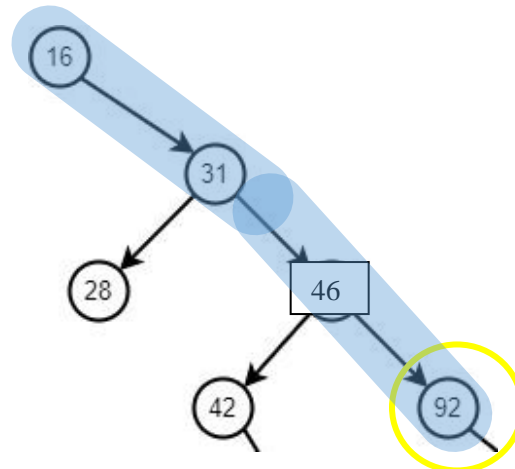
$92 \geq 16$, root.rightChild()



$92 \geq 31$, root.rightChild()



$92 \geq 46$, root.getrightChild()

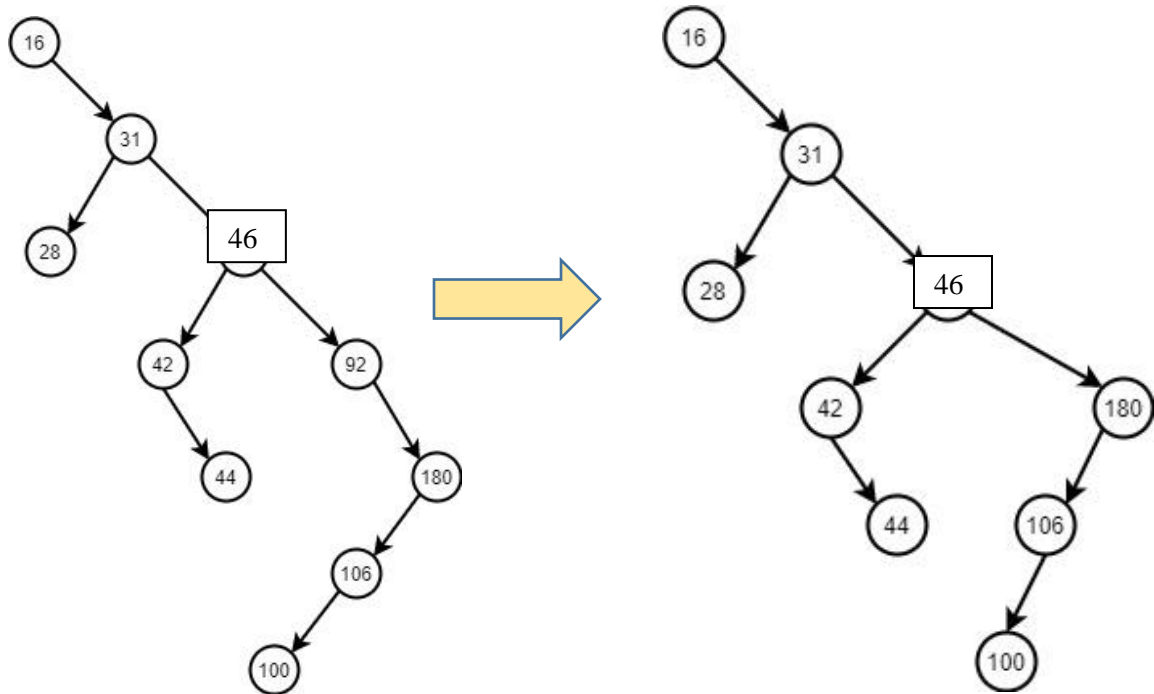


92 found, check with the textfield of kode senyawa in search menu.
Anggap 92 hanya masuk sekali karena C3H8 (karena tidak menggunakan mainclass untuk tes textfield jadi hanya bisa anggapan). Return nilai node (tipe TreeNode).

c. Delete

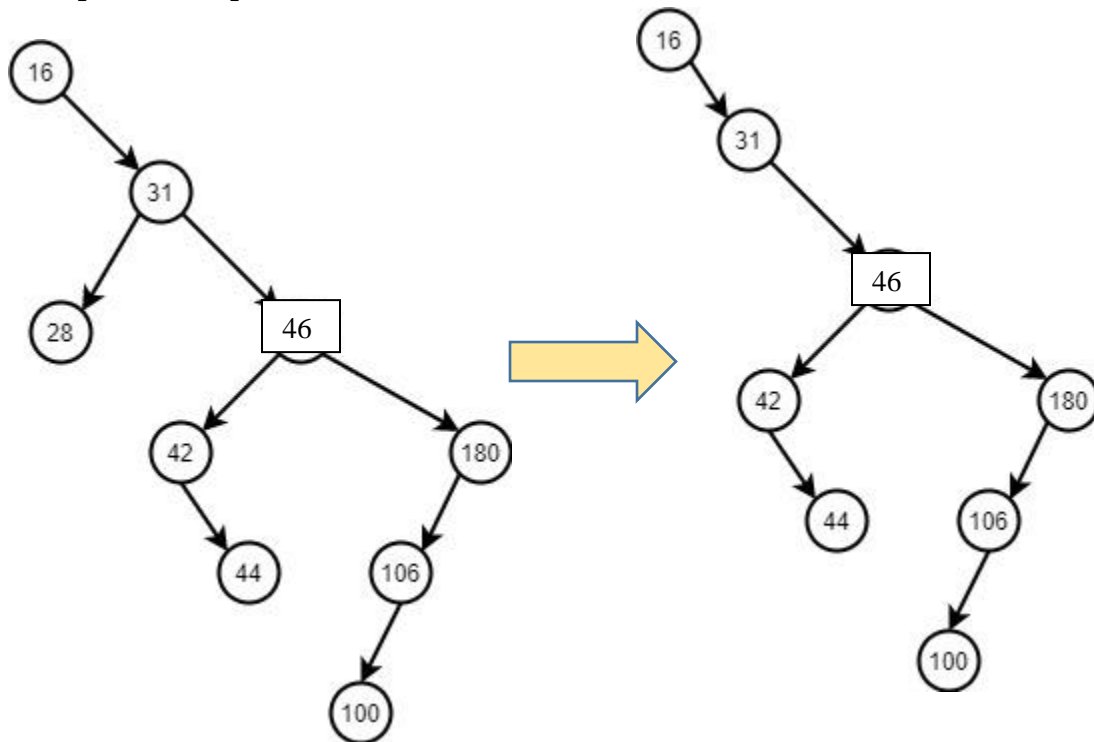
Dibuat Hapus 1- hapus 0 – hapus 2

Hapus : 92 (Hapus 1 anak)



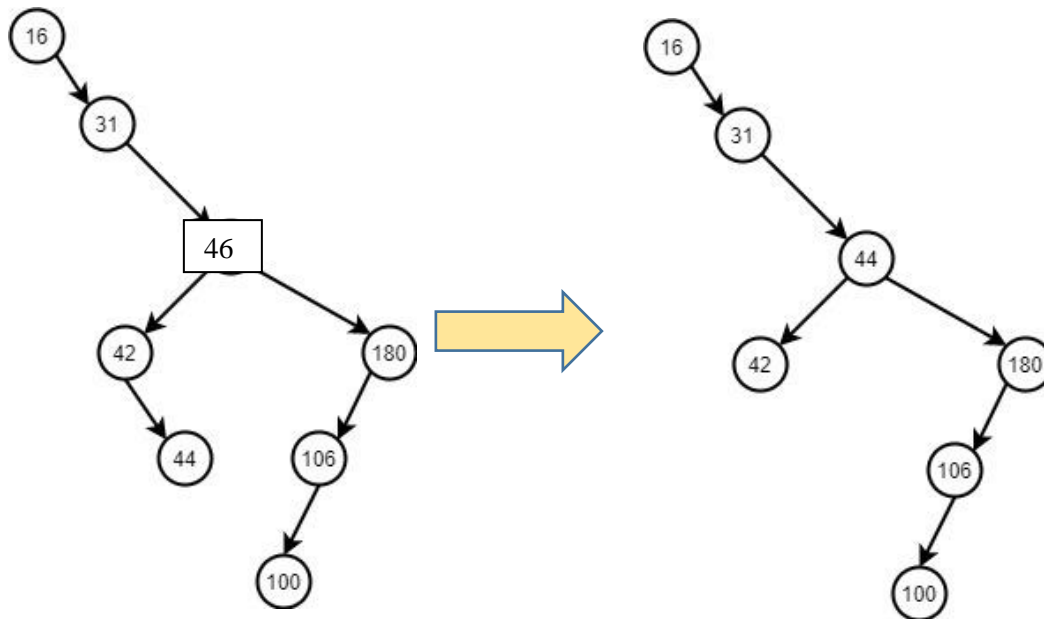
Dari tree diatas, menunjukkan bahwa node 92 yang memiliki 1 anak dan anak tersebut berada di rightChild dari parent 92 karena anak tersebut lebih besar daripada parent nya sendiri dengan nilai 180, maka yang akan dihapus pada Pohon Biner ini hanya node 92, karena hanya memiliki 1 anak. Method removeRecur akan mengeksekusi program didalamnya yang pertama adalah membuat variable bantu terlebih dahulu yang bertipe TreeNode dan berisi double key. key disini digunakan untuk mencari bilangan yang akan dihapus pada tree tersebut kemudian root nanti akan memberi sebuah nilai jika bilangan tidak ditemukan atau root bernilai null, maka akan mengembalikan nilai null karena bilangan yang akan dihapus tidak terdapat dalam tree, kemudian program akan mengecek node 92, apakah node 92 ada didalam Pohon Biner dan memiliki 1 anak atau tidak, jika ya maka node 92 akan terhapus dari Pohon Biner. Jika node 92 memiliki 2 anak maka node 92 tidak bisa terhapus. Kemudian kita cek parent dari node 92 yaitu 46 apakah node 46 memiliki 1 anak atau tidak, jika ya maka node 46 akan terhapus, jika tidak, maka node 46 tidak terhapus. Namun untuk node 92 ternyata hanya memiliki 1 anak, maka node 92 akan terhapus dari Pohon Biner. Maka Pohon Biner yang terbentuk akan seperti ini. Node 92 sudah terhapus dan node 180 pindah menjadi anak dari node 46.

Hapus : 28 (Hapus 0 anak)



Dari tree diatas kita akan membandingkan terlebih dahulu apakah $key <$ dari parent atau $key >$ parent, dikarenakan $28 <$ dari parent yaitu 31, maka program akan menjalankan kondisi dimana $key <$ root.data kemudian akan di cek apakah root merupakan sebuah leaf, jika ya node 28 merupakan sebuah leaf karena node 28 tidak memiliki anak, maka nilai tersebut akan bernilai null. Dengan begitu, hapus 0 anak berhasil dengan Pohon Biner seperti dibawah ini. Dengan terhapusnya edge node 28 dari pohon biner maka node 28 juga akan terhapus dari Pohon Biner, dan akan menampilkan Pohon Biner seperti dibawah ini. Node 28 sudah terhapus dari parent node 31.

Hapus : 46 (Hapus 2 anak)



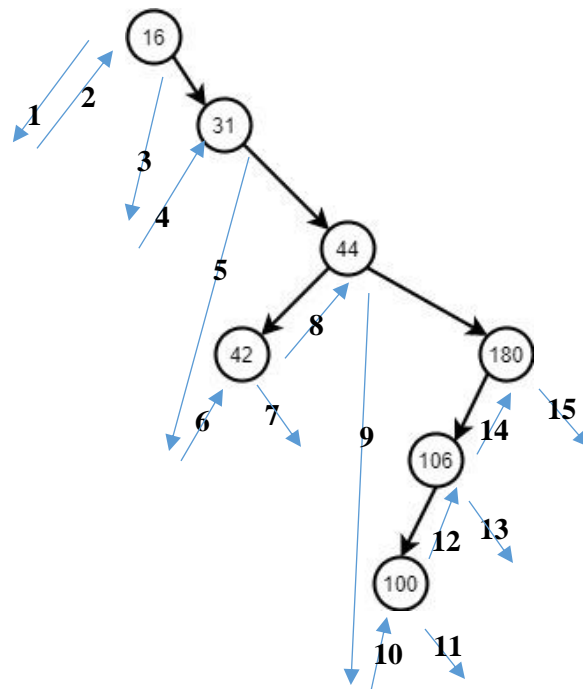
Dari tree diatas, menunjukkan bahwa node 46 yang memiliki 2 anak dan anak tersebut berada di leftChild dari node 46 dengan nilai 42 dan rightChild dari node 46 dengan nilai 180, yang akan dihapus pada Pohon Biner ini hanya node 46. Method removeRecur akan mengeksekusi program didalamnya yang pertama adalah membuat variable root terlebih dahulu yang bertipe TreeNode dan berisi double key. key disini digunakan untuk mencari bilangan yang akan dihapus pada tree tersebut kemudian root nanti akan memberi sebuah nilai jika bilangan tidak ditemukan atau root bernilai null, maka akan mengembalikan nilai null karena bilangan yang akan dihapus tidak terdapat dalam tree, kemudian program akan mengecek node 46, apakah node 46 ada didalam Pohon Biner dan memiliki 2 anak atau tidak, jika ya maka node 46 akan terhapus dari Pohon Biner. Maka Pohon Biner yang terbentuk akan seperti ini. Node 46 terhapus dan node 44 (elemen terbesar dari left sub-tree) pindah menjadi anak dari 31.

e. Traversal (in-order) setelah deletion/removal

Mencetak/membaca data/menyimpan data ke ArrayList dengan mengunjungi node-node yang jalurnya dari cabang kiri (dari yang menjadi daun) naik ke root terdekat kemudian ke cabang kanan dengan pola pengunjungan yang sama (dimulai dari cabang kiri ke root ke kanan jika memiliki child).

Pada method **inOrder()** dengan parameter, dikunjungi root, lalu masuk ke method rekursif dari method **inOrder()** dengan parameter masukan yaitu **leftChild**, dan jika pada rekursif berulang tersebut di bagian **leftChild** dari root tree yang besar sudah selesai dikunjungi dan tidak memiliki simpul lagi maka kembali ke tumpukan method sebelumnya untuk mencetak data yang ada pada simpul (masi dalam bagian **leftChild** root tree yang besar, namun dianggap sebagai root subtree). Setelah masing-masing sudah berperan sebagai root subtree yang akhirnya dicetak, selanjutnya method rekursif **inOrder()** dengan parameter masukan **rightChild** dari root tree terbesar yang berjalan sama seperti pola sebelumnya, berjalan dari **leftChild** jika memiliki sampe lagi semua berperan menjadi root subtree (artinya sudah dikunjungi) dan tercetak/tersimpan di ArrayList.

jalan: kiri-root-kanan



Urutan= 16→31→42→44→100→106→180

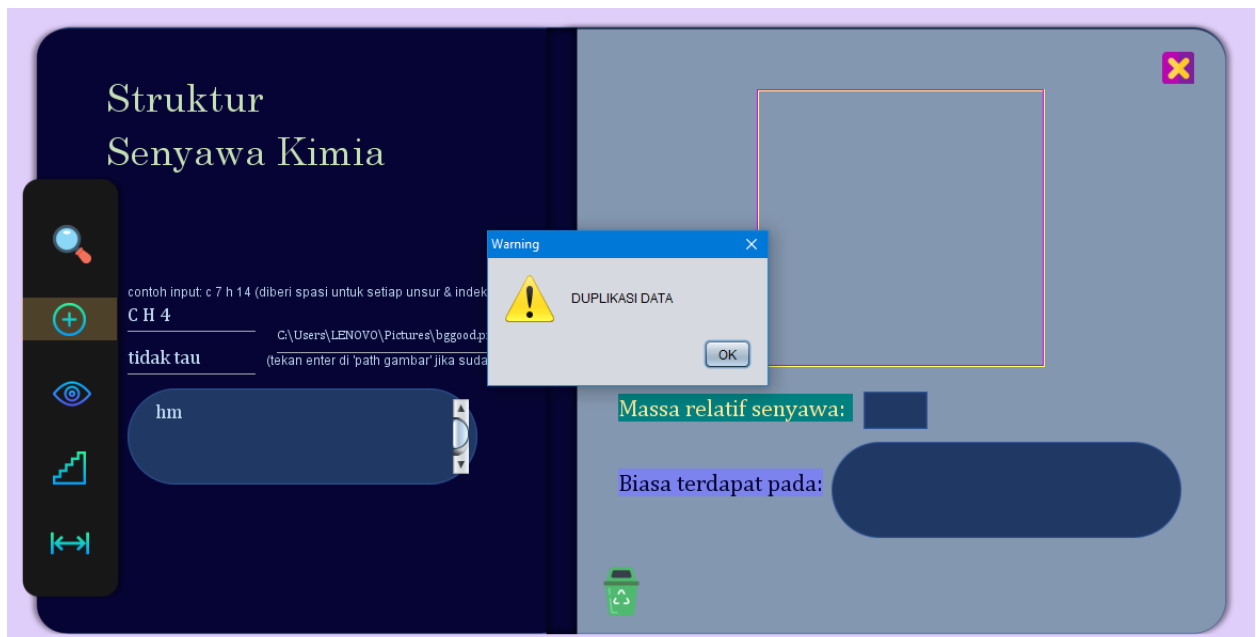
Input dan Output

Cek add

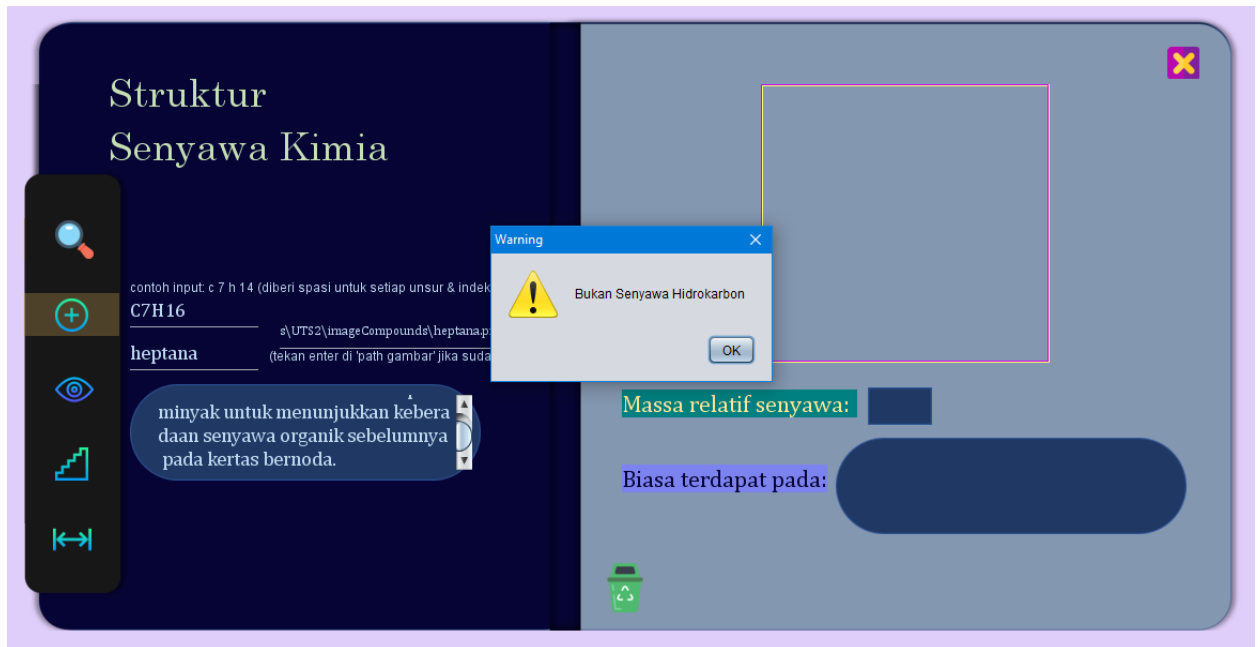
Input sesuai instruksi:



Input 2x rumus senyawa (mr akan bernilai sama, kode/rumus senyawa akan bernilai sama karena identik):

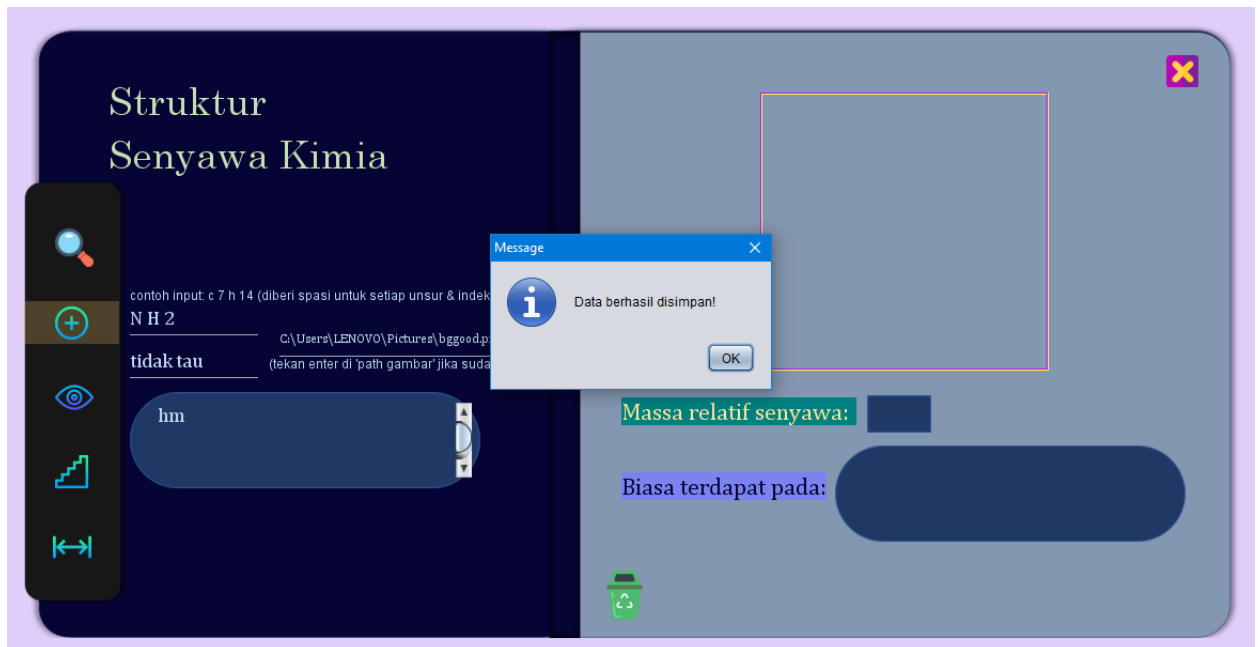


Input tanpa spasi:



Input mr yang sama dengan CH4 (CH4 sebelumnya sudah diinput)

$$\text{NH}_2 = 14 + 1 \times 2 = 16$$

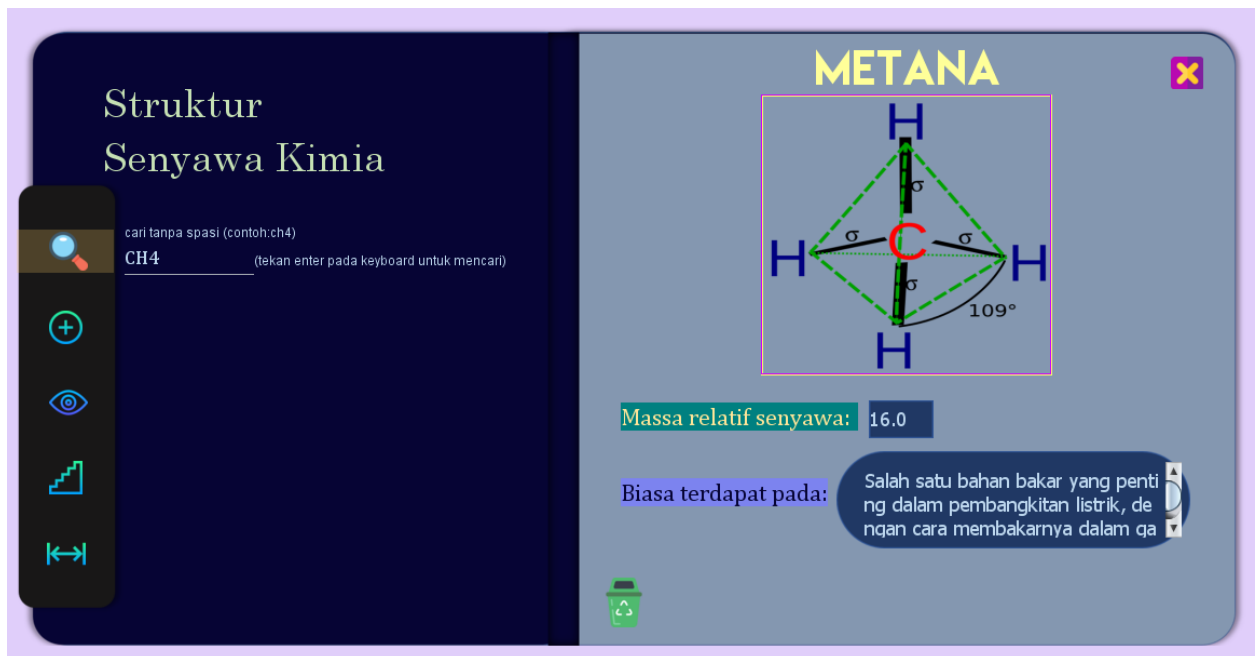


Cek search

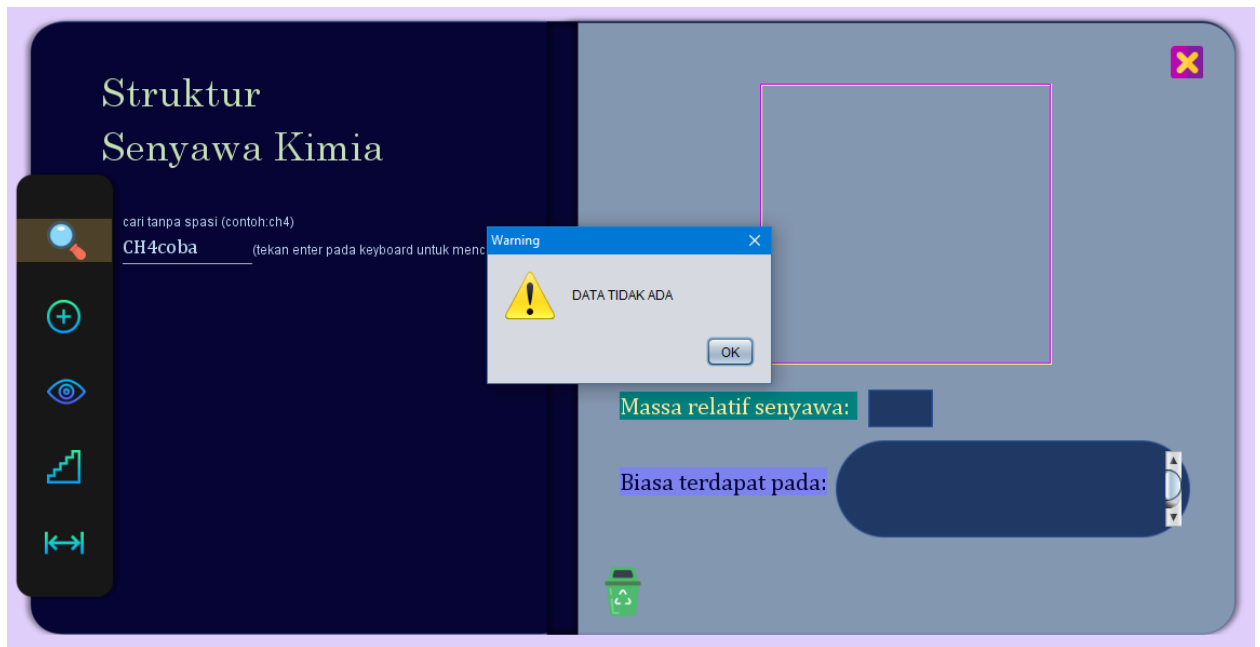
Tampilan mula-mula:



Setelah dienter:



Apabila tidak ada data tersebut di pohon:

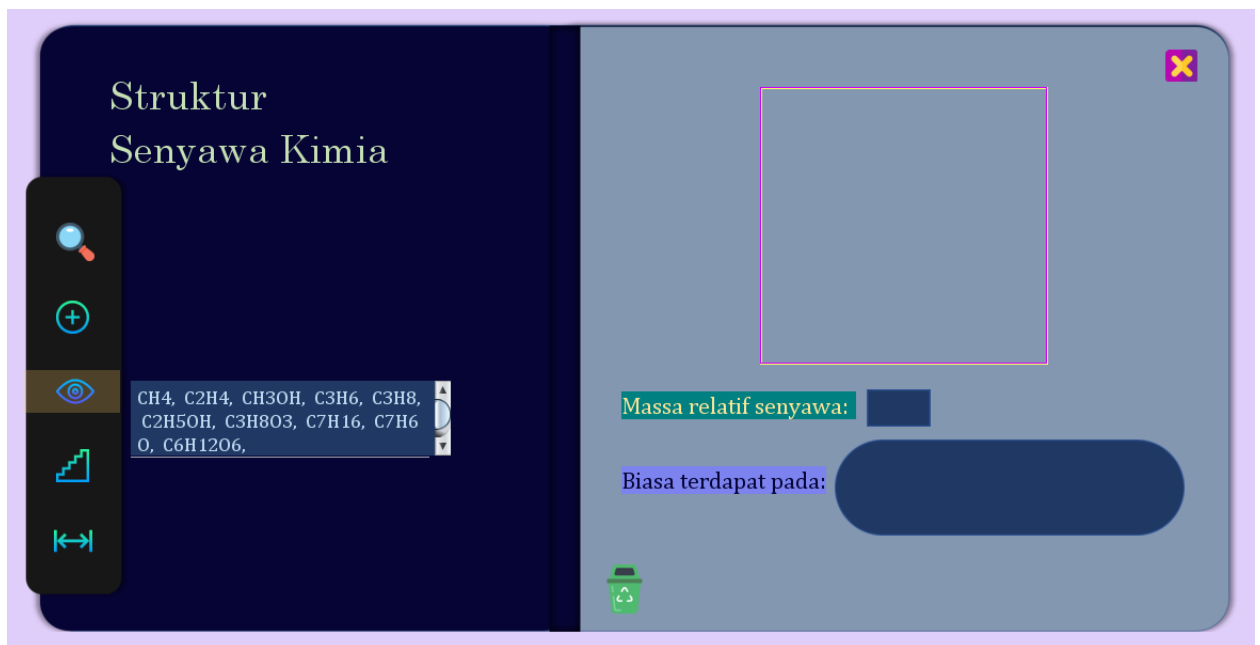


Cek see all (sudah diinput semua yang ada pada “tabel input senyawa”)

In-order berdasar traversal analisis bst bagian add (diliat dari ilustrasi):

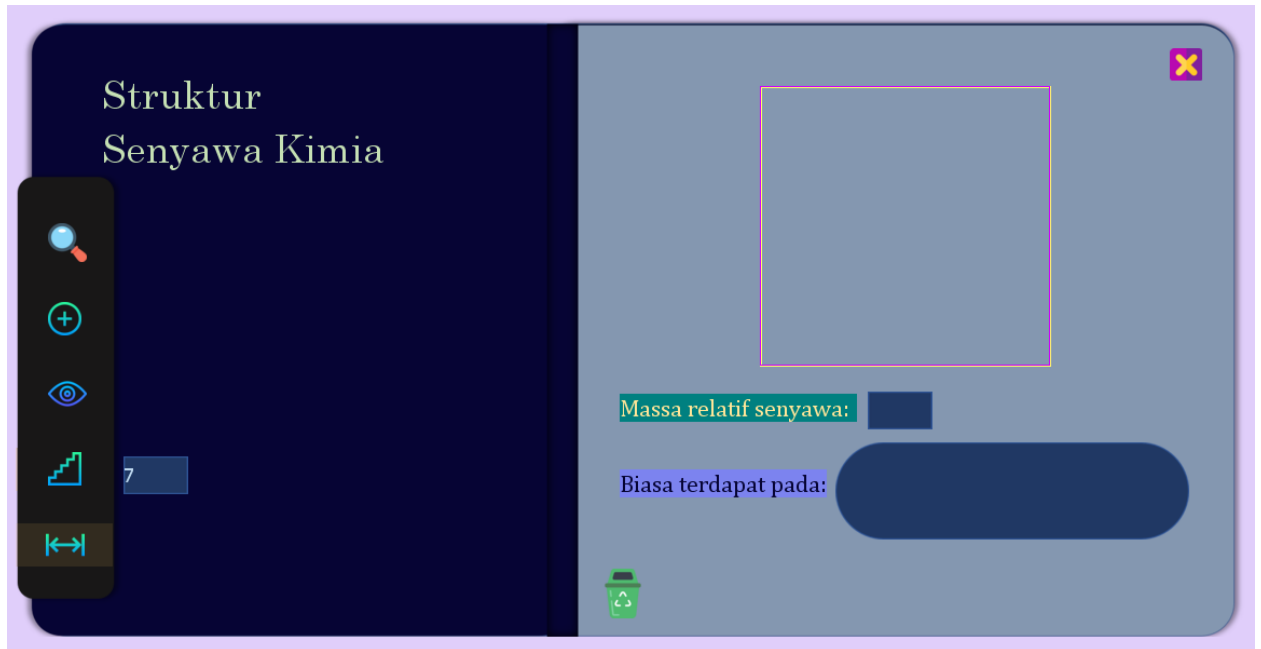
16, 28, 31, 42, 44, 46, 92, 100, 106, 180

CH₄, C₂H₄, CH₃OH, C₃H₆, C₃H₈, C₂H₅OH, C₃H₈O₃, C₇H₁₆, C₇H₆O, C₆H₁₂O₆



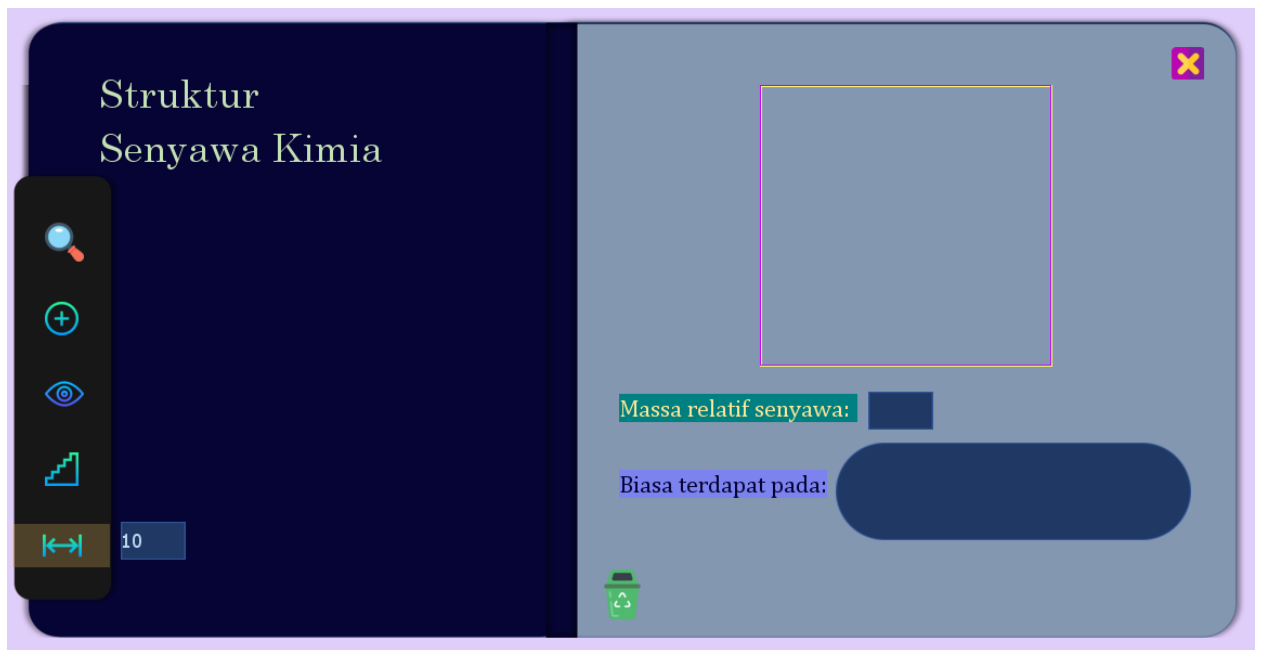
Cek level

Jika dilihat di ilustrasi analisis BST bagian insert, pohon memiliki 7 level.



Cek size

Data senyawa berjumlah 10



Cek delete

92, C3H8O3

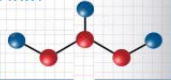
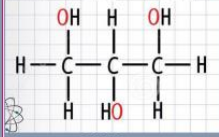
Struktur Senyawa Kimia

cari tanpa spasi (contoh: ch4)
C3H8O3 (tekan enter pada keyboard untuk mencari)

+
👁
📈
↔

GLISEROL

Glycerol $C_3H_5(OH)_3$



Massa relatif senyawa: 92.0

Biasa terdapat pada: Pembuatan resin sintetis dan ester gums, obat-obatan, kosmetika, dan pasta gigi.

🗑

Struktur Senyawa Kimia

C3H8O3 (tekan enter pada keyboard untuk mencari)

+
👁
📈
↔

Message

📘 Data C3H8O3 berhasil dihapus!

OK

Massa relatif senyawa:

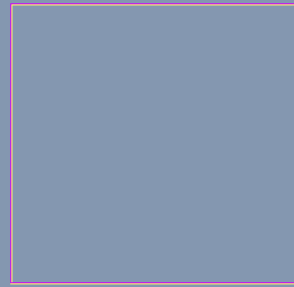
Biasa terdapat pada:

🗑

Struktur Senyawa Kimia



CH₄, C₂H₄, CH₃OH, C₃H₆, C₃H₈,
C₂H₅OH, C₇H₆O, C₆H₁₂O₆,



Massa relatif senyawa:

Biasa terdapat pada:

