Program to convert chemical structural formulae into their organic molecule names

Leo Jones

Candidate #: 8700

Centre #: 66553

13bIC1

Royal Wootton Bassett Academy

GIT Repo: <https://github.com/ChonkyDonkey/NEA-Molecule-namer>

Table of Contents

[Analysis 3](#_Toc163893876)

[Background 3](#_Toc163893877)

[Naming conventions 3](#_Toc163893878)

[Functional groups 4](#_Toc163893879)

[Process for molecule identification. 6](#_Toc163893880)

[Modelling of the problem and solution 6](#_Toc163893881)

[Challenges to overcome. 7](#_Toc163893882)

[Implementation analysis: 8](#_Toc163893883)

[Target implementation options 9](#_Toc163893884)

[Objectives 11](#_Toc163893885)

[Database 11](#_Toc163893886)

[Data volume 11](#_Toc163893887)

[Research into similar applications. 12](#_Toc163893888)

[Marvin JS 12](#_Toc163893889)

[Coding Software 12](#_Toc163893890)

[Log of research 12](#_Toc163893891)

[Design 14](#_Toc163893892)

[High level overview 14](#_Toc163893893)

[Data structures representation 17](#_Toc163893894)

[representation 17](#_Toc163893895)

[Description of algorithms 18](#_Toc163893896)

[Molecule 23](#_Toc163893897)

[System security and integrity of data 23](#_Toc163893898)

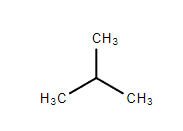
# Analysis

## Background

Many chemistry students of all educational levels must name molecules and it is easy to mess up. It can distract a researcher from key problems or aspects of an experiment that cause a health risk if they miss a functional group or misidentify it.

The project is used to solve the problem of naming organic molecules with their scientific names along with some extra information about it.

Input = CHHHCH(CHHH)CHHH

(where there is a join or turn on the diagram is another Carbon atom)

The structural formula will have to be manipulated to find the longest path to find the base chain. In this example 3 carbons long so propane.

And it has a methyl group on the second carbon in the base chain.

So it will output "2-Methylpropane".

### Naming conventions

Naming conventions refer to the universal way of naming organic molecules so it can be recognised throughout the world without much of a language barrier separating these standards and preventing the scientific development of the world. Which means if a molecule is misnamed the whole experiment could be nulled especially if dealing with larger and expensive equipment that costs hundreds of pounds per run.

For this project I will be using IUPAC nomenclature. The IUPAC rules can be found here <https://iupac.qmul.ac.uk/BlueBook/PDF/P1.pdf>

<https://www2.chemistry.msu.edu/faculty/reusch/virttxtjml/nomen1.htm> (08/04/24)

As you can see the naming conventions can get extremely complicated. The second link shows a less complicated and more directed for current need explanation of the nomenclature. Even with the more directed scope the difficulties would be too great for an A-Level project. So, I have made some reasonable adjustment to only include alkanes and methyl groups.

### Functional groups

Functional groups are sections of a molecule that help to determine the compounds properties.

The following functional groups are all part of the A-level chemistry spec.

**Key:**

* C = Carbon
* H = Hydrogen
* N = Nitrogen
* O = Oxygen
* R = Further Carbon Chain

|  |  |  |
| --- | --- | --- |
| Family | Functional Group | Plan to implement? |
| Alkanes | A basic carbon chain with no double bonds | y |
| Alkyl groups | An alkane that branches of to another side | y |
| Alkene |  |  |
| Haloalkane |  |  |
| Alcohol |  |  |
| Aldehydes |  |  |
| Ketone |  |  |
| Carboxylic Acid |  |  |
| Ester |  |  |
| Primary Amine |  |  |
| Nitrile |  |  |

These are obviously quite thorough, and it will not be practical to implement functionality to parse for all the groups, but I can treat them as a prioritised list of requirements.

Initially I am only implementing tests for alkanes and alkyl groups as they are simpler to describe as:

* they only include Carbon and Hydrogen atoms
* they all follow the same bonding rule as all the bonds are between carbon atoms (c-c).

Implementing the other functional groups will require more complexity due to handling C=C and the other elements like Oxygen and Nitrogen and their bonding rules as oxygen can bond to 2 different or the same atoms and nitrogen can bond to 3.all these require unique rules for both the naming and processing of the molecules.

Alkanes (straight chain)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| #carbons | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| name of alkane | methane | ethane | propane | butane | pentane | hexane | heptane | octane | nonane | decane |
|  |  |  |  |  |  |  |  |  |  |  |
| #carbons | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| name of alkane | undecane | dodecane | tridecane | tetradecane | pentadecane | hexadecane | heptadecane | octadecane | nonadecane | icosane |

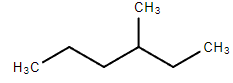
Alkyl groups (branches from the main chain)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| #carbons | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| name of alkyl | methyl | ethyl | propyl | butyl | pentyl | hexyl | heptyl | octyl | nonyl | decyl |
|  |  |  |  |  |  |  |  |  |  |  |

Process for molecule identification.

Here we will use a worked example which shows how molecules are processed.

Structural formula of a moleculeCH3CH2CH2C(CH3)CH2CH3

 this is the skeletal formula of this molecule

The longest chain consists of 6 carbons so the alkane will have the prefix of Hex-

There is another chain (alkyl group) branching from the third carbon from the right. Therefore, the alkyl position will be represented by 3

The alkyl group I only 1 carbon long so will be denoted as methyl.

Adding these factors up you get the full name: 3-methylhexane

## Modelling of the problem and solution

A molecule structure with letters and numbers

Description automatically generatedCH3CH(CH3)CH3
CHHHCH(CHHH)CHHH

C1
C2
C3
C4
H1
H2
H3
H4
H5
H6
H7
H8
H9
H10
C1
0
1


1
1
1







C2
1
0
1
1



1






C3

1
0





1
1
1



C4

1

0







1
1
1
H1
1



0









H2
1




0








H3
1





0







H4

1





0






H5


1





0





H6


1






0




H7


1







0



H8



1







0


H9



1








0

H10



1









0
Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
= longest chain calculator
Validity checker = counting the number
                                 of 1s connected to 
each atom


Set rules.

Carbon only has 4 adjacencies.

Hydrogen only has 1.

For the basic naming all that will be needed would be the Carbon nodes so a carbon adjacency matrix can be used but is compared to the hydrogens to verify if the structural formula is valid at the end

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 |
| 0 | X | 1 |  |  |
| 1 | 1 | X | 1 | 1 |
| 2 |  | 1 | X |  |
| 3 |  | 1 |  | X |

This will help with the abstraction of the problem.

### Challenges to overcome.

1. Finding the longest carbon chain then number each carbon.
2. Use the right chain based on the functional groups present. (only if I were to expand into further functional groups than alkanes and alkyl groups)

A picture containing diagram, line, design, origami

Description automatically generated

1. Validating the molecule to make sure each element has the correct number of attachments.

### Implementation analysis:

After discussions with my chemistry teacher, we have agreed on a chemical namer that takes an input of a structural formula of a molecule. It should then output the IUPAC name for said molecule.

To research this problem, I have searched multiple molecule namers on the internet. Finding sites like

* <https://www.chemspider.com/StructureSearch.aspx>
* <https://web.chemdoodle.com/demos/iupac-naming>
* <https://pubchem.ncbi.nlm.nih.gov>

These are all great namers however they all use preexisting databases of all the molecules. I will not have access to any of these databases to uses in my project so I will need to develop a method of dynamically parsing molecules.

After discussions with my computer science teacher on the feasibility of this project I have decided to make a few reasonable adjustments.

* Instead of making a namer for all molecules. I will only make a namer for alkanes and alkyl groups.
  + This is to reduce the quantity of the project because there are far too many rules to be implemented to be coded for in the given time.
* As I will not have access to a database of molecules, I will have to change my approach from a database look-up to a from scratch longest path algorithm and more.
* I will also need to find a way to model the molecule in such a way that a computer can understand.

With this project idea I have now got to start to think about my approach to coding and the feasibility of the algorithm. For the development side of my project I had 2 ideas, the first being test after creation. The second being test-driven. Test after creation can be useful to get a base line code as it is easy to see improvements at the start. However, it gets more difficult when you enter the more detailed end of the code. This is particularly the case when you have multiple rules that need to be followed, which can introduce edge cases. I have opted for a test-driven development.

Test driven development is the idea of setting test cases and making code that will produce the desired outcome. This helps with abstraction of the algorithm as I only need to think about the current test case I am working on. This method requires a good understanding of the expected outcome and inputs to understand the test cases needed from the start of the project. Having a large body of tests provides evidence that the code still works while we refactor and extend it.

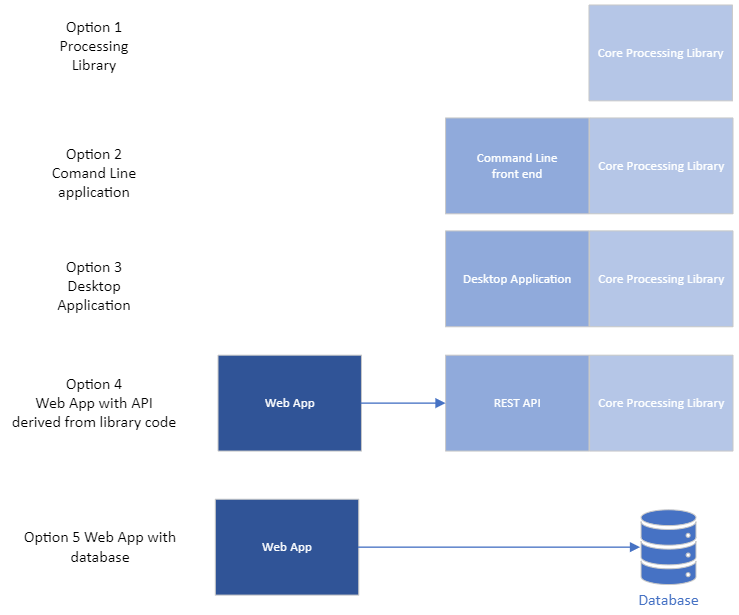
On the basis from the above my idea whittles down to this:

* A molecule namer that takes structural formula and converts it into its IUPAC name, initially restricted to alkanes and alkyl groups.
* It will do this by:
  + converting the structural formula through several processes into an adjacency matrix.
  + I will then traverse the adjacency matrix to find the longest path
  + I will then search the matrix to find any branching points from the longest chain
  + I will then apply rules to identify functional groups in the branches in the molecule. This will use iteration and recursion to produce the longest path on branches
* The development will be test-driven. This can be done simply by creating and running test cases on Visual studio code (VSC).

### Target implementation options

There are a few options for consideration when determining the target software architecture.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Option | | Description /Tech implementation | Pros | Cons |
| 1 | Processing Library | By implementing as a library can be included as part of a broader app.  Can be implemented in C# | Fundamental to multiple other options  Provides a framework for adding functionality |  |
| 2 | Command Line application | Implement in C# invoking the core library | Simple to implement |  |
| 3 | Desktop Application | Implement in C# invoking the core library | Allows more complex presentation of molecule and data | Additional complexity |
| 4 | Web App with API derived from library code | Will require HTML/CSS / JS  Will require adding a REST API to the core library  Introduce a login system? |  | Additional complexity due to multiple languages  Requires authentication |
| 5 | Web App with database | Will require HTML/CSS / JS  Requires DB created which is prepopulated with all of the molecules.  Introduce a login system? | Simpler to implement as pre  Complex to prepopulate | Will require addition of a database and prepopulating.  Requires authentication |

Visually these can be shown as:  


Most all of my examples I have used in my analysis have used option 5 as it is the easiest to implement however, I don’t have access to the databases required to create such a program. Though ultimately it would be nice to present this as a Web App in some form, my initial focus is on building the core processing library, as this is a common feature for most of the further options.

Due to the versatility of directions this could be taken in the future I have decided to create a minimum viable product. This fulfils the project objectives but still allows for adaptation in the future if it is needed. This is also the case because I will not be able to name all molecules accurately but the program should be able to name compounds to a high enough degree of accuracy that it is useable.

### Objectives

|  |  |  |
| --- | --- | --- |
| **Priority** | **#** | **Description** |
| Must | 1 | Be able to receive inputs of structural formula.  Be able to convert it into a carbon and bracket only string.  Validate that the input string is a valid formula. |
| 2 | Identify longest carbon chain.  Via using adjacency matrices.  Connect the respective carbons (arcs)  Able to work around/with “(“and”)” |
| 3 | Find functional groups location on carbon chain.  a. Number the final carbon chain to find where the extra alkyl groups attach.  i. Recursively check arcs for the carbon to find alternative routes to count as alkyl groups.  ii. Find the longest chain from the root node that doesn’t follow the original longest chain.  iii. This must be repeated in case you get alkyl groups on alkyl groups. |
| 4 | Output the correct name for the molecule stated.  This should be based on the IUPAC nomenclature. |
| Could | 1 | Identify:  Alkenes (C=C)  Aldehydes (R-C=O-H)  Ketones (R-C(=O)-R)  Carboxylic acid (R-COOH) |
| 2 | Give the MR of the molecule.  By having a dictionary of elements and their Relative Atomic Mass |
| 3 | Give the ways the molecule can react. |
| Wont | 1 | Add a drawing GUI. |
| 2 | Get an image of the molecule named. |
| 3 | Create a Database |

## Database

At level I will be doing with my project, a database would be out of scope.

## Data volume

No database will be needed as well as current project isn’t using any web API so data volume is an unnecessary scope to my project

## Research into similar applications.

### Marvin JS

One library I found was [Marvin JS](https://chemaxon.com/chemical-naming-and-structure-conversion) (Marvin, 2023). Their identification and naming process goes through a large database of very many molecules. This seems to be a bit too inaccessible to me and doesn’t follow the pathway I am wanting to follow to name the molecule. Marvin JS also supports naming molecules from a drawing input on a premade molecule drawing application. I was planning to use this for my original project scope, but I deemed this to be too much for one project.

Issues with using graph searching algorithms for shortest paths is it is difficult to find similar examples online. This is because any search algorithm online only shows a single output. This differs to my requirements because I will need to find all possible routes while blocking off the routes that have already been taken without blocking paths that I haven’t searched yet (This is elaborated further in the “description of algorithm section”). Because of this every possible path is needed to be searched making the O complexity of the algorithm into polynomial time O(n^2) or at least very close.

## Coding Software

I have decided to use C# for this project as I have been learning it for the last year and it is a high-level language with many libraries and public help for programming issues.

This has then left me with a question of what software to use. There were 2 that I chose from. Visual Studio 2022(VS) and Visual Studio Code (VSC). Both are good systems that gives you all the high-level commands. But ultimately, I chose to go with VSC because It has easier access to Git Hub as well as the possible addition of extensions. The extensions I have used are:

|  |  |
| --- | --- |
| .Net Install Tool | is needed for C# and C# Dev Kit to allow them to install local versions of .net. |
| C# | allows me to code in C#. |
| C# Dev kit | Introduces the test abilities which allows me to easily test possible test cases and display the output in a nice format. (very useful for test-driven development) |
| CIntelli Code for C# Dev Kit | A convention checker that helps me to maintain convention for variable names and use correct instantiations |

# 

# Design

## Class diagram

A white sheet with black lines and text

Description automatically generated with medium confidence

## High level overview

A blue rectangular object with white text

Description automatically generated

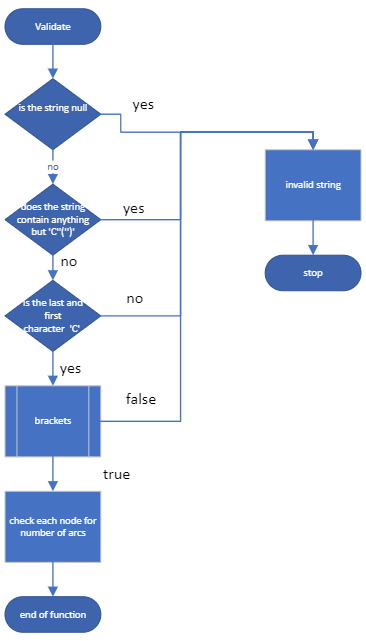
You start by getting an input of the structural formula. I will then convert the string to a string comprising of ‘C’, ‘(‘, and’)’. This is done to make it easier to implement the code, as identification of alkyl groups only have to consider the bonds between carbon atoms.

**Note:** when the code is extended to further functional groups, hydrogen atoms will need to be considered.

### Validating input data for security

Using the VSC devkit test feature it has allowed me to create all possible situations for inputs by using bounds testing and having cases that are meant to fail and some that are meant to pass.

In my code I have created a validation section which is supposed to invalidate any invalid inputs and stop them from entering the program and causing errors that are undesirable. This prevents invalid inputs that can disrupt the code.



The sub routine validate is then called. First thing it does is a presence check which if the string comes back as null the code recognises the string as invalid and ends the code. Any invalid string will fall the same fate and cause the program to end. The next step after the presence check is checking the only characters in the string are the ones mentioned above. If they are the code, then checks the first and last characters to check if they are a ‘C’. this is done to follow the syntax I have set out to allow the complexity of the code to decrease otherwise the complexity would go beyond A-Level complexity.

A diagram of a system

Description automatically generated

The brackets (validation) subroutine will then be called first thing it checks for is an even number of brackets. Then it checks that the brackets are all paired correctly otherwise the syntax is wrong and can cause crashes. Then it checks that ‘(‘is the first type of bracket in the string. This will then return to the validation subroutine.

### Extracting molecule structure

Process Molecule is then called by the main flow chain

A diagram of a number node

Description automatically generated

This identifies the number of nodes / atoms in the formula and labels them (based upon position in the string). It then works to find the arcs/connect between nodes and adds them to the graph representation.

A blue rectangular object with white text

Description automatically generated

The next section then checks for the connections between Carbons and makes notes of it making sure that a carbon can’t have too many bonds/arcs to it. The code then returns to the main program.

Find arc left will identify atom bonds/arcs when parsing the formula string from left to right.

Find Arc Right identifies atom bonds/arcs when parsing the formula string from right to left.

The “find arc left/right” has a lot of intricacies that I will sort out in the code but are too low level for this overview.

Once we have built the graph representation of the molecule, for convenience we convert it to an adjacency matrix, which is easier to traverse.

### Molecule Classification

A screenshot of a diagram

Description automatically generated

As per my final design this is the longest path algorithm.

It starts by creating an empty list of lists which will serve as a storage for all finished paths (note, these paths may not be fully developed but through the process longer ones have been made so nullify any possibility of an error being made due to unfinished chains. Except the potential for holding more data then necessary.) the code will then “Find paths” from a root node and add them to the list. Then it handles the possibility that the root node isn’t the last or first node in the longest chain by using the “mergeRoute” function. The code will then iterate through the routes collected in the allRouteCombinations to find the longest route and making note of it.

A diagram of a network

Description automatically generated

This code will straight away add the current node to the route. It then will check through the current nodes row on the adj matrix to check if there are any neighbours that aren’t in the main chain (to prevent going back on self). If there are more neighbours to search the program will then iterate through the neighbours and recurse the “FindPaths” subroutine. This continues until there are no valid neighbours where the code will then add the path to “allRouteCombinations” this will repeat until there are no more possible routes from the root node.

A diagram of a process

Description automatically generated

This subroutine will be called, and it will iterate through all the found lists so far and find the common nodes and cut and stick routes together to potentially unveil routes that are longer. This deals with the possibility that the root node is either not in the main chain or not at either end of the main chain.

This has a more in-depth explanation further on.

## Data structures

### Molecule representation as a graph

The molecule will be inputted, and the software will need to rewrite it in a way it can understand.

The initial representation is as a graph structure which is made up of nodes (the atoms) and arcs which represent the links between them.

A diagram of a diagram

Description automatically generated

Note: the arcs can have a weight associated with them, however for simplicity this is always 1 in the current implementation.

The class diagram for this structure is as follows:

A graph with text and numbers

Description automatically generated with medium confidence

The graph representation is useful because it can be built as we gradually parse the chemical structure and identify atoms and their linkages without needing to know what the overall structure is.

### Molecule representation as an adjacency matrix

To simplify processing the graph processing, it is useful to convert the graph to another format – that of an adjacency matrix which is a two dimensional array.

Int adj\_matrix[starting nodes][destination nodes]

**Note:** the two dimensions are identical as they are the full set of all the modes in the molecues.

The value in each cell of the array indicates whether there is a connection/arc between the starting node and destination node. This is using the weight value from the arc. Note: Atoms cannot connect to themselves, and we represent this invalid arc with an x.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | Source Node | | | |
| C1 | C2 | C3 | C4 |
| Destination Node | C1 | x | 1 | 0 | 0 |
| C2 | 1 | x | 1 | 1 |
| C3 | 0 | 1 | x | 0 |
| C4 | 0 | 1 | 0 | x |

### A more complex Adjacency Matrix

For this project I will need to convert the structural formula to what I call extended structural formula then to an adjacency table as shown in the table below e.g. CH3CH(CH3)CH3

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **Starting Node** | | | | | | | | | | | | | |
| **C1** | **C2** | **C3** | **C4** | **H1** | **H2** | **H3** | **H4** | **H5** | **H6** | **H7** | **H8** | **H9** | **H10** |
| **Destination Node** | **C1** | x | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **C2** | 1 | x | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **C3** | 0 | 1 | x | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| **C4** | 0 | 1 | 0 | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| **H1** | 1 | 0 | 0 | 0 | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **H2** | 1 | 0 | 0 | 0 | 0 | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **H3** | 1 | 0 | 0 | 0 | 0 | 0 | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **H4** | 0 | 1 | 0 | 0 | 0 | 0 | 0 | x | 0 | 0 | 0 | 0 | 0 | 0 |
| **H5** | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | x | 0 | 0 | 0 | 0 | 0 |
| **H6** | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | x | 0 | 0 | 0 | 0 |
| **H7** | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | 0 | 0 | 0 |
| **H8** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | 0 | 0 |
| **H9** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | 0 |
| **H10** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x |

Issues with this method is it makes it difficult to validate the molecule that have double bonds in them.

// don’t need hydrogens why? For alkyl case do not need to consider hydrogen.

This depiction can be simplified for current use by only showing the carbons.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Starting Node** | | | | |
| Destination Cell |  | C1 | C2 | C3 | C4 |
| C1 | X | 1 | 0 | 0 |
| C2 | 1 | X | 1 | 1 |
| C3 | 0 | 1 | X | 0 |
| C4 | 0 | 1 | 0 | X |

### Prefix-Naming Dictionaries

Within the naming we would also need a few dictionaries to provide some lookups for different naming conversions.

The following one is used when I am referencing the number of carbons in the chain with the prefix needed to be given:

A screen shot of a computer code

Description automatically generated

Th following dictionary is used to convert from the words to the number of carbons. This is useful after we put the alkyl groups in alphabetical order so it can be correctly named:

**A screen shot of a computer code

Description automatically generated**

This dictionary is used when multiple of the same alkyl group is used (number of times alkyl group is used, multiplicity prefix):



All of these can be expanded for larger molecules however I deemed it sufficient with the given number of possibilities this provides.

And I modified the classes more efficiently in more organised classes. I will also adjust the node creations to be dynamic to an input based on the number of “C” in the string. I also adjusted the arc creation to follow rules to correctly create arcs based on a string input.

I took a skeleton code from stack overflow that made it easier to visualise the adjacency matrix that I would be using. As well as creating a basis that I could easily build upon. (<https://stackoverflow.com/questions/15306040/generate-an-adjacency-matrix-for-a-weighted-graph>)

## Description of algorithms

### Find Longest Path:

This process is rather simple. Or so I thought, as of now I have spent around 6 lessons purely working on these problems using graphs, adjacency matrices, search algorithms etcetera. And now I finally think I have 2 possible solutions to the problem. Initially when I started this problem, I thought an adjacency matrix and a 20-line code would solve my problem.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | C1 | C2 | C3 | C4 | H1 | H2 | H3 | H4 | H5 | H6 | H7 | H8 | H9 | H10 |
| C1 | X | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| C2 | 1 | X | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| C3 | 0 | 1 | X | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| C4 | 0 | 1 | 0 | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| H1 | 1 | 0 | 0 | 0 | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| H2 | 1 | 0 | 0 | 0 | 0 | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| H3 | 1 | 0 | 0 | 0 | 0 | 0 | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| H4 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | X | 0 | 0 | 0 | 0 | 0 | 0 |
| H5 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | X | 0 | 0 | 0 | 0 | 0 |
| H6 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | X | 0 | 0 | 0 | 0 |
| H7 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | 0 | 0 | 0 |
| H8 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | 0 | 0 |
| H9 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | 0 |
| H10 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | X |

Above shows my first idea.

Starting at c2,c1 with the chain length of 2 the checker would move to the cell to the right and check if it has a “0”if it does then it would check the cell below it and if it has a one then it would check the “1” on its right and if there was a “0” it would go down unless it is the second “0” the code would stop and out put the number of times the program moved one cell. This initially looked like it was good however it didn’t consider the possibility on methane-based molecules with one carbon. It also didn’t allow for the first carbon to not be the first in the chain.

And to fix the positioning problem I used Butane and started at the second carbon this was the adjacency matrix that it produces.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | C1 | C2 | C3 | C4 | H1 | H2 | H3 | H4 | H5 | H6 | H7 | H8 | H9 | H10 |
| C1 |  | 1 |  | 1 | 1 | 1 |  |  |  |  |  |  |  |  |
| C2 | 1 |  | 1 |  |  |  | 1 | 1 |  |  |  |  |  |  |
| C3 |  | 1 |  |  |  |  |  |  | 1 | 1 | 1 |  |  |  |
| C4 | 1 |  |  |  |  |  |  |  |  |  |  | 1 | 1 | 1 |
| H1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| H2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| H3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| H4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| H5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| H6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| H7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| H8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| H9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| H10 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

First program second idea

If I had followed the initial algorithm would have outputted a chain length of 3 which is the wrong output

My second idea was to check the cell above and below and follow the first one. This would have outputted the correct output of 4 however this didn’t consider alkyl groups, so you could ask “why don’t you check both the lines from carbons attached to c1”. If you asked this, then you would be wrong as it assumes C1 lies on the longest path.

At this point I started to look at graphs and traversal techniques.

Now I have decided on a solution to the problem. I will set a root carbon. Then I would make a list of all the different routes that could be taken without going back on itself. Then I would compare lists and I would find the longest chain and then compare it to the second longest and find the similar carbons then add to the list of it makes the chain longer from the point of the last common carbon. This will repeat until all the combinations have been tried, then an output will be given.

I plan to achieve this algorithm by using lists of lists to store all possible paths then compare them all get every possible merge then find the one with the longest chain. I will also convert the input to be a string made up of ‘C’,’(‘ and ’)’. This would simplify the solution to recursion and linear search.

A diagram of a network

Description automatically generated

This diagram shows the depth first traversal of the adjacency matrix to find all the paths. An example of this would be

A diagram of a diagram

Description automatically generated

[1] route with branches pending (3)

[1,2] current working route

The algorithm will start by storing the first node/carbon. It will then see how many neighbours it has that aren’t already in the current path. Then it will choose 2 and follows down the path.

A diagram of a network

Description automatically generated

[1] route with branches pending (3)

[1,2,4] current working route

The code will then again look for neighbours and then follow its path. The only neighbour is 4 so it adds 4 to the list.

A diagram of a diagram

Description automatically generated

[1] route with branches pending (3)

[1,2,4] route with branches pending (6)

[1,2,4,5] current working route (route ended so it is stored)

The code then searches for neighbours (5,6)

The algorithm will then pick the first one and then follows it down and then adds it to the list.

5 has no valid neighbours so it will store the full list into another list that stores all the possible

A diagram of a diagram

Description automatically generated

[1] route with branches pending(3)

[1,2,4] route with branches pending (all neighbour routes explored)

[1,2,4,6] current working route (route ended so it is stored)

1,2,4 will be returned as the original route and it will then follow the second valid neighbour and making a list of 1,2,4,6. This list is then stored into the list of lists

A diagram of a diagram

Description automatically generated

[1] route with branches pending (all neighbour routes explored)

[1,3] current working route (route ended so it is stored)

The original list is then returned to its state when the last fork was. Aka 1. then follows the next neighbour 3 to make the list 1,3 there are no neighbours it will add the list to the list of routes found.

Routes found:

[1,2,4,5]

[1,2,4,6]

[1,3]

I will then run a program to merge these lists by removing all but the last matching elements. Giving an output of:

A diagram of a process

Description automatically generated

1,3

1,2,4,5

1,2,4,6

5,4,6

3,1,2,4,6

3,1,2,4,5

The code will then find the longest path this will be the longest path. It will also check for other paths the same length as the longest path to save for later verification. (This extra verification is another case of preparing the code for the possibility to be improved and adjusted implementation later on)

# Testing

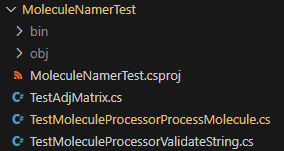
I used Test-driven development. This means that I created test cases where I then made the code that gave the right outcome. To do this easily I used a feature of Visual Studio Code C# dev kit (testing) this allows me to write code to automatically implement the required test cases into the code without human input. This sped up the development process it meant I didn’t have top build and run the code every time by hand, and I could test my code with one click of my mouse.

// video script

|  |
| --- |
| An introduction or overview |
| Test plan   * Typical data * Erroneous data * Boundary (extreme) data |
| Screenshot evidence |

## Test Case Structure

The MoleculeNamerTest module holds all my test cases used in my development.



Some example test case code:

A screen shot of a computer program

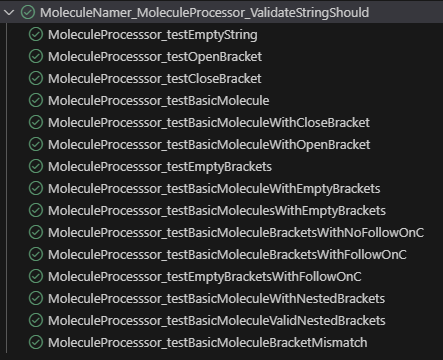
Description automatically generated

## Test Case Titles

### Validate String Test Titles:

Taken from file: TestModuleProcessorValidateString.cs

This class makes sure the input validation is working correctly and no invalid inputs are being passed through before the input is processed.



### Adj Matrix Test Titles:

Taken from file: TestAdjMatrix.cs

This class makes sure the process is doing as it is supposed to after being given exact processing requirements.

A screenshot of a computer program

Description automatically generated

### Molecule Namer tests:

Taken from file: TestMoleculeProcessorProcessMolecule.cs

This class test the whole process is doing at is supposed to after taking inputs.

A screenshot of a computer

Description automatically generated

//testing section explain video location

## Git Hub Actions CI

//check with dad

When developing code, you don’t want to mess up previously working code so git hub has a feature called “actions CI” when you want to merge a branch of code you have been working on onto the main repository this feature is used. This feature will run tests on the code both the tests that I have made as well as the others (e.g. Does the code build). This prevents bad code messing up the project.

// Which file .github/workflows/dotnet.yml

A screenshot of a computer program

Description automatically generated

// screen shot and descriptions from the pipeline run

Example run: <https://github.com/ChonkyDonkey/NEA-Molecule-namer/actions/runs/8676291035/job/23790642456>

A black rectangular object with a white border

Description automatically generated

A black and white text

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer screen shot of a black screen

Description automatically generated

# Evaluation

// Students should consider and assess how well the outcome meets its requirements. Students should obtain independent feedback on how well the outcome meets its requirements and discuss this feedback. Some of this feedback could be generated during prototyping. If so, this feedback, and how/why it was taken account must be presented and referenced so it can be found easily. Students should also consider and discuss how the outcome could be improved more realistically if the problem/investigation were to be revisited.

# Log of research

Some of the sources used for inspiration and to support research include:

IUPAC

<https://iupac.qmul.ac.uk/BlueBook/PDF/P1.pdf>

<https://www2.chemistry.msu.edu/faculty/reusch/virttxtjml/nomen1.htm> (08/04/24)

Researched molecule Namers:

Marvin

<https://www.chemspider.com/StructureSearch.aspx> (08/04/24)

<https://web.chemdoodle.com/demos/iupac-naming> (08/04/24)

<https://pubchem.ncbi.nlm.nih.gov> (08/04/24)