Problem Set 3

Contents

Python - Molecules as Graphs

So far in this course, we've talked about running molecular mechanics and quantum chemistry simulations of molecules. But, how do you store information about molecules in databases? Or, if you have a set of molecules, how do you determine which molecules are similar to one another? When you have a large amount of data, visual inspection to determine similarity or equivalence is impractical. It also requires high skill and may be prone to errors. Thus, for cheminformatics or machine learning applications, we have to be able to express a molecule mathematically.

Many of the representations for small molecules that are used in cheminformatics and drug discovery are based on representing molecules using graph theory. There are algorithms and molecular representations that build on top of this concept.

A mathematical graph is made up of "nodes" or "vertices" (circles in the image below) and "edges" which connect nodes (line between circles in the image below).

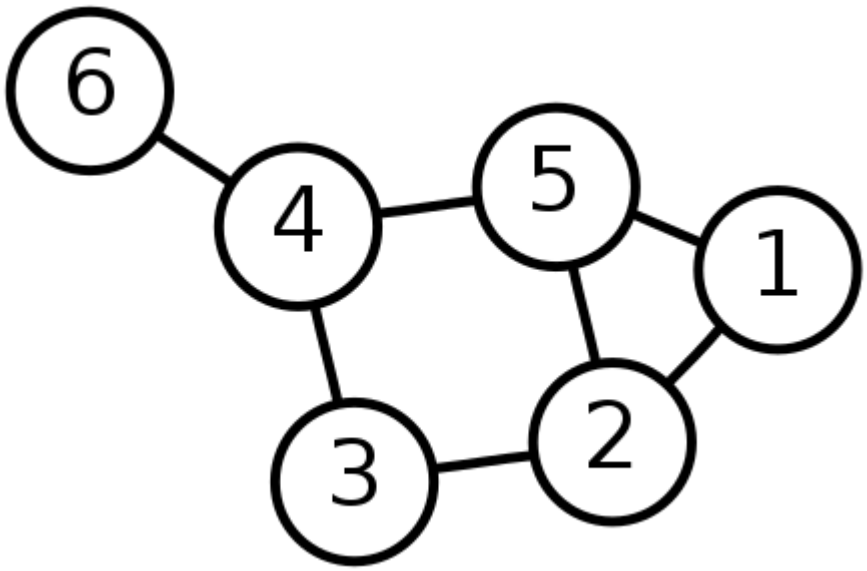

Figure 1 - A depiction of a graph with six nodes and seven edges. source

When molecules are represented as graphs, the atoms are represented as nodes in the graph, and the bonds are represented as edges.

Your task for this homework is to use Python to represent molecular information read from an sdf file as a graph. You have been provided with a function called `parse_sdf` in `read.py` which will return a list of elements and bonds from an sdf file.

You will use a Python library called NetworkX to create a graph from molecular information read from an SDF file. You can then use graph functions to determine things about the molecular structure, such as the presence of rings.

Your code should create a graph using the information from the sdf file. You should then print out the number of rings in the molecule and the number of atoms in each ring. You should also use NetworkX to create a visualization of the network and save it to a file.

You might find it useful to work in a Jupyter notebook for this homework, but **your final code should be turned in as a .py file**.

Here are some suggested steps:

1. Install NetworkX in a new environment with Python 3.10. The networkx documentation suggests installing using pip, but I found installing from conda to work better for me.
2. Read the NetworkX documentation to learn how to create a graph using NetworkX. Create a graph with no nodes.
3. Use object introspection to read about the `add_node` method and the `add_edge` methods of the graph objects.
4. Use NetworkX to create a graph for a provided molecule.
5. Analyze the number of rings in the molecule and the ring size.
6. Create a visualization of the network.

Questions

Answer these questions in your `README.md`.

1. What is an important feature of a NetworkX node? What data type did you choose to represent a node, and why? If you did not include information about the atom identity in your node, what type of Python data type could you have used to do that?
2. Use an object introspection technique on the NetworkX graph object. What method(s) did you use, and what did you learn about the object?
3. Use PubChem to get an SDF file for a molecule of choice and use your code to analyze it. What molecule did you choose and why?
4. **Bonus** - How could you color the nodes in the network based on the element of the atom? Implement this in your code.

Files

Include the following files in your repo:

1. Your code which can create a NetworkX network from information in an SDF file.
2. A `README.md` which explains the repo purpose and how to run the code in yourproject.
3. A `Makefile` with the following targets:

1. `environment` - creates the Python environment needed to run your code.
2. `analyze` - analyze your molecule of choice.
3. `clean` - remove images from `analyze`
4. `lint` - run black and flake8

If you're interested in learning more about molecular representations, you might consider checking out the following review: "Molecular representations in AI-driven drug discovery: a review and practical guide"

C++ - Generic printing function

Write a (templated) function that takes in an std::vector containing any type, and then loops over it and prints all the contents, with each element of the vector on its own line.

Overloading the stream insertion operator

In C++, you can overload the stream insertion operator (`<<`). You can do this by writing a function that takes in two arguments, one being the output stream object, and the second argument being what is going to be inserted. In this case, the first argument is a generic `std::ostream` (output stream) object, and the second is what you want to print.

We will cover streams in a little more detail in Week 5. However, we will just say now that `std::cout` is a type of `std::ostream`. Therefore, by overloading this operator we can say `std::cout << vec`.

The function should return the `std::ostream` object that was passed in. This is what allows chaining calls to the operator `<<`. Of course, this function can be templated!

```
std::ostream & operator<<(std::ostream & os, ...)
{
    return os;
}
```