**Problem introduction**

In 2015, Mark Burnett released 10 million passwords people use to access online accounts. Your job is to download the passwords file from [this link](https://xato.net/today-i-am-releasing-ten-million-passwords-b6278bbe7495), and write a Python 3 program that finds the 20 most used password.

To accomplish this, your program must create a linked list to store all of the passwords contained in the file. Your linked list must not contain any duplicates, so each node must store the following values:

* + A unique password extracted from the file (string)
  + The number of times the password appears in the file (integer)
  + A reference called ‘next’ that points to the next node in the linked list

You are required to code two different solutions to create this list:

* + Solution A: Every time you read a password from the file, check (using a loop) if that password has already been added to the linked list. That is, you need to traverse the linked list to see if that password has been added already. If the password is already in your linked list, update the number of times the password has been seen in the file. Otherwise, add a the password to the linked list.

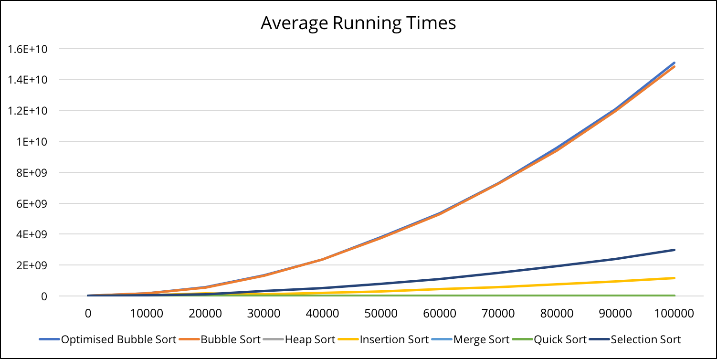
Solution B: This is a variation of Solution A. Instead of traversing the linked list to check if a password has been seen before, we will be using what is called [a dictionary](https://docs.python.org/3/tutorial/datastructures.html#dictionaries)

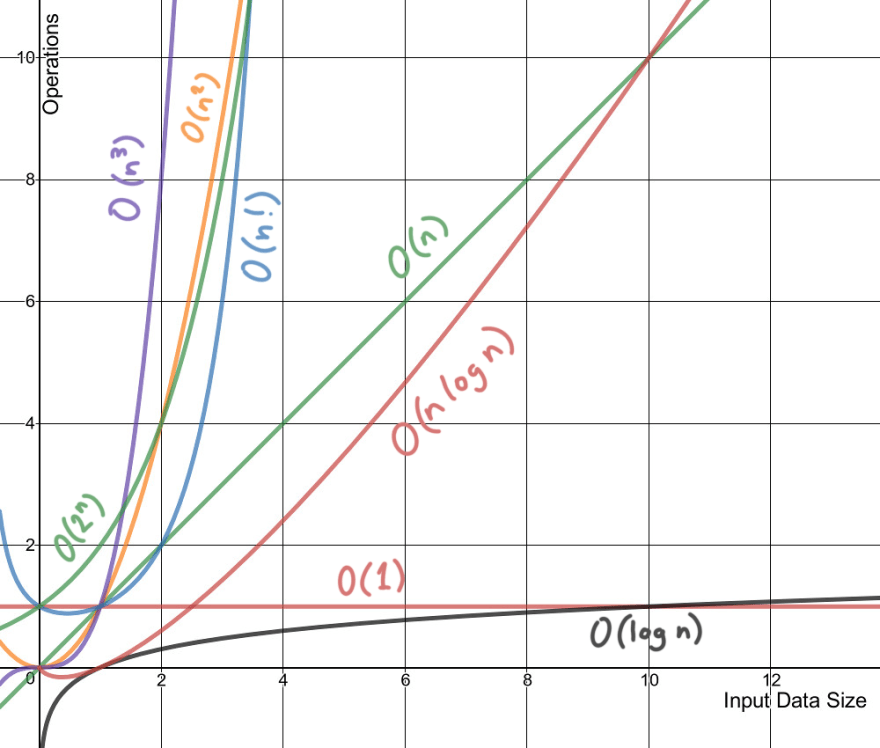
**Solution design and implementation**

I used the linked list class that was already given to us by Diego. Most of the methods were not needed for this specific lab. As far as formatting the code, I only had to add an extra method outside of the main linked list class. The read\_file function reads every single line in the given password file, extracts the password only since the username is irrelevant. For every password, it checks if it’s already in the linked list, if it is, it adds 1 to the count, if it is not, it adds it with a count of 1. It does the same thing with a dictionary.

**Experimental results**

Since the big file given to us was way too big for my computer to handle, I used a very small portion of it that includes about 172 passwords. It’s easy to see that the code is working just by looking at the password files and looking at the counts of each password. Another way to see that the code is working is that both the linked list implementation and the dictionary implementation give the same results. The linked list way of checking runs in approximately O(n) since it has to traverse the linked list linearly to find a password. The dictionary runs in constant O(1) time since it just checks if the password is there.





Dictionary

Linked list

**Conclusion**

The dictionary is a way better data structure to use for this problem since it is much faster at checking for a key and it is way less code. I learned how different data structures can do different operations in various time complexities. It also helped me learn the importance of using the right data structure for a certain problem.

**Source code**

<https://github.com/lebriones/CS2302/blob/master/Lab2.py>