INF-1101

Obligatory assignment

Magnus Dahl-Hansen

March 2020

1 Introduction

In this assignment I describe how I implemented two abstract data types that could perform different functions on some given elements/sets. I also describe how I implemented a code that determines whether a given mail is considered spam or not spam.

1.1 Requirements

- First requirement: implement an ordered set and its functions correctly. The set should support basic functionalities such as creating and destroying a set and sorting, adding and checking for elements. In addition to these basic functions, it also should support taking the union, difference and intersection of two sets. To perform these actions correctly an iterator structure must be made to iterate over elements and return them.
- Second requirement: implement an array that supports the same functions as the set, as well as an algorithm for sorting elements.
- Third requirement: implement an algorithm that determines whether a given mail contains words known to be spam or not.
- Last requirement: Do a performance evaluation of the two different set implementations.

2 Technical Background

Some of the topics covered in this assignment are abstract data types, lists, sets and arrays.

Abstract is something that's not physical but more like a thought or idea, which is kind of what an abstract data type is. It's an abstract type of model that's defined by how it behaves when using a set of values and/or operations.

A set and a list are not so different from each other. I quote from Wikipedia that says "*A list is an abstract data type that represents a countable number of ordered values, where the same value may occur more than once"*. Though a set is "*a collection data structure that stores certain values in a way that values are not repeated".* So, I understand that they are sort of the same thing just that the set cannot contain more than one of the same elements, whilst a list can.

An array is a data structure which can hold a collection of values/variables. An array can store variables as a name and all the variables can be accessed by referring to an index number.

3 Design/Implementation

The first thing I did was to implement the set structure which contains a list, a size and a compare function for comparing variables. This set contains several different functions, for example like creating a set, destroying it, adding elements to it, check for already existing elements and taking the union, intersection and difference between two different sets. It also contains an iterator structure that can create an iterator, destroy it, iterate over elements in a set and return them. When I implemented these functions, I made good use of the already existing functions in the "linkedlist" file.

Implementing the union, difference and intersection functions was a bit more complicated, but I will explain how I solved them. The union function copies all the elements from a set B into a new set C. Then I iterate over the elements in set A and checking if they are different from the elements in set C. If they are different, they are added to set C. The intersection is implemented in a way that checks whether elements from two sets are equal. If the elements are equal, they are added to a new set, if not equal the algorithm iterates to the next element in the first set, comparing it with the first element in the second set. The elements are iterated over so that all the elements in each set is compared to each other. Lastly the difference function iterates over all the element from set A and checking whether this element is different from one in set B, if this is true the element is added to a new set and so on.

The array is implemented almost in the same way as the set, but there are some differences. Defining a maximum size for the array was the first step, assuring that it does not use to much of the memory capacity. Should there be use for more memory such as in the add function, I simply multiply the maximum size by two.

The union, intersection and difference are implemented the same way as the set, but in addition I had to implement a sort function to sort a given set so it can be iterated trough. I implemented the selection sort algorithm which compares two values, checking which one is the lowest to then replace them so that the lowest appears first in the array. The algorithm moves on to compare the next elements, switching their positions until the given set is sorted.

Lastly, I implemented the code for determining whether a given mail is spam or not. This algorithm opens the mails known to contain spam-words and puts all the words into a new set. The words are then compared to each other and all the words than appears more than once are put in a second set. After this step, all the mails known not to be spam are tokenized and all the non-spam words are put inside a unique set.

Now that we have two sets, one containing common and spam words and another containing only common words, an algorithm compares these two sets and removes the common words from the spam words, so that we are left with only words known to be spam. Then the unknown mails are tokenized one by one and each word is compared with the spam words. So, if a given mail contains a spam word it will be revealed as a spam mail and how many spam words it contains.

4 Discussion

When I run the *spamfilter* program with both the set and array on a Linux system the results are correct, marking mail number 3 to contain one spam-word and mail number 4 to contain two spam-words. However, running the program on my Windows laptop, the outcome differs. The result is that all the mails are marked as spam… I think this is a weird case and I’m not sure where the problem lies yet.

When I compared the two implementations, I found that the set uses on average 2,405 seconds to complete the spamfilter, while the array took 3,2 seconds. I think the sorting function in the array is slower than the one in the set and that's why the array is slower. The array uses the selection sort algorithm which must complete more steps to sort elements than the mergesort algorithm used in the set. Selection sort compares two values and change their position if the first value is bigger than the second value. The algorithm must iterate over elements, compare them and change their position over and over until the set is sorted. This takes a lot of time compared to the mergesort algorithm which divides the set into multiple arrays until there is only single values in each array, then compares them and changes their position. This is better because it doesn't have to iterate and compare each value over and over. Honestly, I knew that selection sort is a slower sorting algorithm than mergesort, but I chose to implement this one because I struggled with this assignment it was the easier one to implement.

5 Conclusion

In this report I described how I implemented an ordered set, the array and the spamfilter functions. I have learned a lot about writing code in C and how lists, arrays and different sorting algorithm work, but I still feel I lack the knowledge to write a good complementary report. However, I managed to complete the assignment and run the program correctly.

6 References

- https://stackoverflow.com/questions/13694034/is-a-python-list-guaranteed-to-have-its-elements-stay-in-the-order-they-are-inse
- http://malloryfeuer.net/selection_sort_vs_merge_sort
- https://en.wikipedia.org/wiki/List_(abstract_data_type)
- https://en.wikipedia.org/wiki/Abstract_data_type