Comp 2000: Data Structures, Fall, 2017

Group 36

Scott Austin

Erich Hauntsman

Jack MacVicar

…

Assignment 1: Bag Application: Spell Checker

Due: M 9/25/2017

**Analysis**

1. Our application reads in a .txt file and spell checks it against a dictionary .txt file. We find any misspelled words in the .txt file and then display these words, if any, to the user in the console. We used delimiters to separate words from punctuation marks and we stored the dictionary words in all lower-case and checked our words using .toLowerCase() to ensure capitalization did not affect the results. Coordinating all our code to work together was our main issue with completing the application. It was difficult to communicate what each component was supposed to do before we had all the pieces meant to fit together. Our solution was to meet in person and talk over our code integration a couple times until we got it set up correctly.
2. A bag of groceries is a close to being accurately represented by the bag ADT, because there can be many grocery items of the same kind in a bag and the order does not matter. There are some issues that make it inaccurate, however. For example, the bag ADT does not implement any heuristics about the data so logic such as keeping heavy items away from light items, for example, must be handled outside the bag by the user.
3. The bag ADT is not accurate for representing a dictionary because a bag could have multiple instances of the same word, which does not make sense for a dictionary. Also a dictionary is sorted, whereas a bag in unordered list of items, so we lose some properties of a dictionary by not being able to see use the words in a sorted list.
4. A bag is not a useful ADT in general, because it’s unorganized and unrestrictive with very limited use cases.
5. A fixed-size array bag implementation would be an issue if the dictionary file has more words in it than the size of the bag we set up for it. Also, we do not know how big our misspelled words bag will be before running the spell checker so a fixed-size array would limit how many misspelled words we can store arbitrarily. The linked-list implementation and the resizable-array implementation do not have these issues.

**Summary**

1. **About your team:**
   1. We divided the programming tasks between the three of us so that each of us had an equal amount of code and that we all developed something that had us coding with the spell checker and with bags.
   2. We met in person to coordinate our tasks and fix integration issues and we used github to upload our files for the rest of the team to use.
   3. Working around each other’s schedules is tricky since we all have different schedules and coordinating with people we don’t know very well can be difficult at times.
2. **Where did you have trouble with this assignment? How did you move forward? What topics still confuse you?**

We had trouble integrating all our code together and communicating our expectations of one another. We met in person multiple times to divvy out our tasks and to work out our implementation to get everything functioning. We’re still confused on how to properly get all of our small parts coordinated so that they fit into the main application effectively.

1. **What did you learn from this assignment? (Please be specific)**

We learned how to work in a team on a small project. We learned how to develop an application using the Bag ADT, giving us an example of a use case for bags.

1. **How could this assignment be improved in the future?**

It would be nice to have more ideas or options for what we can develop with a Bag ADT.