Searching for Food

Nicolai Banke

Codecademy

2019

Searching for Food Types

- To search for the food types I implemented the Trie data structure
- This structure allows for fast lookup and is a natural way to store words, since any number of words sharing the same prefix of letters can occupy the same set of trie nodes
- Searching for a word in a trie has worst-case runtime $\mathcal{O}(m)$ where m is the number of letters in the word
- This is in contrast to an unsorted array, sorted array or a linked list where the worst-case runtime is respectively $\mathcal{O}(N)$, $\mathcal{O}(\log N)$ and $\mathcal{O}(N)$, where N is the size of the data structure

Searching for Food Types

- The worst-case runtime for searching in an imperfect hash table (i.e. including key collisions) is $\mathcal{O}(N)$, but is typically the same runtime as in the worst-case for a trie
- Since the worst-case lookup time in a trie is reduced to the size of the string being searched, it is hard to imagine a more efficient data structure

Searching for Restaurants

- When a food type has been picked, we need to find restaurants serving that type of food
- To structure the restaurant data, I inserted the food types as keys in a hashmap
- Each key mapped to a linked list with restaurant info contained in each node
- To this end I added a method to the hashmap class that checks if a key is already contained in the hashmap, so that if the key already exists, the linked list with that key can be retrieved and a new node added

Searching for Restaurants

- The hashmap in this case is the more efficient data structure, since the worst-case runtime for searching a linked list is, as mentioned, $\mathcal{O}(N)$, where for a hashmap we get constant runtime $\mathcal{O}(1)$, and it therefore seems to be the fastest possible runtime
- Once the type of restaurants has been found we want to print out all of the content of the linked list instead of searching, so no further considerations about runtime need to be made

Further Uses for Data Structures

- An obvious use of the trie data structure would be, as we have seen in this project, auto-completion features such as those used in search engines
- Another use is for spell-checking, where a slightly misspelled words will have a very similar path to the correctly spelled word
- For sorting algorithms, using a trie can can allow for a radix sort-like algorithm, since a pre-order traversal of the tree will result in an output that is in increasing order