Soho Restaurants

Capstone project for Computer Science Basics: Data Structures

1. *Which data structure(s) did you use for part 1? Why did you select these data structures?*

   While the list of food types (cuisines) could be dispensed with altogether, performing partial string searches for cuisines would require program logic that extracts the cuisine information from the data structures storing the restaurant data. Also, such search login would likely be computationally expensive. For these reasons, i chose to implement a Trie data structure.

2. *What is the runtime (in asymptotic notation) of searching for a food type? Do you think there is a more efficient runtime?*

   My Trie class implements two search methods: a full-text match (`Trie.get()`) & prefix search (`Trie.find()`). *Trie.get()* has a linear runtime — O(n) — as one operation must be performed for each character in the search string. The runtime of *Trie.find()* is O(m*n) as the worst case (returning all terminal nodes in the Trie) must traverse every node to the length of the longest word (n) and the width of the widest level of the tree (m).

3. *Which data structures did you use for part 2? Why did you select these data structures?*

   Overall, the restaurant data is stored in a hashmap of cuisines (food types), each node being a linked list of restaurants, each restaurant node being a hashmap of restaurant data.

   The hashmap of cuisines allows for easy retrieval of the list of restaurants by passing in the user's chosen cuisine.

   As there was no need to structure the list of restaurants, a linked list of all of the restaurants serving a given cuisine was sufficient.

   For the individual restaurant data itself, a linked list of data fields would also have sufficed, but given that the restaurant data is structured a hashmap would allow for retrieval of arbitrary data without knowledge of the internal storage structure. Being able to access restaurant data-points by name is easier and more flexible than accessing those data-points positionally.

   An alternative is to render the individual restaurant data as static text: this would reduce the complexity of the data structures storing the restaurant data at the expense of flexibility and with little real gain in runtime efficiency (see below).

4. *What is the runtime (in asymptotic notation) of retrieving the restaurant data? Do you think there is a more efficient runtime?*

   - Retrieving the list of restaurants from the hashmap is constant: O(1)
   - Retrieving each restaurant from the linked list is linear: O(n)
   - Retrieving the restaurant data from the hashmap is constant: O(1)
   - Total complexity of restaurant data retrieval is linear: O(n)

Soho Restaurants (Capstone project for Computer Science Basics: Data Structures)

5. *Outside of this project, what are other innovative ways you can utilize data structures?*

   Trie data structures have obvious uses in applications such as autocomplete (like in smartphone keyboards) and hyphenation tools (found in word processing applications, for example).

   Hashmaps form the foundation of dictionaries and other key:value pair data structures. As such, they can be used to store data in a more structured, easier to access way than is available in linked lists. For example, one could track the number of occurrences of a certain key by simply incrementing its associated value.

Submission by Dean Ransevycz <dean.ransevycz@gmail.com>