

SoHo Restaurants Data Structure Overview

Presentation by Karen Liu





Part 1: Choosing a Data Structure to Search for Food Types

- HashMap data structure was chosen as it is the quickest to search through with a runtime of $O(1)$ after the data is organized into the array.
- Needed a way to sort the keys and link them with their corresponding food types values:
- Key: first letter of the food type
- Value: food type



- Created a function to create a list of keys
- Created a function to create a list of values based on the key

Example:

```
HashMap = { 'a': ['american', 'african'],  
            'c': ['chinese', 'czech', 'cafe'],  
            'j': ['japanese']}
```



Runtime of $O(N)$

Function to extract the first letter of each food_type and put in a list for use in part 2. No duplicate keys are added.

```
def char_keys(lists):  
    char_keys = []  
    for item in lists:  
        if item[0] not in char_keys:  
            char_keys.append(item[0])  
        else:  
            pass  
    return char_keys
```



Runtime $O(N)$

Used a simple list to store the values

```
##create a list of values for keys starting with the first letter
#if there is more than one value, it is added to the values list
def find_values(key,value_list):

    key_value_list = []

    for item in value_list:
        if item[0] == key:
            key_value_list.append(item)

    return key_value_list
```



Runtime $O(N^2)$

- Hashmap is created by iterating through each key and then creating a list of values from food_types to be stored as a value.
- Runtime is $O(N^2)$ because there is a nested for loop in the code

```
food_types_keys = char_keys(food_types)
food_types_hashmap = HashMap(len(food_types_keys))
for i in food_types_keys:
    values_list = find_values(i, food_types)
    food_types_hashmap.assign(i, values_list)
```



This search function takes advantage of there being unique strings in the list to search. It matches only to the length of the input. Runtime is $O(N)$. Unfortunately, this function is built only for this project and will need to be refined when it comes to words that are less unique and require several refined searches to narrow down to the specific word.

```
def search_food_types(input_char, food_list):  
  
    word_list = food_list  
    len_list = len(word_list)  
    list_idx = 0  
    word_idx = len(input_char)  
  
    while len_list > 0:  
  
        if word_list[list_idx][0:word_idx] == input_char:  
            return word_list[list_idx]  
            break  
        else:  
            len_list -= 1  
            list_idx += 1  
  
    return("Option not found, please try again.")
```



Part 2: Choosing a Data Structure to Retrieve Restaurant Data


The same data structure in part 1 was applied to part 2:

A hashmap utilised the food_types as keys and a list of restaurants as values.

```
restaurant_data_hashmap = HashMap( len(food_types) )

for i in food_types:

    restaurant_values_list = find_values(i,restaurants_list)
    restaurant_data_hashmap.assign(i,restaurant_values_list)
```

By using a list to store the values in the restaurant, I used the indexing system to print the data out after retrieval.

It is not overly complex and has a runtime of $O(N)$.

```
def restaurants_sort_print(restaurants):  
    separator = "*****=====*****"  
  
    print("\n\nThe following restaurants found are:\n")  
  
    for item in restaurants:  
        print (separator)  
  
        print("\nName: {0}\n"  
              "Price: {1}/5\n"  
              "Rating: {2}/5\n"  
              "Address: {3}\n".format(item[1],item[2],item[3],item[4]))
```



Example of the run:

Overall Runtime is $O(N^2)$
based on the largest runtime
code written: hashmap generator

```
What type of food would you like to eat?
Type the beginning of that food type and press enter to see if it's here.
c
The food choice beginning with 'c' are: ['czech', 'chinese', 'cafe']
Please enter the first letters of the food you want.
```

```
ch
chinese
The food choice beginning with 'ch' is: chinese
```

```
Would you like to see the list of restaurants? y/n
y
```

```
The following restaurants found are:
```

```
*****
```

```
Name: Beijing Express
Price: 2/5
Rating: 4/5
Address: 38 Teutonic Ave.
```

```
*****
```

```
Name: Dragon's Tail
Price: 1/5
Rating: 4/5
Address: 8 Jasmine Rd.
```

```
*****
```

```
Name: Super Wonton Express
Price: 2/5
Rating: 1/5
Address: 223 Milliways Ave
```

```
*****
```

```
Name: Shandong Lu
Price: 4/5
Rating: 3/5
Address: 335 University
```



Considerations to improve the code

- Create the `search_food_types` function using the Trie method of string searching. This will allow for more robust checking.
- Use a linkedlist node system to store the values instead of a list system because lists require specific indexing to correctly access the data whereas nodes can be traversed.
- Refine the `user_input` check statements to account for
- Improve user interaction and print format



Things I've learned:

- Always test each line of code or function.
- Coding needs to be very specific and instructive or else the computer doesn't know what it's doing.
- Break tasks/instructions into functions
- Be patient when things don't work out
- There's still plenty to learn so I should enjoy the journey
- Coding is a challenge, but it's fun and rewarding

**THANK YOU
CODECADEMY**

