Christopher Schutte

Logan Mann

Data Structures Final Project:

Retail Based Customer Analytics/Recommendations Software

**Project Overview:** This project aims to solve the common problem in retail of being able to accurately and quickly pull up customer information and use this information to inform how you interact with them. More specifically, their recent purchases, their favorite product categories, and some recommendations for them. We aim to solve this problem using multiple Hash Tables, Linked Lists, and Max Heaps.

**Inventory Class and Hash Table:** The first problem to solve was where to store the available products in a way that is easily searchable and organized by product-category. We decided that a Hash Table was the best data structure to store this data for a few reasons:

1. Hash tables allow for quick searching, making many searches of inventory necessary for many of the functions faster.
2. Hash tables can be hashed with a particular string, the string we’re using is the product category, effectively organizing the products by their category and making sure that the linked lists used for the collisions are much smaller than a full linked list of every product.

For the actual products we used a data struct that contains: the product name, its category, its size, its color, number bought, and a pointer to the element in front and behind it. These data members were chosen because its all the information we need to know what they bought, if it’s been bought at all, and what to recommend.

This inventory Hash Table functions as both the available products list and the purchased list because the int that keeps track of number bought is also used as a bool to find out if said product has been purchased or not.

**Customer Class and Hash Table:** The customer class contains a modified product struct that eliminates any unnecessary data members, and a customer class. The customer class also contains a pointer to a max heap of the modified product structs that is used in the product recommendation function in the driver class and to keep track of customer purchases.

We used the max heap because the recommend function needs to know what the most bought product category is, so because this heap is organized by number purchased, knowing what the most popular category is, is as simple as calling the first member in the heap.

The Customer Hash Table is hashed using the string of the customers email, converted into a unique int and indexed to a fixed size array. This ensures that collisions are exceedingly rare but in the event of a collision the customer struct contains a linked list.

Again, a hash table was used for this problem because of the search functions needed to store individual customer data.

**Database (Driver) Class:** Lastly is the driver class which contains an Inventory Class, and a Customer Class. This class has functions that transfer information between the two and make for much easier debugging then if we had used one class for the whole problem.

**Inventory Hash Table:**

Fourth Product

Third Product

Second Product

[Category 1]

First Product

Second Product

[Category 2]

First Product

Second Product

[Category 3]

First Product

**Customer Hash Table**

Purchased Product Max Heap Head

Purchased Product Max Heap Head

[Email 1]

Customer 1 Information

[Email 1]

Customer 2 Information

Purchased Product 2

Purchased Product 2

Purchased Product Max Heap Head

[Email 2]

Customer 3 Information