Research Paper

Matthew Connors, Jack Reedy, Justin Keyes, Mithilesh Pabba

CSC 212: Data Structures and Algorithms

Johnathan Schrader

University of Rhode Island

04/15/2024

Introduction

For our class project, we have created a POS(Point-of-Scale) system used to simplify and track daily operations for businesses. In today's retail world, staying organized and efficient is the key to success. Our program aims to help with common challenges faced by retailers, such as scheduling conflicts, inventory tracking, and time management. By using interval trees, treaps, and bucket sort algorithms to manage and sort our data, we offer a comprehensive solution that streamlines information and enhances productivity. In our paper, we will first break down each of these three algorithms, how they work, why they are efficient, and how they benefit our overall program. Then we will go further into the strengths and weaknesses of our program, as well as the limitations we believe it has in its current state. We have included pictures and diagrams throughout this paper to better visualize our program's strengths. Now let’s take a look at each of our algorithms and figure out exactly how each one works.

Bucket Sort

Bucket sort, also known as bin sort, is a sorting algorithm. It distributes elements of an array into several buckets. Each bucket is then sorted individually by the bucket sort recursively. This sorting method is useful when the input is distributed over a range. For the POS system, bucket sort is used to manage the inventory stock numbers efficiently. Stock numbers often vary over a wide range and bucket sort is an ideal for this.

Each element in the input array represents a stock number for an item. The range of stock numbers is first determined to scale the bucket sizes. Buckets are created for the range of stock numbers. Each stock number is placed into a bucket based on its value. Once all elements are in their buckets, each bucket is sorted independently. Each bucket is then sorted individually by the bucket sort recursively. After sorting, the contents of all buckets are in order to form a single sorted array.

When the elements are distributed, bucket sort has a nearly linear time (O(n+k)), where n is the number of elements and k is the number of buckets. It can handle large numbers of items which is good for large inventory systems in retail environments. This algorithm integrates well with data structures like arrays and lists used in the software.

Interval Tree

The interval tree is a data structure optimized for the organization of intervals. Based around a binary search tree, intervals are inserted into the tree based on a comparison between the lower end of the inserted interval with the lower end of each necessary node in the tree. Lesser lower bounds are stored in the nodes left of the compared-with node, while those that are greater are stored in the right nodes. This allows the tree to be searched much more efficiently than a linear data structure. When searching for a given interval, one is able to eliminate half of the tree after each comparison, leading to a time complexity of O(log(n) + k), where k is the number of elements stored in the tree. Additional operations that are typically implemented are searching for overlapping intervals, as well as a function that calculates the other component of each node: a max interval value. After the insertion of each node, a function is called on each node which finds the maximum value of all intervals in the node's subtree, and stores the value as the other component of the node. This means that the interval tree is fundamentally a binary search tree with nodes that store intervals and maximum subtree interval values, rather than just a singular key.

Treap

The Treap data structure integrates the principles of both binary search trees (BSTs) and heaps, offering a balanced and efficient solution for storing and retrieving data. At its core, the Treap maintains two key properties to ensure its functionality: the Binary Search Tree property and the Heap property. These properties collectively guarantee that the data structure is both ordered and balanced, facilitating efficient search, insertion, and deletion operations.

In adhering to the Binary Search Tree property, each node within the Treap is organized such that the key of any given node is greater than all keys in its left subtree and less than all keys in its right subtree. This property allows for fast lookup of elements within the tree, optimizing search operations. Meanwhile, the Heap property ensures that each node maintains a priority value, typically assigned randomly. This priority value determines the position of the node within the heap structure and helps to maintain balance during insertion and deletion operations.

The structure of a Treap node encompasses several components: the key, which serves as the identifier for comparison and placement within the tree; the priority, a randomly generated value that influences the node's position within the heap; and pointers to the left and right children, which establish the hierarchical relationships between nodes. Searching within a Treap involves traversing the tree from the root downward, comparing the target key with the keys of each node encountered along the path. This recursive process continues until the desired key is found or until a leaf node is reached, indicating the absence of the key within the tree. The efficiency of the search operation is largely contingent on the balanced nature of the Treap. Insertion into a Treap begins with the standard procedure of locating the appropriate position for the new node based on its key, adhering to the rules of the Binary Search Tree property. Once inserted, the new node is assigned a random priority value, which may necessitate adjustments to the tree structure to maintain both the Binary Search Tree property and the Heap property. These adjustments often involve rotations, which reorganize the tree while preserving its ordered and balanced nature. Similarly, deletion from a Treap entails locating the node to be removed and then adjusting the tree structure as needed to maintain its properties. Upon removal, rotations may be performed to ensure that the resulting tree remains both a valid BST and a valid heap.

Rotation operations, such as right rotation and left rotation, are instrumental in maintaining the integrity of the Treap structure. Right rotation involves promoting a node's left child to become the new root of the subtree, while left rotation promotes a node's right child to the root position. These rotations facilitate the restructuring of the tree while preserving its ordered nature and balance. In terms of time complexity, the average-case performance of search, insertion, and deletion operations in a Treap is O(log n), where n represents the number of nodes in the tree. However, in scenarios where the tree becomes highly unbalanced, the worst-case time complexity may degrade to O(n). Nonetheless, the random assignment of priorities typically results in a well-balanced tree, ensuring efficient operation in practice.

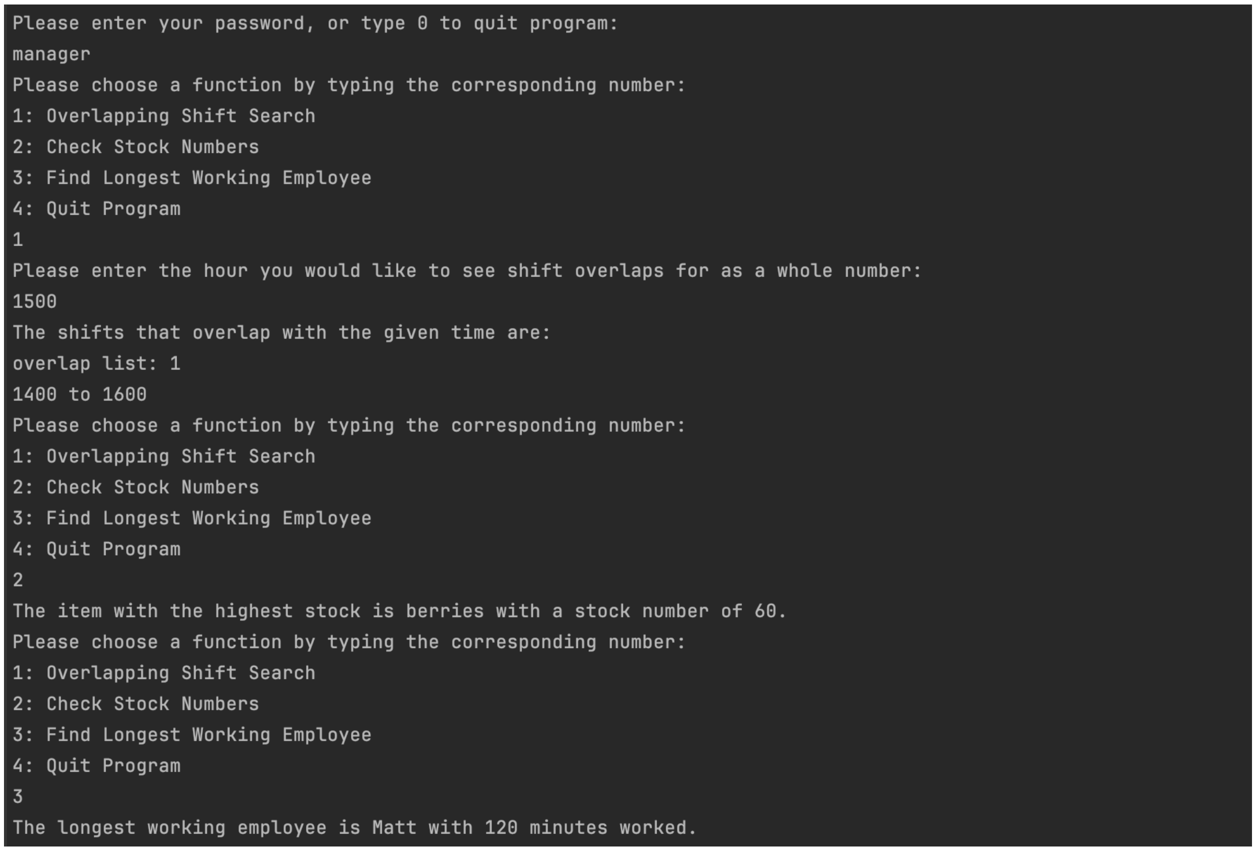
Program Usage

Usage of our POS system is quite simple, when you start up the program you'll get a message letting you know the system is ready, then from there you move on to either tracking an employee's weekly hours or to monitoring stock numbers. For employee's you simply enter their name, shift, and total hours worked, the program then stores that information for you to either use or come back to later. For stock information, you simply enter the name and the total stock information, from then on all that information is stored and available whenever you need it.

Functions to find the shifts that overlap with a certain time, the item with the highest stock number, and finding the employee with the most hours worked is also available to the manager.

Implementation

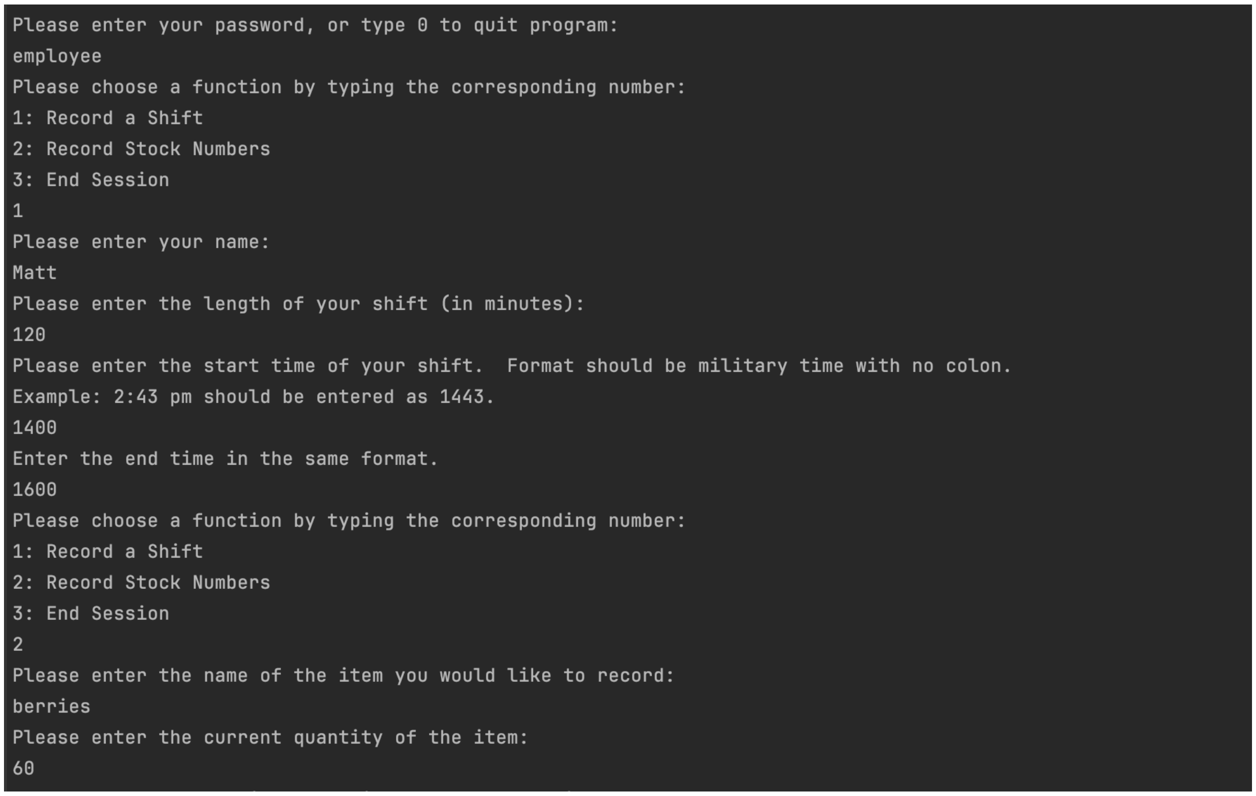
The program's implementation starts with the main file. Upon running the program, it prompts the user to enter a password. Depending on this information, which is hardcoded into the program, it enters one of two modes: manager or employee. If the manager mode is the one selected, then the user is provided with three options for functionality: searching for shifts that overlap with a given time interval, finding the item with the highest stock number, or finding the longest working employee. If the manager selects to search for overlapping shifts, a new prompt is generated that asks for a time to search for. Once the required data is inputted, the program searches the interval tree for any intervals that overlap the one given, then returns them to the manager. If the manager instead opts to check stock numbers, the program returns the item with the highest priority in the treap that they are stored in, which is also the item with the highest amount of stock. If the manager would like to see the longest shift worked, they simply need to select that option and the corresponding entry from the sorted vector is returned to the console. Finally, if the manager decides to add a new item, then the terminal prompts the user for certain information about the item, then creates a new item with the information.



```
Please enter your password, or type 0 to quit program:
manager
Please choose a function by typing the corresponding number:
1: Overlapping Shift Search
2: Check Stock Numbers
3: Find Longest Working Employee
4: Quit Program
1
Please enter the hour you would like to see shift overlaps for as a whole number:
1500
The shifts that overlap with the given time are:
overlap list: 1
1400 to 1600
Please choose a function by typing the corresponding number:
1: Overlapping Shift Search
2: Check Stock Numbers
3: Find Longest Working Employee
4: Quit Program
2
The item with the highest stock is berries with a stock number of 60.
Please choose a function by typing the corresponding number:
1: Overlapping Shift Search
2: Check Stock Numbers
3: Find Longest Working Employee
4: Quit Program
3
The longest working employee is Matt with 120 minutes worked.
```

The above image shows an example of the functions of the manager mode.

If the employee mode is the one selected, then two options are provided to the user. Either a shift start and end point can be recorded, or stock numbers can be inputted. If the user chooses to enter their shift, then the terminal prompts them for a start and end time. This prompt also includes specifics about how the times should be entered, in military time without punctuation. It also asks for their name, and the length of their shift. Once this information is inputted, a new interval is added to the shift interval tree. The length of the employee's shift is also added to their total hours worked. Because of this change, the vector which stores the employee structs must be sorted again (using bucket sort) in order to maintain its sorted state. If the user instead chooses to enter stock numbers for an item, then the terminal prompts them to enter the name of the item, and then the number in stock. The program then modifies the treap which stores the stock numbers in order to first insert the item's stock number, and then ensure that the integrity of the treap has been maintained. For both of the possible modes, the menus include both the functions described above as well as a "quit program" option. This option, as well as the possible functions, are all assigned an integer value to aid with selection.



```
Please enter your password, or type 0 to quit program:
employee
Please choose a function by typing the corresponding number:
1: Record a Shift
2: Record Stock Numbers
3: End Session
1
Please enter your name:
Matt
Please enter the length of your shift (in minutes):
120
Please enter the start time of your shift.  Format should be military time with no colon.
Example: 2:43 pm should be entered as 1443.
1400
Enter the end time in the same format.
1600
Please choose a function by typing the corresponding number:
1: Record a Shift
2: Record Stock Numbers
3: End Session
2
Please enter the name of the item you would like to record:
berries
Please enter the current quantity of the item:
60
```

The above image shows an example of the functions of the employee mode.

The entire microcosm depends completely on the correct implementation of an interval tree, treap, and bucket sort algorithm. The interval tree stores the shifts inputted by the employees, associated with the employee who inputted them. Using an interval tree for their storage makes it easy for the manager to search for overlapping intervals, in order to see which employees were working during a specific time period. Furthermore, the interval tree makes it easy to see the latest that an employee worked on any given day, since each interval node also stores the maximum value (latest time) present in the intervals of their subtree. The treap is integral in the stock-tracking functionality. Storing items with a priority of the number of them that are in stock makes it easy to access the highest-stock items, allowing the manager to quickly see which items do not need to be reordered. Finally, a bucket sort algorithm is used to sort a vector which stores employees based on the total number of hours worked. This allows the manager to quickly see who has worked the most number of hours, as well as how much they will have to pay them for their labor. Without these three data structures, the processes that the microcosm implements would be much more difficult, and less efficient.

Program's Future

Many POS systems found in the retail sector are very barebones, most times you'll see the screen that the workers use and it'll be like a glimpse from the early 2000's. While our POS is still in its early stages, we have the ultimate goal of providing a new and improved system for this sector, as well as possibly making it available to use in other fruitful business industries. We strive for the program to be simple and easy to use, as well as being just as easy to set up for business owners. Leave the 2000's behind and look towards the future of business. With our simple and easy to use tracking system, the worry of managing transactions and employee time goes away. There's a strong and thriving market to be seen in industries such as restaurants, grocery stores, salons, spas, hospitals, and more that all get a huge benefit from POS systems. Once we revolutionize the retail industry, the ability to do the same for such industries will become available.

Limitations

While the use of Interval Trees, Bucket Sort, and Treaps provides many strengths, they also have certain limitations.

Bucket sort's performance declines if the elements are not uniformly distributed. Skewed data can also lead to some buckets having more elements than others, which reduces the efficiency of the sort. Bucket sort requires additional memory for each bucket. If the range of input values is very large, the memory requirement for buckets can also become significant.

The efficiency of treaps depends on the randomness of priorities assigned to nodes. Poor distribution of priorities can decline the performance to a regular binary search tree. Treaps need careful balancing through rotations, which can add complexity to the maintenance of the data structure.

Updating the interval tree, like adding or removing intervals, can be complex and time-consuming, especially in an environment where intervals change frequently. Interval trees require additional space for storing metadata, like the maximum value in the subtree, which increases the space complexity compared to simpler data structures.

While these data structures give advantages in terms of efficiency and suitability for the retail POS system, there are limitations as well.

Strengths

In the fast-paced world of retail, efficiency is the key to success. Our program is designed to streamline operations using the algorithms and data structures: interval tree, treap, and bucket sort. Our program's largest strength, which I will go further into, is having these algorithms work together. By combining them, we have created a more efficient, yet simple Point-of-Scale system which can be used by the largest retail chains, to small mom and pop stores. First, let’s discuss the strengths of the algorithms individually in our program.

The interval tree is like a digital planner for employee shifts. It keeps track of who's working when, making it easy for managers to see scheduling conflicts and ensure adequate coverage. By organizing shift intervals alongside employee names, the interval tree simplifies the process of tracking employee availability and identifying gaps in the schedule. Additionally, it helps managers track employee hours, helping to ensure compliance with labor regulations. Complementing the interval tree is the treap, a powerful tool for managing inventory. Instead of just listing items, the treap sorts them based on their stock levels, prioritizing items that are running low. This makes it easy for managers to identify items that need to be restocked and ensure that popular items are always available for customers. By prioritizing items with lower stock levels, the treap helps minimize stockouts and maximize sales opportunities. The bucket sort algorithm is another valuable tool in our program. It helps organize employee data based on total hours worked, making it easy for managers to track employee performance and calculate payroll. By ordering employee data based on total hours worked, the bucket sort algorithm simplifies payroll processing and ensures accurate compensation for employees.

By integrating these tools together, our program offers an efficient POS system for retail management. The interval tree manages shifts, the treap handles inventory management, and the bucket sort algorithm organizes employee data. This integration enables managers to make informed decisions and streamline operations. Synergizing these tools enhances the program's effectiveness. For example, the interval tree's ability to manage shifts complements the treap's functionality in inventory management. By combining these tools, managers can optimize staffing levels and ensure that inventory levels meet customer demand. Whether you're managing a small boutique or a large retail chain, our program can help you succeed in today's competitive retail environment. With our program, you can simplify retail management and focus on growing your business.

Weaknesses

There are two main weaknesses with the microcosm that could be improved before a real-world application is implemented. First of all, the program in its current state has no database integration. For one, that means that data that is inputted on one device cannot be accessed on another, even if the program is run in the same conditions. It also means that the complex data structures that hold the data that is inputted are not maintained between sessions. This is certainly a weakness that could be resolved in a more fleshed-out version of the program, especially with the integration of a more sophisticated storage system.

The second weakness of the program is the lack of a GUI. Although this is by no means a necessary component of the program, a POS system with a graphical user interface is typically easier to use, especially for those unacquainted with text-based programs such as the one implemented. If this program had a GUI, it likely would consist of multiple interlinked pages with different functions. For instance, there would be a login page with boxes prompting necessary information, rather than a command line prompt to enter a username/password combination. Furthermore, a landing page would be present, where users could select the functions necessary, and then be redirected to a new page based on their selection. Finally, it would probably contain a visual representation of the data stored in the program. This could potentially be easier to understand than the solely text-based output of the command line program. This is certainly a weakness, as a GUI would make the program a lot more user-friendly.

Use Cases

Point of Sale (POS) systems are critical components in the operations of service and retail businesses. Some such businesses are restaurants, cafes, grocery stores, hospitality, salons, spas, healthcare facilities, fitness centers, and e-commerce. Many POS systems are used in relation to management and sales processing.

In relation to retail, POS systems are typically found being used in inventory management, sales processing, and customer management. Inventory management to track stock levels, which helps in ordering new stock, and also monitors product movements. Sales processing handles transactions, calculates totals, applies discounts, and processes payments. Customer management maintains a database of customer information, it tracks things such as purchase history, and manages loyalty programs. Additionally, these systems can be seen in grocery stores, doing the same jobs as in retail, however as well as that, they are capable of being integrated with scales for products that are sold by weight. As well as used for barcode scanning, which speeds up the checkout process and helps to manage long queues effectively. Along with things such as loyalty programs the systems are also seen in promotion and discount management, to automatically apply discounts or promotions for customers at checkout. Such systems are also found in e-commerce, they’re usually used for synchronizing inventory and sales data between physical and online stores. As well as getting customer insights by collecting data on the buying patterns of customers to tailor marketing campaigns.

Moving more deeply into the service industry, we can see these systems being integrated into places such as salons, spas, restaurants, cafes, hotels, and fitness centers. In salons and spas, POS systems are typically integrated with room booking systems to manage charges and checkouts. They're also found allowing customers to customize their services, by offering various service packages and addons which are usually selected directly from the POS system. When used in restaurants and cafes, POS systems can be used for order and table management, as well as allowing customers to split their bill by item or equally amongst them directly through the system. When being used for orders, the system sends orders directly to the kitchen from the service counter or table, reducing order errors and speeding up service in general. For tables the system manages table availability, and tracks the duration of guest stays. Hotels or places for hospitality in general typically use such systems for room booking management, much like the table management it manages room availability and check outs. Along with this it also is used to automatically add service charges to bills and facilities tipping. In fitness centers, the use cases can be found in membership management, class booking, and of course product sales. Typically fitness centers use a POS system to deal with memberships being renewed, canceled, or frozen. Another use is found in allowing members to book classes online or in person, as well as allowing members to purchase products like supplements or fitness gear. Finally, in the healthcare sector, POS systems are typically used to manage billing for treatments and process insurance claims. While also keeping inventory for Pharmaceuticals and maintaining records of patient appointments and treatments.